Nonconvex Optimization and Its Applications

Volume 89

Radosław Pytlak

Conjugate Gradient Algorithms in Nonconvex Optimization

Springer

Radosław Pytlak

Warsaw University of Technology
Institute of Automatic Control & Robotics
ul. Sw. Andrzeja Boboli 8
02-525 Warszawa
Poland

ISBN 978-3-540-85633-7 e-ISBN 978-3-540-85634-4

Nonconvex Optimization and Its Applications ISSN 1571-568X

Library of Congress Control Number: 2008937497

Cover design: WMX Design GmbH, Heidelberg

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Dedicated to Ela, Hania and Paweł

Preface

Conjugate direction methods were proposed in the early 1950s. When high speed digital computing machines were developed, attempts were made to lay the foundations for the mathematical aspects of computations which could take advantage of the efficiency of digital computers. The National Bureau of Standards sponsored the Institute for Numerical Analysis, which was established at the University of California in Los Angeles. A seminar held there on numerical methods for linear equations was attended by Magnus Hestenes, Eduard Stiefel and Cornelius Lanczos. This led to the first communication between Lanczos and Hestenes (researchers of the NBS) and Stiefel (of the ETH in Zurich) on the conjugate direction algorithm. The method is attributed to Hestenes and Stiefel who published their joint paper in 1952 [101] in which they presented both the method of conjugate gradient and the conjugate direction methods including conjugate Gram–Schmidt processes. A closely related algorithm was proposed by Lanczos [114] who worked on algorithms for determining eigenvalues of a matrix. His iterative algorithm yields the similarity transformation of a matrix into the tridiagonal form from which eigenvalues can be well approximated. The three-term recurrence relation of the Lanczos procedure can be obtained by eliminating a vector from the conjugate direction algorithm scheme. Initially the conjugate gradient algorithm was called the Hestenes–Stiefel–Lanczos method [86].

According to Hestenes [100] the concept of conjugacy was introduced in his joint paper with G.D. Birkhoff in 1936 in the context of the variational theory. At that time Hestenes also worked on the Gram–Schmidt process for finding conjugate diameters of an ellipsoid. However the latter work was never published; instead the concept of conjugacy was used by Hestenes to develop a general theory of quadratic forms in Hilbert space.

The initial numerical experience with conjugate gradient algorithms was not very encouraging. Although widely used in the 1960s their application to ill-conditioned problems gave rather poor results. At that time preconditioning techniques were not well-understood. They were developed in the 1970s together with methods intended for large sparse linear systems; these methods were prompted by the paper by Reid [184].

Although Hestenes and Stiefel stated their algorithm for sets of linear equations, from the beginning it was viewed as an optimization technique for quadratic functions. In the 1960s conjugate gradient and conjugate direction methods were extended to the optimization of nonquadratic functions. The first algorithm for nonconvex problems was proposed by Feder [64] who suggested using conjugate gradient algorithms for solving some problems in optics. His work referred to the earlier paper of Davidon [50] in which the prime concern was the updating formula for the inverse Hessian approximation. Then the convergence of several versions of a conjugate gradient algorithm for nonquadratic functions were discussed by Fletcher and Reeves [73], by Polak and Ribière [161] and by Polyak [162].

The work by Davidon on a variable metric algorithm was followed by that of Fletcher and Powell [72]. Other variants of these methods were established by Broyden [17], Fletcher [68], Goldfarb [83] and Shanno [190] creating one of the most effective techniques for minimizing a nonquadratic function. The main idea behind variable metric methods is the construction of a sequence of matrices providing improved approximation of the Hessian matrix (or its inverse) by applying rank-one (or rank-two) update formulae. These methods are also conjugate gradient algorithms when applied to a quadratic function and in that case they evaluate the exact Hessian matrix in a finite number of iterates. Thus variable metric methods have a special feature which is not present in other conjugate gradient algorithms. In the monograph we do not pay much attention to this feature of variable metric techniques since our main interests are algorithms for large scale problems. Variable metric approximations to the Hessian matrix are in general dense matrices and for that reason they are not suitable for problems with many variables.

This does not mean that I rule out quasi-Newton methods altogether since one of the main themes of the monograph are limited memory quasi-Newton methods which use a variable metric updating procedure but within the prespecified memory storage. These methods no longer guarantee the exact Hessian reconstruction in a finite number of steps yet they are conjugate gradient algorithms and are intended for large scale problems. The family of these methods was started by the work of Nocedal [144] and one of the most interesting techniques with limited memory feature are limited memory reduced-Hessian algorithms discussed by Gill and Leonard [81].

My interest in conjugate gradient algorithms started during my work on my doctoral thesis which focused on minimax optimal control problems. There is substantial evidence of application of conjugate gradient algorithms to solving control problems [7], [116], [155], [198]. At the time of my PhD work, we observed spectacular development of nondifferentiable optimization. Beginning from the influential paper by Clarke [32] on a subdifferential of a locally Lipschitzian function, efforts were undertaken to use a new concept of differentiability with the aim of developing algorithms for problems which do not have continuously differentiable functions. Lemaréchal [118] and Wolfe [209] proposed such algorithms for convex functions. They observed that their methods are conjugate direction methods when applied to quadratic functions. Therefore, it was natural to place the Lemaréchal–Wolfe algorithm among other versions of conjugate gradient algorithms. This was done in my

PhD thesis [172] and later presented in a paper [174]. Since at that time my interest in conjugate gradient algorithms was related to optimal control problems the early application of new versions of Lemaréchal–Wolfe algorithm were reported in the context of minimax optimal control problems [173, 178].

I showed that the Lemaréchal–Wolfe method is equivalent to the Fletcher–Reeves version of a conjugate gradient algorithm provided that the exact line search is employed [174]. Later, Day and Yuan referred to the algorithm as the method of shortest residuals [43] pointing out that under that name the algorithm for quadratic functions was also discussed in the monograph by Hestenes [100].

This book reflects my propensity for the method of shortest residuals. It gives the comparison of the original method stated in [174] to other conjugate gradient algorithms making the monograph the overview of standard conjugate gradient algorithms. However, the monograph goes beyond the treatment of techniques which can be regarded as the extension of the method originally proposed by Hestenes, Lanczos and Stiefel. It draws much attention to preconditioned versions of the method, and since limited memory quasi-Newton algorithms are preconditioned conjugate gradient algorithms when applied to quadratics, these variable metric techniques are also discussed.

Many optimization problems are formulated with bounds on the variables. Such problems cannot be treated directly by conjugate gradient, or variable metric algorithms. There were several proposed modifications to these methods to cope efficiently with these constraints. It appears that the way the bounds are treated by optimization techniques has a significant impact on their efficiency. In the monograph, conjugate gradient, preconditioned conjugate gradient and limited memory quasi-Newton algorithms with modifications aimed at simple constraints are presented.

I made efforts to compare numerically the discussed algorithms; however in some cases the changes to a code required adapting it to the used testing environment, were not so obvious to me. In these cases I decided not to present the code numerical behavior as it was not observed on the common set of testing problems. Nevertheless, I hope that the included numerical comparisons give an insight into the efficiency of different classes of conjugate gradient and quasi-Newton algorithms intended for large scale problems. If so, one of the aims of my work has been achieved. But, undoubtedly, this work has come into existence due to my fascination for the optimization technique which I regard to be at the core of optimization.

I am indebted to various writers on numerical methods for optimization. I have been particularly influenced by the writing of Fletcher [71], Nocedal and Wright [146] and by Hestenes [100] on the particular subject considered in the monograph. Part of the work was carried out during my PhD studies at the Institute of Automatic Control of Warsaw University of Technology; the other at the Center for Processes Systems Engineering at Imperial College in London. Both places provided a research-stimulating atmosphere without which the work reported in the monograph would not have been possible.

Several people have influenced this book in various ways. No attempt has been made to give individual credit except in special instances. I am most grateful to

Lucien Polak for several discussions we had during his visit to Imperial College, to Mehiddin Al-Baali for pointing out his joint work with Robert Fletcher on preconditioned conjugate gradient algorithms, and to my PhD student Tomasz Tarnawski for his coding of some of the algorithms presented in this book.

Warsaw, July 2008 *Radosław Pytlak*

Acknowledgements

Chapters 7, 8 and 10 are to a great extent, based on the author papers published in IMA Journal of Numerical Analysis [174], Journal of Optimization Theory and Applications [183], Pacific Journal of Optimization [181] and SIAM Journal on Optimization [175]. The permission of Oxford University Press, Springer-Verlag, Yokohama Publishers and SIAM to include this material in the monograph is acknowledged.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Conjugate Direction Methods for Quadratic Problems

1.1 Introduction

The problem of solving a set of linear equations with a symmetric positive definite matrix is equivalent to the problem of minimizing a quadratic function. Consider the problem of finding $x \in R^n$ satisfying

$$Ax = b,$$

where $A \in R^{n\times n}$, $b \in R^n$ and A is symmetric positive definite. The solution to this problem is also a solution of the optimization problem (**P**):

$$\min_{x\in R^n} \left[f(x) = \frac{1}{2}x^T Ax - b^T x \right]. \tag{1.1}$$

Consider the point $\bar{x}$ such that

$$\nabla f(\bar{x}) = g(\bar{x}) = A\bar{x} - b = 0. \tag{1.2}$$

We can show that (1.2) are the necessary optimality conditions for problem (1.1).

Lemma 1.1. *Suppose that A is a symmetric positive definite matrix. If $\bar{x}$ solves the problem (1.1) then (1.2) holds.*

Proof. Assume that $g(\bar{x}) = \bar{r} \neq 0$ and evaluate f at the point $\bar{x} - \alpha\bar{r}$, where α is some positive number:

$$\begin{aligned} f(\bar{x} - \alpha\bar{r}) &= \frac{1}{2}(\bar{x} - \alpha\bar{r})^T A(\bar{x} - \alpha\bar{r}) - b^T(\bar{x} - \alpha\bar{r}) \\ &= \frac{1}{2}\bar{x}^T A\bar{x} - \alpha\bar{x}^T A\bar{r} + \frac{1}{2}\alpha^2\bar{r}^T A\bar{r} - b^T\bar{x} + \alpha b^T\bar{r} \end{aligned}$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

$$= f(\bar{x}) - \alpha (A\bar{x} - b)^T \bar{r} + \frac{1}{2}\alpha^2 \bar{r}^T A\bar{r}$$
$$= f(\bar{x}) - \alpha \bar{r}^T \bar{r} + \frac{1}{2}\alpha^2 \bar{r}^T A\bar{r}$$

and for small values of α we obtain

$$f(\bar{x} - \alpha\bar{r}) < f(\bar{x})$$

which contradicts our assumption that $\bar{x}$ solves the problem (1.1). □

The conjugate direction method solves the problem (1.1) by referring to a sequence of simpler optimization problems.

Assume first that A is a diagonal matrix with elements on a diagonal equal to $(\lambda_1, \lambda_2, ...\lambda_n)$, where $\lambda_1 \geqslant \lambda_2 \geqslant ...\lambda_n > 0$. Then, in order to find a minimum of f we solve a sequence of one-dimensional optimizations problems $(\mathbf{P_i})$:

$$\min_{x_i \in \mathscr{R}} \left[f_i(x_i) = \frac{1}{2}\lambda_i (x_i)^2 - b_i x_i \right]$$

Each problem $(\mathbf{P_i})$ has the solution $\bar{x}_i = b_i/\lambda_i$, $i = 1, 2, ..., n$ and these solutions combined together constitute the solution of $(\mathbf{P})$.

The same result we obtain by applying the following iterative procedure.

Algorithm 1.1. (The simple conjugate direction method)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, set $k = 1$.
2. Set $p_k = e_k$, where e_k is a vector which has zero elements except the kth element which is equal to 1.
3. Find $\alpha_k > 0$ such that

$$f(x_k + \alpha_k p_k) = \min_{\alpha > 0} f(x_k + \alpha p_k).$$

4. Substitute $x_k + \alpha_k p_k$ for x_{k+1}, increase k by one.
   If $k \leqslant n$ then go to Step 2, otherwise STOP.

The minimization in Step 3 leads to the relation

$$g(x_k + \alpha_k p_k)^T p_k = 0$$

which reads as

$$\sum_{i=1}^{n} [\lambda_i [(x_k)_i + \alpha_k (p_k)_i] - b_i] (p_k)_i = 0$$

and since $p_k = e_k$ we also have

$$\alpha_k = -(x_k)_i + \frac{b_i}{\lambda_i}.$$

This implies that $(x_{k+1})_k$ assumes value b_k/λ_k. Moreover, after completing the kth step x_{k+1} minimizes f on the subspace spanned by vectors $e_1, e_2, ..., e_k$ and passing through the point x_1:

$$\mathscr{P}_k = \left\{ x \in \mathscr{R}^n : \; x = x_1 + \sum_{i=1}^{k} \gamma_i e_i, \; \gamma_i \in \mathscr{R}, \; i = 1, \ldots, k \right\}. \tag{1.3}$$

This property can be shown if we prove that gradient g_{k+1} satisfies

$$g_{k+1}^T p_i = g_{k+1}^T e_i = 0, \;\; \forall i \leqslant k. \tag{1.4}$$

To this end consider gradients evaluated at points: x_1, x_2,...,x_n. We have

$$x_{k+1} = \begin{bmatrix} b_1/\lambda_1 \\ b_2/\lambda_2 \\ \vdots \\ b_k/\lambda_k \\ (x_1)_{k+1} \\ \vdots \\ (x_1)_n \end{bmatrix}$$

which implies that

$$g_{k+1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \lambda_{k+1}(x_1)_{k+1} - b_{k+1} \\ \vdots \\ \lambda_n (x_1)_n - b_n \end{bmatrix}$$

and thus (1.4) holds.

The solution to the problem (**P**) lies in the subspace perpendicular to the space $\mathscr{P}_k$ and passing through the point x_{k+1}

$$\hat{\mathscr{P}}_{n-k} = \left\{ x \in \mathscr{R}^n : \; x = x_{k+1} + \sum_{i=k+1}^{n} \gamma_i e_i, \; \gamma_i \in \mathscr{R}, \; i = k+1, \ldots, n \right\}. \tag{1.5}$$

Because $\hat{\mathscr{P}}_{n-k}$ is the subspace of $\mathscr{R}^n$ and each iteration of Algorithm 1.1 reduces its dimension by one in at most n iterations the solution to the problem $(\mathbf{P})$ will be found.

Notice that due to the fact that A is a diagonal matrix we also have

$$p_i^T A p_j = e_i^T A e_j = 0, \ \ \forall j \neq i, \ i,j = 1,\ldots,n. \tag{1.6}$$

This property, in general nondiagonal case, will be considered as the essential feature of the methods considered in this chapter. As we will shown later it guarantees to find a solution to the problem $(\mathbf{P})$ in a finite number of operations.

When A is not a diagonal matrix the set of vectors $\{e_k\}_1^n$ does not guarantee convergence in a finite number of iterations. However, if we can transform matrix A by a nonsingular matrix S to the diagonal matrix

$$\hat{A} = S^T A S \tag{1.7}$$

then the optimization problem becomes

$$\min_{\hat{x}\in\mathscr{R}^n} \left[\hat{f}(\hat{x}) = \frac{1}{2}\hat{x}^T S^T A S \hat{x} - b^T S \hat{x} \right].$$

in the space of new variables: $x = S\hat{x}$.

In the space of variables $\hat{x}$ we can use Algorithm 1.1 to find a minimum of $\hat{f}$ in a finite number of iterations. However, this approach applied to an arbitrary positive definite matrix A has a serious drawback – we must either know the matrix S, or vectors $\{p_i\}_1^n$ such that $p_i = S^{-1}e_i$.

Notice that the relation (1.6), in the space of variables $\hat{x}$, becomes

$$p_i^T A p_j = 0, \quad \forall i \neq j, \ i,j = 1,\ldots,n. \tag{1.8}$$

1.2 Conjugate Direction Methods

Vectors $\{p_i\}_{i=1}^n$ which satisfy (1.8) are called *conjugate*. If we assume that a set of conjugate directions is given, then the optimization procedure, which finds a solution to the problem $(\mathbf{P})$ in a finite number of iterations, is as follows.

Algorithm 1.2. (The conjugate direction method – I)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, set $k = 1$.
2. Find $\alpha_k > 0$ such that

$$f(x_k + \alpha_k p_k) = \min_{\alpha>0} f(x_k + \alpha p_k). \tag{1.9}$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}, increase k by one.
   If $k \leqslant n$ then go to Step 2, otherwise STOP.

The major drawback of the above procedure is *a'priori* knowledge of conjugate directions $\{p_i\}_1^n$. Conjugate direction algorithms owe their popularity to the existence of efficient procedures for generating directions $\{p_i\}_1^n$ while they these procedures proceed. Before evaluating these procedures we present one straightforward way for constructing a set conjugate directions.

Suppose that A is a positive definite matrix which has n distinct eigenvalues λ_i, $i = 1,\ldots,n$. The corresponding v_i eigenvectors satisfy the equations

$$Av_i = \lambda_i v_i, \quad i = 1,\ldots,n.$$

We will show that vectors $\{v_i\}_1^n$ are conjugate. First, we prove that they are orthogonal and thus linearly independent. To this end consider

$$Av_i = \lambda_i v_i \tag{1.10}$$

$$Av_j = \lambda_j v_j \tag{1.11}$$

with $i \neq j$. If we multiply both sides of (1.10) by v_j^T, and both sides of (1.11) by v_i^T and subtract one of the transformed equation form the other we will obtain

$$0 = (\lambda_i - \lambda_j)\, v_j^T v_i.$$

Since $\lambda_i \neq \lambda_j$ this can happen only if $v_j^T v_i = 0$.

The normalized vectors v_i:

$$s_i = \frac{v_i}{\|v_i\|}, \quad i = 1,\ldots,n$$

satisfy

$$As_i = \lambda_i s_i,\ i = 1,\ldots,n \tag{1.12}$$

and since s_i, $i = 1,\ldots,n$ are orthogonal,

$$s_j^T A s_i = 0, \quad \forall i \neq j.$$

Therefore, vectors s_i, $i = 1,\ldots,n$ are conjugate with respect to the matrix A. According to Appendix A the described scheme for creating conjugate directions is closely related to the transformation of the matrix A to its diagonal representation – take

$$S = [s_1\ s_2\ \cdots\ s_n] \tag{1.13}$$

and, due to (1.12), (1.7) follows with the diagonal matrix

$$\hat{A} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}.$$

The subject of the chapter is to analyze the conjugate direction methods. In particular we will show how to construct effectively a set of conjugate directions without explicitly resorting to a nonsingular matrix S transforming A to its diagonal counterpart. Throughout the chapter we assume that A is a symmetric positive definite matrix.

First, we simplify the rule of determining α_k as a solution to one-dimensional minimization problem stated in (1.9). If $r_k = Ax_k - b = g_k$ is the residual of the equation $Ax = b$ evaluated at the point x_k then α_k satisfies

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}. \tag{1.14}$$

It follows from the fact that the function $\phi_k(\alpha) = f(x_k + \alpha p_k)$ attains at the point α_k its minimum so $\dot{\phi}_k(\alpha_k) = 0$ and this is equivalent to

$$\dot{\phi}_k(\alpha_k) = g(x_k + \alpha_k p_k)^T p_k = (A(x_k + \alpha_k p_k) - b)^T p_k = 0.$$

Suppose now that $\{p_i\}_1^n$ are conjugate directions with respect to matrix A. Equation (1.3) suggests that a conjugate direction method is related to the problem of minimizing quadratic function on the k-plane defined as follows. If $q_i \in \mathscr{R}^n$, $i = 1,\ldots,n-k$ are linearly independent vectors then the k-plane is the set of points satisfying

$$\mathscr{P}_k = \left\{x \in \mathscr{R}^n : \; q_i^T x = \rho_i, \quad i = 1,\ldots,n-k\right\},$$

where $\rho_i \in \mathscr{R}$, $i = 1,\ldots,n-k$.

If $x_1 \in \mathscr{P}_k$ then

$$q_i^T (x - x_1) = 0, \quad i = 1,\ldots,n-k.$$

The k-plane is also the set of points x which can be represented as

$$x = x_1 + \alpha_1 s_1 + \alpha_2 s_2 + \ldots + \alpha_k s_k,$$

for some $s_i \in \mathscr{R}^n$, $i = 1,\ldots,k$, where $\alpha_i \in \mathscr{R}$, $i = 1,\ldots,k$ and $x_1 \in \mathscr{P}_k$. Thus, any $x \in \mathscr{P}_k$ is expressed by

$$x = x_1 + \sum_{i=1}^{k} \alpha_i s_i = x_1 + Sz$$
$$z = (\alpha_1, \alpha_2, \ldots, \alpha_k)^T.$$

Therefore, the problem of minimizing f on the k-plane can be stated as the unconstrained optimization problem:

$$\min_{z \in \mathscr{R}^k} \left[h(z) = f(x_1 + Sz)\right]. \tag{1.15}$$

If we assume that the problem (1.1) has the unique solution then the solution to problem (1.15) is also unique. Furthermore, its solution $\hat{z}$ satisfies

$$\nabla h(\hat{z}) = 0$$

which means that

$$S^T g(x_1 + S\hat{z}) = 0. \tag{1.16}$$

We can conclude our consideration by the following theorem.

Theorem 1.1. *If A is a symmetric positive definite matrix then point $\hat{x}$ is the point minimizing f on $\mathscr{P}_k$ if and only if*

$$s_i^T g(\hat{x}) = 0, \quad i = 1, 2, \ldots, k.$$

Proof. The thesis is a direct consequence of (1.16) if we notice that $\hat{x} = x_1 + S^T\hat{z}$. □

The significance of the above theorem is illustrated in the case of 1-plane. Consider a line defined by

$$x = x_1 + \alpha p_1, \ \ \alpha \in \mathscr{R}. \tag{1.17}$$

The point x_2 minimizes f on (1.17) if

$$p_1^T (Ax_2 - b) = 0. \tag{1.18}$$

Notice, that if $\bar{x}$ solves the problem (1.1) then we have $A\bar{x} - b = 0$ which means that it also satisfies (1.18). Therefore, we have proved the theorem.

Theorem 1.2. *Points which minimize f on lines with the direction p_1 lie in the plane*

$$\hat{\mathscr{P}}_{n-1} = \left\{ x \in \mathscr{R}^n : \ p_1^T (Ax - b) = 0 \right\}$$

which passes through the minimum point of f.

Consider two different points of the plane $\hat{\mathscr{P}}_{n-1}$: x, $\tilde{x}$. If $y = x - \tilde{x}$ then

$$\begin{aligned} p_1^T (Ax - b) &= 0 \\ p_1^T (A\tilde{x} - b) &= 0, \end{aligned} \tag{1.19}$$

thus

$$p_1^T A y = 0$$

and that means that the $(n-1)$-plane $\hat{\mathscr{P}}_{n-1}$ is conjugate to p_1. Since $\bar{x} \in \hat{\mathscr{P}}_{n-1}$ it is natural to seek for a new approximation to $\bar{x}$ in $\hat{\mathscr{P}}_{n-1}$, for example along a line which is conjugate to p_1.

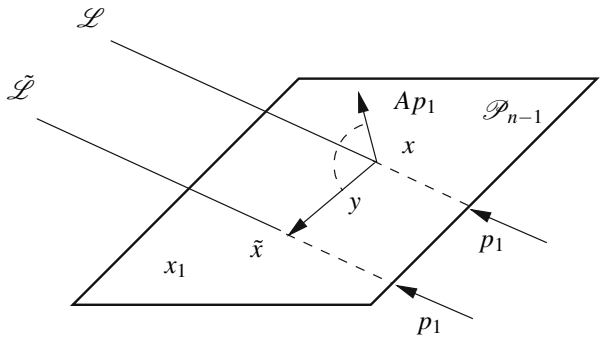

Fig. 1.1 Properties of the $(n-1)$-plane: $\hat{\mathscr{P}}_{n-1}$

Figure 1.1 (based on Fig. 3.1 given in Chap. II of [100]) illustrates the $(n-1)$-plane conjugate to p_1. When x and $\tilde{x}$ belong to $\hat{\mathscr{P}}_{n-1}$ then $y \in \hat{\mathscr{P}}_{n-1}$. If $\mathscr{L}$ and $\tilde{\mathscr{L}}$ are lines passing through the points x and $\tilde{x}$, respectively, and are determined by the vector p_1 then y is orthogonal to Ap_1. In fact any vector in $\hat{\mathscr{P}}_{n-1}$ has this property. Observe that Ap_1 is normal of $\hat{\mathscr{P}}_{n-1}$ and that minimum of f on any line parallel to $\mathscr{L}$ must be attained at its intersection with $\hat{\mathscr{P}}_{n-1}$ (since (1.19) holds).

Assume now that p_2 is conjugate to p_1. We seek a minimum of f on the line

$$\mathscr{L}_2 = \{x \in \mathscr{R}^n : \; x = x_2 + \alpha p_2, \; \alpha \in \mathscr{R}\}. \tag{1.20}$$

The point x_3 which minimizes f on the line $\mathscr{L}_2$ must satisfy

$$p_2^T (Ax_3 - b) = 0.$$

However,

$$x_3 = x_1 + \alpha_1 p_1 + \alpha_2 p_2,$$

and from (1.14) α_1 and α_2 have to fulfill

$$\alpha_1 = -\frac{r_1^T p_1}{p_1^T A p_1}$$
$$\alpha_2 = -\frac{r_2^T p_2}{p_2^T A p_2}.$$

In addition, taking into account the relation

$$\begin{aligned} r_2 &= Ax_2 - b = A(x_1 + \alpha_1 p_1) - b \\ &= r_1 + \alpha_1 A p_1 \end{aligned}$$

and the fact that vectors p_1 and p_2 are conjugate we have

$$p_2^T r_1 = p_2^T (r_2 - \alpha_1 A p_1) = p_2^T r_2. \tag{1.21}$$

It remains to show that these conditions imply that point x_3 is a minimum point of f on the 2-plane:

$$\mathscr{P}_2 = \{x \in \mathscr{R}^n : \; x = x_1 + \gamma_1 p_1 + \gamma_2 p_2, \; \gamma_1, \gamma_2 \in \mathscr{R}\}.$$

A minimum point $\tilde{x}$ must satisfy

$$g(\tilde{x})^T p_1 = 0$$
$$g(\tilde{x})^T p_2 = 0$$

which can be expressed as

$$(Ax_1 + \tilde{\gamma}_1 A p_1 + \tilde{\gamma}_2 A p_2 - b)^T \, p_1 = 0$$
$$(Ax_1 + \tilde{\gamma}_1 A p_1 + \tilde{\gamma}_2 A p_2 - b)^T \, p_2 = 0.$$

Since p_1, p_2 are conjugate:

$$\tilde{\gamma}_1 = -\frac{r_1^T p_1}{p_1^T A p_1}$$
$$\tilde{\gamma}_2 = -\frac{r_1^T p_2}{p_2^T A p_2}.$$

Because $\tilde{\gamma}_1 = \alpha_1$ (due to (1.21)) and $\tilde{\gamma}_2 = \alpha_2$ we have proved that x_3 is the minimum point of f on the 2-plane $\mathscr{P}_2$. Therefore, $g(x_3)$ is perpendicular to vectors p_1 and p_2:

$$p_1^T (Ax_3 - b) = 0$$
$$p_2^T (Ax_3 - b) = 0,$$

thus x_3 belongs to the $(n-2)$-plane:

$$\hat{\mathscr{P}}_{n-2} = \left\{x \in \mathscr{R}^n : \; p_i^T (Ax - b) = 0, \; i = 1,2\right\}$$

which also includes $\bar{x}$.

We can continue the procedure described above, i.e. a new point x_4 is found as a point minimizing f on the line $\mathscr{L}_3 = \{x \in \mathscr{R}^n : \; x = x_3 + \alpha p_3, \; \alpha \in \mathscr{R}\}$, where p_3 is a vector conjugate to p_1 and p_2. That line lies in $\hat{\mathscr{P}}_{n-2}$. Each consecutive line $\mathscr{L}_k$ and the minimization of f on $\mathscr{L}_k$ reduces the dimension of a subspace in which $\bar{x}$ lies. Eventually, $\bar{x}$ is the minimum of f on the line $\mathscr{L}_m = \{x \in \mathscr{R}^n : \; x = x_m + \alpha p_m, \; \alpha \in \mathscr{R}\}$ where $m \leqslant n$.

Algorithm 1.2 can be restated as follows.

Algorithm 1.3. (The conjugate direction method – II)

1. Choose an arbitrary point $x_1 \in \mathscr{R}^n$ and a direction $p_1 \neq 0$. Set $k = 1$
2. Find a minimum point of f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \; x = x_k + \alpha p_k, \; \alpha \in \mathscr{R}\}$. Let this point be $x_{k+1} = x_k + \alpha_k p_k$.

3. Find a direction p_{k+1} which is conjugate to directions $p_1, p_2, \ldots, p_k$.
4. Increase k by one. If $k \leqslant n$ go to Step 2.

The conjugate direction method has several properties which are listed in the theorems stated below. The first one establishes conditions which must be met by a minimum point on the k-plane. This result is then used to show essential properties of the points $x_2, \ldots, x_k, \ldots$ generated by the method.

Theorem 1.3. *If $\mathscr{P}_k$ is a k-plane through a point x_1:*

$$\mathscr{P}_k = \left\{ x \in \mathscr{R}^n : \; x = x_1 + \sum_{i=1}^{k} \gamma_i p_i, \; \gamma_i \in \mathscr{R}, \; i = 1, \ldots, k \right\}$$

and vectors $\{p_i\}_1^k$ are conjugate, then the minimum point x_{k+1} of f on $\mathscr{P}_k$ satisfies

$$x_{k+1} = x_1 + \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k,$$

where

$$\alpha_i = -\frac{c_i}{d_i}, \;\; c_i = r_1^T p_i, \;\; d_i = p_i^T A p_i, \;\; i = 1, \ldots, k \tag{1.22}$$

and $r_1 = Ax_1 - b = g_1$.

Proof. Consider the residual of f at the point x_{k+1}: $r_{k+1} = g_{k+1} = Ax_{k+1} - b$. It must be perpendicular to the k-plane $\mathscr{P}_k$, thus

$$p_i^T r_{k+1} = 0, \;\; i = 1, \ldots, k. \tag{1.23}$$

Taking into account that vectors p_i, $i = 1, \ldots, k$ are conjugate we come to the relations

$$\alpha_i p_i^T A p_i + p_i^T A x_1 - p_i^T b = \alpha_i p_i^T A p_i + p_i^T r_1 = 0, \;\; i = 1, \ldots, k$$

from which the thesis follows. □

As we have already observed x_{k+1} belongs to the $(n-k)$-plane $\hat{\mathscr{P}}_k$ passing through the point x_k.

Corollary 1.1. *The minimum point x_{k+1} of f on the k-plane $\mathscr{P}_k$ belongs also to the $(n-k)$-plane $\hat{\mathscr{P}}_k$:*

$$\hat{\mathscr{P}}_k = \left\{ x \in \mathscr{R}^n : \; p_i^T (Ax - b) = 0, \;\; i = 1, \ldots, k \right\}.$$

Proof. The thesis follows from relation (1.23). □

Finally we refer to the essential properties of conjugate directions. They will be useful in deriving constructive ways of generating conjugate directions within the method described above.

Theorem 1.4. *Let $p_1, p_2, \ldots, p_k$ be conjugate vectors and x_1 be an arbitrary point. If $x_i,\ i = 2, \ldots, k+1$ are defined recursively by the formula*

$$x_{i+1} = x_i + \alpha_i p_i$$

where x_{i+1} is the minimum point of f on the line $\mathscr{L}_i = \{x \in \mathscr{R}^n :\ x + x_i + \alpha p_i, \alpha \in \mathscr{R}\}$, then

$$\alpha_i = -\frac{c_i}{d_i},\ \ c_i = p_i^T r_i,\ \ d_i = p_i^T A p_i \tag{1.24}$$

and x_{k+1} minimizes f on the k-plane $\mathscr{P}_k$. Moreover, we have relations

$$p_i^T r_j = c_i,\ \ 1 \leqslant j \leqslant i, \tag{1.25}$$

$$p_i^T r_j = 0,\ \ \ i < j \leqslant k+1. \tag{1.26}$$

Proof. x_{i+1} minimizes f on the line $\mathscr{L}_i$ thus

$$r_{i+1}^T p_i = 0$$

and due to the formula for r_{i+1}

$$p_i^T (r_i + \alpha_i A p_i) = 0,$$

therefore (1.24) follows. Since vectors $p_1, p_2, \ldots, p_k$ are conjugate

$$p_i^T r_{j+1} = p_i^T (r_j + \alpha_j A p_j) = p_i^T r_j,$$

if $j \neq i$ and, due to $r_j = r_1 + \sum_{l=1}^{j-1} \alpha_l A p_l$,

$$p_i^T r_j = p_i^T r_1 = p_i^T r_2 = \ldots = p_i^T r_i = c_i$$

and

$$p_i^T r_{i+1} = p_i^T r_{i+2} = \ldots = p_i^T r_{k+1} = 0$$

which states the thesis. □

So far we have assumed that vectors $\{p_i\}_1^m$ are conjugate. Now suppose that vectors $\{p_i\}_1^m$ are linearly independent and consider again the procedure described above. We will show that, under certain conditions, these vectors are also conjugate.

Theorem 1.5. *Let vectors $\{p_i\}_1^m$ be linearly independent, x_1 be an arbitrary point and the sequence $\{x_k\}_1^{m+1}$ is defined as follows*

$$x_{k+1} = x_k + \alpha_k p_k = x_1 + \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k$$

with $\alpha_k \neq 0$, $k = 1,2,\ldots,m$. Suppose also that for $k = 1,\ldots,m$ x_{k+1} is the minimum point of f on the k-plane

$$\mathscr{P}_k = \left\{ x \in \mathscr{R}^n : x = x_1 + \sum_{i=1}^{k} \gamma_i p_i,\ \gamma_i \in \mathscr{R},\ i = 1,\ldots,k \right\}.$$

Then vectors $\{p_k\}_1^m$ are conjugate.

Proof. Since x_{k+1} minimizes f on the plane $\mathscr{P}_k$

$$r_{k+1}^T p_i = 0,\ \ i = 1,2,\ldots,k. \tag{1.27}$$

Moreover, $p_i^T r_k = 0$, if $i < k$, which together with (1.27) imply that

$$p_i^T r_{k+1} = p_i^T (r_k + \alpha_k A p_k) = \alpha_k p_i^T A p_k = 0$$

for $i < k$. Since $\alpha_k \neq 0$ we also have $p_i^T A p_k = 0$. □

In fact we can remove the assumption that vectors $\{p_k\}_1^m$ are linearly independent if the following conditions are met

$$p_j^T r_k = 0,\ \ 1 \leqslant j < k \leqslant m. \tag{1.28}$$

Theorem 1.6. *If vectors p_k, $k = 1,2,\ldots,m$ are nonzero, points x_k, $k = 2,3,\ldots,m+1$ are defined recursively by the formulae (starting with x_1 as an arbitrary point):*

$$x_{k+1} = x_k + \alpha_k p_k,\ \alpha_k = -\frac{c_k}{d_k},$$
$$c_k = r_k^T p_k \neq 0,\ \ d_k = p_k^T A p_k,\ \ k = 1,2,\ldots,m$$

and relations (1.28) are satisfied, then vectors $\{p_k\}_1^m$ are conjugate.

Proof. Because (1.28) holds point x_{k+1} is the minimum point of f on the plane $\mathscr{P}_k$

$$\mathscr{P}_k = \left\{ x \in \mathscr{R}^n :\ x = x_1 + \sum_{i=1}^{k} \gamma_i p_i,\ \gamma_i \in \mathscr{R},\ i = 1,\ldots,k \right\}.$$

However, due to (1.28), we also have

$$p_j^T r_{k+1} = p_j^T (r_k + \alpha_k A p_k) = \alpha_k p_j^T A p_k,$$

and since $\alpha_j \neq 0$ we have $p_j^T A p_k = 0$, $1 \leqslant j < k \leqslant m$. Thus vectors $p_1,\ldots,p_m$ are conjugate. □

1.3 Conjugate Gradient Error Algorithms

A conjugate direction method requires only that at the kth iteration the direction p_k is conjugate to the previous directions p_1, p_2,...,p_{k-1}. On one hand we are free in choosing these directions, on the other hand the procedure of calculating p_k must be efficient. Previously we have described the conjugate direction method based on the eigenvectors of the matrix A. However, it is expensive to evaluate eigenvectors of a matrix. Another conjugate direction method could be derived from the process of the Gram–Schmidt orthogonalization, yet it requires storing entire direction set and that is very inconvenient [100].

An ingenious approach was proposed by Hestenes and Stiefel in 1952 [101]. They discovered the conjugate direction method which calculates the direction p_k on the basis of only the previous direction and the current gradient. They set

$$p_k = -r_k + \beta_k p_{k-1} = -g_k + \beta_k p_{k-1}$$

where β_k is some coefficient dependent on data from the current and the previous iterations. Since p_k and p_{k-1} are assumed to be conjugate

$$p_k^T A p_{k-1} = -r_k^T A p_{k-1} + \beta_k p_{k-1}^T A p_{k-1} = 0$$

so β_k can be evaluated according to the formula

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}. \tag{1.29}$$

They assumed that $p_1 = -g_1$ (seems to be very natural choice of p_1) and their algorithm, which was called the conjugate gradient algorithm, can be stated as follows.

Algorithm 1.4. (The conjugate gradient algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

   $$p_1 = -r_1 = -g_1$$

   and $k = 1$.
2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : = x_k + \alpha p_k,\ \alpha \in \mathscr{R}\}$:

   $$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}.$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate β_{k+1} according to

   $$\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k} \tag{1.30}$$

and p_{k+1} according to

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k,$$

where

$$r_{k+1} = Ax_{k+1} - b = g_{k+1}.$$

4. Increase k by one and go to Step 2.

Before we show that the conjugate gradient algorithm is a conjugate direction method we provide alternative formulae for β_k. Observe first that c_k (cf. (1.22) and (1.24)) is equal to $-\|r_k\|^2$:

$$c_k = r_k^T p_k = r_k^T \left(-r_k + \beta_k p_{k-1}\right) = -\|r_k\|^2. \tag{1.31}$$

Using relation (1.29) and

$$\|r_{k+1}\|^2 = \left(r_k + \alpha_k A p_k\right)^T r_{k+1} = r_{k+1}^T r_k - c_k \beta_{k+1}$$

we find the second formula for β_{k+1}

$$\beta_{k+1} = \frac{\|r_{k+1}\|^2 - r_{k+1}^T r_k}{\|r_k\|^2} = \frac{r_{k+1}^T \left(r_{k+1} - r_k\right)}{\|r_k\|^2}.$$

Finally, since, as we show in Theorem 1.7, $r_{k+1}^T r_k = 0$, we have yet another formula for β_{k+1}

$$\beta_{k+1} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}.$$

We present convergence properties of the conjugate gradient algorithm by showing that directions $\{p_i\}_1^n$ are conjugate. Moreover, we establish that residuals $\{r_i\}_1^n$ are mutually orthogonal and, if $\mathscr{K}(r_1;i)$ is the Krylov subspace of degree i for r_1 defined as follows

$$\mathscr{K}(r_1;i) = \operatorname{span}\left\{r_1, Ar_1, \ldots, A^i r_1\right\}, \tag{1.32}$$

then the direction p_i and the residual r_i are contained in $\mathscr{K}(r_1;i-1)$.

Theorem 1.7. *Suppose that the point x_k, generated by the conjugate gradient algorithm, is not the minimum point of f. Then the following hold*

$$\operatorname{span}\{r_1, r_2, \ldots, r_{k+1}\} = \mathscr{K}(r_1;k) \tag{1.33}$$

$$\operatorname{span}\{p_1, p_2, \ldots, p_{k+1}\} = \mathscr{K}(r_1;k) \tag{1.34}$$

$$p_k^T A p_i = 0, \quad i = 1,2,\ldots,k-1 \tag{1.35}$$

$$r_k^T r_i = 0, \quad i = 1,2,\ldots,k-1. \tag{1.36}$$

Proof. Suppose that (1.33)–(1.34) hold for k – they are obviously satisfied for $k=1$ and $k=2$. Now we show that they also are true for $k+1$.

By induction hypothesis

$$r_k \in \mathscr{K}(r_1;k-1) \tag{1.37}$$
$$p_k \in \mathscr{K}(r_1;k-1). \tag{1.38}$$

Relation (1.38) implies that

$$Ap_k \in \operatorname{span}\left\{Ar_1, A^2r_1, \ldots, Ar_1^k\right\}$$

from which, due to $r_{k+1} = r_k + \alpha_k Ap_k$ and (1.37), follows

$$r_{k+1} \in \operatorname{span}\left\{r_1, Ar_1, \ldots, A^k r_1\right\}.$$

In order to prove the reverse inclusion observe that

$$A^k r_1 = A\left(A^{k-1}r_1\right) \in \operatorname{span}\{Ap_1, Ap_2, \ldots, Ap_k\}. \tag{1.39}$$

However, we also have

$$Ap_i = \frac{(r_{i+1}-r_i)}{\alpha_i}, \;\; i=1,2,\ldots,k,$$

which combined with (1.39) and the induction hypothesis result in

$$\operatorname{span}\left\{r_1, Ar_1, \ldots, A^k r_1\right\} \subset \{r_1, r_2, \ldots, r_{k+1}\}.$$

To prove (1.34) we take into account (1.33), the induction hypotheses and the formula $p_{k+1} = -r_{k+1} + \beta_{k+1}p_k$ to show that

$$\begin{aligned}
\operatorname{span}\{p_1, p_2, \ldots, p_{k+1}\} &= \operatorname{span}\{p_1, p_2, \ldots, p_k, r_{k+1}\} \\
&= \operatorname{span}\left\{r_1, Ar_1, \ldots, A^{k-1}r_1, r_{k+1}\right\} \\
&= \operatorname{span}\{r_1, r_2, \ldots, r_{k+1}\} \\
&= \operatorname{span}\left\{r_1, Ar_1, \ldots, A^k r_1\right\}.
\end{aligned}$$

In order to show (1.35) we refer to Theorem 1.6 which states that if the conditions

$$r_k^T p_i = 0, \;\; i=1,2,\ldots,k-1 \tag{1.40}$$

are fulfilled then the directions $p_1, \ldots, p_k$ are conjugate.

To this end assume that (1.40) holds for some $i < k-1$ – we will show that (1.40) is valid also for $i+1$.

First of all, since (1.40) is satisfied, from Theorem 1.6, we conclude that p_1, $p_2, \ldots, p_i$ are conjugate:

$$p_j^T A p_i = 0, \quad j < i,\ j,i \leqslant k.$$

We have to show that $p_1, \ldots, p_{i+1}$ are also conjugate. Thus, we will prove that

$$p_{i+1}^T A p_j = -r_{i+1}^T A p_j + \beta_{i+1} p_i^T A p_j, \ \ j \leqslant i$$

is equal to zero. To show that observe that

$$\begin{aligned} A p_i \in A \text{ span}\left\{r_1, Ar_1, \ldots, A^{i-1} r_1\right\} &= \text{span}\left\{Ar_1, A^2 r_1, \ldots, A^i r_1\right\} \\ &\subset \text{span}\left\{p_1, p_2, \ldots, p_{i+1}\right\}. \end{aligned} \tag{1.41}$$

Furthermore, $p_1, p_2, \ldots, p_i$ are conjugate thus until the ith iteration the conjugate gradient algorithm performs the iterations of the conjugate direction algorithm resulting in

$$r_{i+1}^T p_j = 0, \ \ j \leqslant i. \tag{1.42}$$

Combining (1.41) and (1.42) gives us

$$p_{i+1}^T A p_j = 0, \ \ j \leqslant i.$$

Finally, we prove (1.36) directly. Since the directions $p_1, \ldots, p_{i+1}$ are conjugate, from Theorem 1.6,

$$r_{i+1}^T p_j = 0, \ \ j \leqslant i, \tag{1.43}$$

and because

$$p_i = -r_i + \beta_i p_{i-1},$$

we have

$$r_i \in \text{span}\left\{p_i, p_{i-1}\right\}.$$

This together with (1.43) prove (1.36). □

The theorem shows that gradients evaluated by the conjugate gradient algorithm are mutually orthogonal. Therefore, the question arises why the method was called the conjugate gradient algorithm? One explanation is that the method starts with $p_1 = -g_1$. In addition Hestenes [100] proves that directions generated by the conjugate gradient method are directions of steepest descent in the subspaces

$$\hat{\mathscr{P}}_{n-k} = \left\{x \in \mathscr{R}^n :\ p_j^T A\left(x - x_{k+1}\right),\ j = 1, \ldots, k\right\}. \tag{1.44}$$

Suppose that p is obtained by the orthogonal projection of r_{k+1} on the subspace $\hat{\mathscr{P}}_{n-k}$. Then

$$p = -r_{k+1} + \theta_1 A p_1 + \theta_2 A p_2 + \ldots + \theta_k A p_k \tag{1.45}$$

for some $\theta_j \in \mathscr{R}$, $j = 1,\ldots,k$, since Ap_i, $i = 1,2,\ldots,k$ are normals of $\hat{\mathscr{P}}_{n-k}$.

By taking into account (1.33)–(1.34) we can rewrite (1.45) as

$$p = -r_{k+1} + \xi_1 p_1 + \xi_2 p_2 + \ldots + \xi_k p_k + \xi_{k+1} p_{k+1}.$$

From the orthogonality of p to Ap_i, $i = 1,2,\ldots,k$ we have

$$p^T A p_i = \xi_i p_i^T A p_i + p_i^T A r_{k+1} = 0, \;\; i = 1,2,\ldots,k. \tag{1.46}$$

Furthermore, $r_{i+1} = r_i + \alpha_i A p_i$ and by (1.35) the following holds $r_{k+1}^T r_i = 0$, $i \leqslant k$, thus

$$\alpha_i p_i^T A r_{k+1} = (r_{i+1} - r_i)^T r_{k+1} = 0, \; i < k$$

and by (1.46) we conclude that $\xi_i = 0$ for $i = 1,2,\ldots,k-1$. On the other hand, by (1.30), $\xi_k = \beta_{k+1}$, hence

$$p = -r_{k+1} + \beta_{k+1} p_k + \xi_{k+1} p_{k+1} = (1 + \xi_{k+1})\, p_{k+1} = \gamma p_{k+1}.$$

It remains to show that $\gamma > 0$. Indeed, directional derivative of f at the point x_{k+1} and along the direction p is negative, since at each iteration we decrease the value of f, and by (1.31) is equal to

$$p^T r_{k+1} = \gamma p_{k+1}^T r_{k+1} = -\gamma \|r_{k+1}\|^2 < 0,$$

thus $\gamma > 0$.

The conjugate gradient algorithm is one representative of the whole family of algorithms which differ by scaling parameters which determine lengths of directions p_k. General algorithm can be stated as follows.

Algorithm 1.5. (The general conjugate gradient algorithm)

Parameters: sequence of positive scaling parameters $\{\sigma_k\}_1^n$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$p_1 = -\sigma_1 r_1 = -\sigma_1 g_1$$

and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \; x = x_k + \alpha p_k, \; \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}.$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate β_{k+1} according to

$$\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

and p_{k+1} according to

$$p_{k+1} = \sigma_{k+1}\left(-r_{k+1} + \beta_{k+1} p_k\right).$$

4. Increase k by one and go to Step 2.

We can choose different scaling parameters $\{\sigma_k\}_1^n$ to obtain different directions p_k but only with respect to their lengths. In fact each algorithm from the family parameterized by a sequence $\{\sigma_k\}_1^n$ generates the same sequence $\{x_k\}_1^n$. In order to verify this notice first that

$$x_{k+1}^1 = x_k + \alpha_k^1 p_k^1 = x_k + \frac{\alpha_k^1}{\sigma_k}\sigma_k p_k^1, \tag{1.47}$$

where x_{k+1}^1, α_k^1 and p_k^1 corresponds to the case $\sigma_k \equiv 1$. However,

$$\begin{aligned}\alpha_k^1 &= -\frac{\left(p_k^1\right)^T r_k}{\left(p_k^1\right)^T A\left(p_k^1\right)} \\ &= -\sigma_k \frac{p_k^T r_k}{p_k^T A p_k} = \sigma_k \alpha_k\end{aligned}$$

and from (1.47)

$$x_{k+1}^1 = x_k + \alpha_k p_k = x_{k+1}.$$

Points generated by each algorithm from the family of methods described by $\{\sigma_k\}_1^n$ are identical in the case of a convex quadratic function f and when minimization along each line $\mathscr{L}_k$ is exact. It is, however, not true in any other case. Therefore, the family deserves further attention. As it is shown in next chapters the proper choice of the sequence $\{\sigma_k\}_1^n$ can result in algorithms of strong convergence properties in a differentiable but nonconvex case of f.

Since for each k

$$p_k = \sigma_k p_k^1,$$

the directions p_1, p_2,...,p_n are conjugate. From our previous considerations the following relations, valid for p_1^1, p_2^1,...,p_n^1, can be extended to the general case:

$$\begin{gathered} r_j^T r_k = 0, \quad \forall j \neq k \\ p_k^T r_j = p_k^T r_k, \quad j \leqslant k, \quad p_k^T r_j = 0, \quad j > k. \end{gathered}$$

These properties lead to the characterization of p_k:

$$p_k = c_k \left(\frac{r_1}{\rho_1} + \frac{r_2}{\rho_2} + \ldots + \frac{r_k}{\rho_k} \right), \tag{1.48}$$

where $\rho_j = \|r_j\|^2$. It is easy to check that (1.48) is valid for $k = 1$. Suppose now that (1.48) is true for some $k > 1$. Since

$$c_{k+1} = -\|r_{k+1}\|^2 = -\rho_{k+1}, \quad \beta_{k+1} = \frac{c_{k+1}}{c_k}$$

we also have

$$\begin{aligned} p_{k+1} &= -r_{k+1} + \beta_{k+1} p_k \\ &= \frac{c_{k+1}}{\rho_{k+1}} r_{k+1} + \frac{c_{k+1}}{c_k} p_k \\ &= c_{k+1} \left(\frac{p_k}{c_k} + \frac{r_{k+1}}{\rho_{k+1}} \right) \end{aligned}$$

and from the induction hypothesis

$$p_{k+1} = c_{k+1} \left(\frac{r_1}{\rho_1} + \frac{r_2}{\rho_2} + \ldots + \frac{r_{k+1}}{\rho_{k+1}} \right)$$

which proves (1.48).

Algorithm 1.5 is a conjugate direction algorithm. Having (1.48), and following Hestenes [100] we can prove that, under the mutual orthogonality assumption of residuals, it is in fact a conjugate gradient algorithm.

Theorem 1.8. *Suppose that directions p_1, p_2,...,p_n are conjugate,*

$$x_{k+1} = x_k + \alpha_k p_k,$$

and

$$\alpha_k = -\frac{c_k}{d_k}, \quad c_k = p_k^T r_k, \quad d_k = p_k^T A p_k.$$

Then, if r_1, r_2,...,r_n are mutually orthogonal, there exist positive numbers σ_k, $k = 1, \ldots, n$ such that

$$p_1 = -\sigma_1 r_1, \quad p_{k+1} = \sigma_{k+1} \left(-r_{k+1} + \beta_{k+1} p_k \right) \tag{1.49}$$

and Algorithm 1.5 *is a conjugate gradient algorithm.*

Proof. Since the algorithm is a conjugate direction method its directions must lie in the space spanned by residuals:

$$p_k = \zeta_1 r_1 + \zeta_2 r_2 + \ldots + \zeta_n p_n.$$

From the orthogonality assumption it follows that

$$\zeta_j = \frac{p_k^T r_j}{\rho_j}, \quad \rho_j = \|r_j\|^2.$$

Furthermore, from the conjugacy assumption we come to the relations

$$r_j^T p_k = r_k^T p_k = c_k, \quad j \leqslant k, \quad r_j^T p_k = 0, \quad j > k$$

which imply

$$p_k = c_k \left(\frac{r_1}{\rho_1} + \frac{r_2}{\rho_2} + \ldots + \frac{r_k}{\rho_k} \right). \tag{1.50}$$

If we define

$$\sigma_k = -\frac{c_k}{\rho_k}, \quad \beta_{k+1} = \frac{c_{k+1}}{c_k}$$

we come to (1.49). □

A conjugate direction algorithm considered so far has the property that x_{k+1} solves the problem

$$\min_{x \in \mathscr{R}^n} \left[\mathscr{E}(x) = \|x - \bar{x}\|_A^2 = (x - \bar{x})^T A (x - \bar{x}) \right] \tag{1.51}$$

on the subspace

$$\mathscr{P}_k = \left\{ x \in \mathscr{R}^n : x = x_1 + \sum_{i=1}^{k} \gamma_i p_i, \ \gamma_i \in \mathscr{R}, \ i = 1, \ldots, k \right\}. \tag{1.52}$$

We remind that $\bar{x}$ is the minimum point of f. This property is the immediate consequence of the fact that x_{k+1} minimizes f on the subspace $\mathscr{P}_k$ given by (1.52). The necessary optimality conditions for that are

$$r_{k+1}^T p_j = 0, \quad j = 1, 2, \ldots, k$$

which can also be stated as

$$(A x_{k+1} - b)^T p_j = 0, \quad j = 1, 2, \ldots, k. \tag{1.53}$$

Since $A\bar{x} = b$ conditions (1.53) are equivalent to

$$(A (x_{k+1} - \bar{x}))^T p_j = 0, \quad j = 1, 2, \ldots, k$$

which, under the assumption that A is positive definite, are sufficient optimality conditions for the problem of minimizing the function $\mathscr{E}$ on the subspace $\mathscr{P}_k$.

Conjugate direction methods, which at every iteration minimize the function $\mathscr{E}$, are called *conjugate gradient error algorithms*. The kth iteration of these algorithms can be represented by the following problem

$$\text{find } \tilde{x} \in x_1 + \mathscr{K}(r_1;k) \text{ such that } (A\tilde{x} - b = g(\tilde{x})) \perp \mathscr{K}(r_1;k), \tag{1.54}$$

where $\mathscr{K}(r_1;k)$ is the Krylov subspace defined in (1.32). Here, $a \perp b$ means that vector $a \in \mathscr{R}^n$ is orthogonal to vector $b \in \mathscr{R}^n$.

1.4 Conjugate Gradient Residual Algorithms

If we change the condition in problem (1.54) by requiring that $g(\tilde{x}) \perp A\mathscr{K}(r_1;k)$ we will define a new problem which uses the so-called *oblique projection* (cf. [189]):

$$\text{find } \tilde{x} \in x_1 + \mathscr{K}(r_1;k) \text{ such that } A\tilde{x} - b \perp A\mathscr{K}(r_1;k). \tag{1.55}$$

The oblique projection is illustrated in Figs. 1.2–1.3. In Fig. 1.2 the point $\mathscr{Q}_{\mathscr{K}}^{\mathscr{L}}[x]$ belonging to the subspace $\mathscr{K}$ is constructed by the condition that $x - \mathscr{Q}_{\mathscr{K}}^{\mathscr{L}}[x] \perp \mathscr{L}$. When $\mathscr{L} = \mathscr{K}$ then the oblique projection gives the point $\mathscr{P}_{\mathscr{K}}[x]$ of the orthogonal projection. Both $\mathscr{Q}_{\mathscr{K}}^{\mathscr{L}}[x]$ and $\mathscr{P}_{\mathscr{K}}[x]$ are shown in Fig. 1.3.

The problem (1.55) gives rise to the following optimization problem

$$\min_{x \in x_1 + \mathscr{K}(r_1;k)} [\mathscr{R}(x) = \|Ax - b\|_2]. \tag{1.56}$$

The necessary and sufficient conditions for $\tilde{x}$ to be a solution of problem (1.56) are

$$(A\tilde{x} - b)^T w = 0, \ \ \forall w \in A\mathscr{K}(r_1;k)$$

which rephrases the problem (1.55).

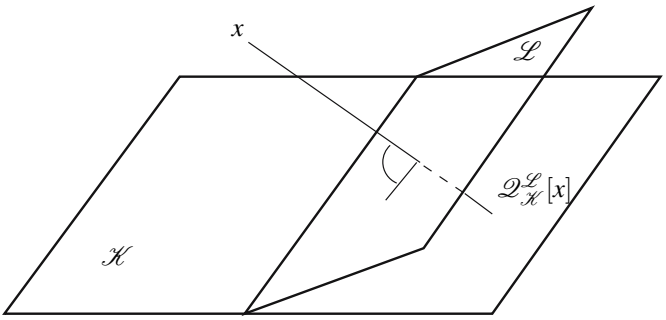

Fig. 1.2 The oblique projection

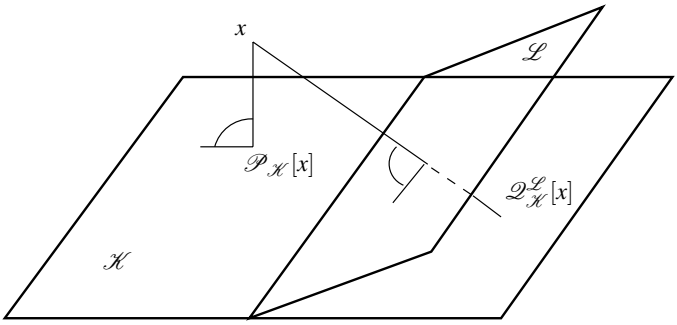

Fig. 1.3 The oblique and orthogonal projections

The theorem given below and proved by Hestenes [100] gives theoretical basis for a conjugate gradient algorithm which is based on problem (1.56) instead of problem (1.51).

Theorem 1.9. *The direction p_{k+1} generated by a conjugate gradient algorithm is the direction of the steepest descent evaluated at the point $\hat{x}_{k+1}$ defined by*

$$\hat{x}_{k+1} = \hat{\rho}_{k+1}\left(\frac{x_1}{\rho_1} + \frac{x_2}{\rho_2} + \ldots + \frac{x_k}{\rho_k} + \frac{x_{k+1}}{\rho_{k+1}}\right),$$

where

$$\frac{1}{\hat{\rho}_{k+1}} = \frac{1}{\rho_1} + \frac{1}{\rho_2} + \ldots + \frac{1}{\rho_k} + \frac{1}{\rho_{k+1}}.$$

Furthermore

$$\min_{x \in x_1 + \mathscr{K}(r_1;k)} \mathscr{R}(x) = \mathscr{R}(\hat{x}_{k+1})$$

and

$$p_{k+1} = -\theta_{k+1} g(\hat{x}_{k+1}) = -\theta_{k+1}\hat{r}_{k+1}, \quad \theta_{k+1} = \left|\frac{c_{k+1}}{\hat{\rho}_{k+1}}\right|.$$

Proof. $\hat{x}_{k+1}$ is a linear combination of $x_1, x_2, ..., x_{k+1}$ thus lies in the k-plane $\mathscr{P}_k$ spanned by vectors $p_1, p_2, ..., p_k$. The residual evaluated at this point is given by

$$\begin{aligned}\hat{r}_{k+1} &= A\hat{x}_{k+1} - b = \hat{\rho}_{k+1}\sum_{i=1}^{k+1}\frac{Ax_i - b}{\rho_i} = \hat{\rho}_{k+1}\sum_{i=1}^{k+1}\frac{r_i}{\rho_i} \\ &= \frac{\hat{\rho}_{k+1}}{c_{k+1}} p_{k+1} \end{aligned} \tag{1.57}$$

where the last equality follows from (1.50).

It remains to show that $\hat{x}_{k+1}$ minimizes $\mathscr{R}(\cdot)$ on the subspace $x_1 + \mathscr{K}(r_1;k)$. The gradient of $\frac{1}{2}\mathscr{R}^2(\cdot)$ at $\hat{x}_{k+1}$ is equal to

$$A\hat{r}_{k+1} = \frac{1}{\theta_{k+1}} A p_{k+1}$$

since (1.50) holds.

p_{k+1} is conjugate to p_1, p_2,..., p_k thus $A\hat{r}_{k+1}$ is orthogonal to the k-plane $\mathscr{P}_k$ which means that $\hat{x}_{k+1}$ minimizes $\mathscr{R}(\cdot)$ on $x_1 + \mathscr{K}(r_1;k)$. □

The points $\hat{x}_k$, $k = 1,2,\ldots,n$ can be calculated from the points x_k, $k = 1,2,\ldots,n$ using the formulae

$$\hat{x}_1 = x_1, \quad \hat{x}_{k+1} = \frac{x_{k+1} + \beta_{k+1}\theta_k \hat{x}_k}{1 + \beta_{k+1}\theta_k},$$
$$\theta_1 = \sigma_1, \quad \theta_{k+1} = \sigma_{k+1}(1 + \beta_{k+1}\theta_k).$$

To prove this notice that

$$\frac{1}{\hat{\rho}_{k+1}} = \frac{1}{\rho_{k+1}} + \frac{1}{\hat{\rho}_k}$$

and that

$$\hat{x}_{k+1} = \hat{\rho}_{k+1}\left(\frac{x_{k+1}}{\rho_{k+1}} + \frac{\hat{x}_k}{\hat{\rho}_k}\right).$$

From the definition we have

$$\beta_{k+1} = -\frac{\rho_{k+1}}{c_k}, \quad \theta_k = -\frac{c_k}{\hat{\rho}_k}$$

thus

$$\beta_{k+1}\theta_k = \frac{\rho_{k+1}}{\hat{\rho}_k}, \;\; 1 + \beta_{k+1}\theta_k = \rho_{k+1}\left(\frac{1}{\rho_{k+1}} + \frac{1}{\hat{\rho}_k}\right) = \frac{\rho_{k+1}}{\hat{\rho}_{k+1}},$$

which imply

$$\theta_{k+1} = \frac{c_{k+1}}{\hat{\rho}_{k+1}} = \sigma_{k+1}\frac{\rho_{k+1}}{\hat{\rho}_{k+1}} = \sigma_{k+1}(1 + \beta_{k+1}\theta_k)$$

and

$$\hat{x}_{k+1} = \frac{\hat{\rho}_{k+1}}{\rho_{k+1}}\left(x_{k+1} + \frac{\rho_{k+1}}{\hat{\rho}_k}\hat{x}_k\right) = \frac{x_{k+1} + \beta_{k+1}\theta_k \hat{x}_k}{1 + \beta_{k+1}\theta_k}.$$

The conjugate gradient algorithm based on problem (1.56) can be stated without the reference to the points x_1, x_2, ..., x_n. In the space of $\hat{x}$ variables its essential ingredients mimic those of Algorithm 1.5. However, as it is shown in the next

theorem, the coefficients $\hat{\beta}_k$ are constructed in order to guarantee the conjugacy of residuals $\hat{r}_1, \ldots, \hat{r}_n$ (instead of the conjugacy of $\hat{p}_1, \ldots, \hat{p}_n$ – however the conjugacy with respect to the matrix A^2 holds).

Algorithm 1.6. (The conjugate gradient residual algorithm)

1. Choose an arbitrary $\hat{x}_1 \in \mathscr{R}^n$. Set

$$\hat{p}_1 = -\hat{r}_1 = -g(\hat{x}_1)$$

and $k = 1$.

2. Find $\hat{\alpha}_k$ which minimizes f on the line $\hat{\mathscr{L}}_k = \{\hat{x} \in \mathscr{R}^n : \hat{x} = \hat{x}_k + \hat{\alpha}\hat{p}_k, \ \hat{\alpha} \in \mathscr{R}\}$:

$$\hat{\alpha}_k = -\frac{\hat{c}_k}{\hat{d}_k},$$

where

$$\hat{c}_k = \hat{r}_k^T A \hat{p}_k$$

and

$$\hat{d}_k = \|A\hat{p}_k\|^2.$$

3. Substitute $\hat{x}_k + \hat{\alpha}_k \hat{p}_k$ for $\hat{x}_{k+1}$. If $\|\hat{r}_{k+1}\| = 0$ then STOP, otherwise calculate $\hat{\beta}_{k+1}$ according to

$$\hat{\beta}_{k+1} = \frac{\hat{c}_{k+1}}{\hat{c}_k}$$

and $\hat{p}_{k+1}$ according to

$$\hat{p}_{k+1} = -\hat{r}_{k+1} + \hat{\beta}_{k+1}\hat{p}_k, \tag{1.58}$$

with

$$\hat{r}_{k+1} = A\hat{x}_{k+1} - b.$$

4. Increase k by one and go to Step 2.

Theorem 1.10. *If $\{\hat{x}_k\}_1^n$, $\{\hat{r}_k\}_1^n$ and $\{\hat{p}_k\}_1^n$ are generated by the conjugate gradient residual algorithm then the following hold:*

$$\hat{r}_j^T A \hat{r}_i = 0, \ \ i \neq j, \quad \hat{p}_i^T A^2 \hat{p}_j = 0, \ \ i \neq j. \tag{1.59}$$

Proof. First consider the problem of minimizing $\mathscr{R}(\cdot)$ on the line $\hat{\mathscr{L}}_k$. The necessary optimality condition is

$$\hat{r}_{k+1}^T A\hat{p}_k = 0 = (\hat{r}_k + \hat{\alpha}_k A\hat{p}_k)^T A\hat{p}_k$$

thus

$$\hat{\alpha}_k = -\frac{\hat{c}_k}{\hat{d}_k},\ \ \hat{c}_k = \hat{r}_k^T A\hat{p}_k,\ \ \hat{d}_k = \|A\hat{p}_k\|^2.$$

From the conjugacy requirement we also have

$$\hat{r}_{k+1}^T A\hat{r}_k = 0$$

which implies that

$$\left(-\hat{p}_{k+1} + \hat{\beta}_{k+1}\hat{p}_k\right)^T A\hat{r}_k = 0$$

and

$$\hat{\beta}_{k+1} = \frac{\hat{p}_{k+1}^T A\hat{r}_k}{\hat{r}_k^T A\hat{p}_k} = \frac{\hat{p}_{k+1}^T A\hat{r}_k}{\hat{c}_k}.$$

Let us check now that our claims formulated in (1.59) are valid for $k = 2$. The conjugacy of $\hat{r}_2$, $\hat{r}_1$, ($\hat{p}_1 = -\hat{r}_1$) follows from the directional minimization. Next notice that

$$\begin{aligned}\hat{p}_2^T A\hat{r}_1 &= \hat{p}_2^T A\left(\hat{r}_2 - \hat{\alpha}_1 A\hat{p}_1\right)\\ &= \hat{\beta}_2\hat{p}_1^T A\hat{r}_2 - \hat{\alpha}_2\hat{p}_2^T A^2\hat{p}_1 = \hat{p}_2^T A\hat{r}_2.\end{aligned} \tag{1.60}$$

This enables us to prove the orthogonality of $A\hat{p}_2$ and $A\hat{p}_1$:

$$\begin{aligned}(A\hat{p}_2)^T A\hat{p}_1 &= \frac{1}{\hat{\alpha}_1}(A\hat{p}_2)^T(\hat{r}_2 - \hat{r}_1)\\ &= \frac{1}{\hat{\alpha}_1}\left(\hat{p}_2^T A\hat{r}_2 - \hat{p}_2^T A\hat{r}_1\right) = 0.\end{aligned}$$

Assume now that (1.59) holds for k, we will show that for $k+1$. We have already shown that (1.59) is true for $i = k$ and $j = k+1$. It is easy to show that

$$\hat{r}_{k+1} = \sum_{i=1}^{k} \gamma_i \hat{r}_i$$

for some numbers γ_i, $i = 1, 2\ldots, k$. Therefore

$$\hat{r}_j^T A\hat{r}_{k+1} = \hat{r}_j^T A\sum_{i=1}^{k}\gamma_i\hat{r}_i = 0,$$

due to our induction hypothesis. Furthermore,

$$(A\hat{p}_j)^T A\hat{p}_{k+1} = \frac{1}{\hat{\alpha}_j}\left(\hat{r}_{j+1} - \hat{r}_j\right)^T A\hat{p}_k, \tag{1.61}$$

and since

$$\hat{r}_j = \hat{r}_1 + \sum_{i=1}^{j-1} \phi_i A\hat{p}_i$$

for some $\phi_i \in \mathscr{R}$, the expression in (1.61) is equal to zero, for $j < k$, according to the induction hypothesis. For $j = k$ we can use similar arguments to those applied in (1.60) to show that

$$\hat{p}_{k+1}^T A\hat{r}_{k+1} = \hat{p}_{k+1}^T A\hat{r}_k$$

which we need to complete the proof. □

Algorithm 1.6 generates points $\hat{x}_k$, $k = 1, \ldots, n$ which can be used to obtain the points x_k, $k = 1, \ldots, n$ calculated in Algorithm 1.5. Due to (1.57) we can write

$$x_{k+1} = x_k - \alpha_k \xi_k \hat{r}_k$$

where ξ_k is some positive number. Then, if we define $\bar{\alpha}_k = \xi_k \alpha_k$,

$$\hat{p}_k = \frac{x_{k+1} - \hat{x}_k}{\bar{\alpha}_k}.$$

we come to the relations

$$\begin{aligned} x_{k+1} &= \hat{x}_{k+1} + \bar{\alpha}_{k+1}\hat{\beta}_{k+1}\hat{p}_k \\ x_{k+2} &= x_{k+1} - \bar{\alpha}_{k+1}\hat{r}_{k+1} \\ x_{k+2} &= \hat{x}_{k+1} + \bar{\alpha}_{k+1}\hat{p}_{k+1} \end{aligned}$$

provided that

$$\bar{\alpha}_{k+1}\hat{p}_{k+1} = -\bar{\alpha}_{k+1}\hat{r}_{k+1} + \bar{\alpha}_{k+1}\hat{\beta}_{k+1}\hat{p}_k$$

but this is guaranteed by (1.58).

1.5 The Method of Shortest Residuals

Our general conjugate gradient algorithm gives us the freedom of choosing the scaling sequence $\{\sigma_k\}_1^n$. The choice $\sigma_k \equiv 1$ defines our original conjugate gradient algorithm. There is another interesting choice which potentially can lead to efficient algorithms for the differentiable nonconvex case. If we take

$$\sigma_1 = 1, \quad \sigma_{k+1} = \frac{1}{1 + \beta_{k+1}}$$

then $\theta_k \equiv 1$ which results in

$$p_k = -\hat{r}_k.$$

Recall that $\hat{x}_{k+1}$ minimizes the function $\mathscr{R}(\cdot)$ on the k-plane $\mathscr{P}_k$, it is also of the form

$$\hat{x}_{k+1} = \frac{x_{k+1} + \beta_{k+1}\hat{x}_k}{1+\beta_{k+1}}.$$

From that we conclude that its residual must be the shortest on the line

$$r = \frac{r_{k+1} - \beta p_k}{1+\beta},$$

since we have

$$\begin{aligned}\hat{r}_{k+1} &= A\hat{x}_{k+1} - b = \frac{1}{1+\beta}\left(Ax_{k+1} + \beta A\hat{x}_k\right) - b \\ &= \frac{Ax_{k+1} - b + \beta\left(A\hat{x}_k - b\right)}{1+\beta} \\ &= \frac{r_{k+1} - \beta p_k}{1+\beta}.\end{aligned}$$

The shortest vector r must be perpendicular to the vector $p_k + r_{k+1}$ (cf. Fig. 1.4):

$$\left(p_k + r_{k+1}\right)^T\left(-r_{k+1} + \beta p_k\right) = \beta\|p_k\|^2 - \|r_{k+1}\|^2 = 0.$$

which gives the solution

$$\beta_{k+1} = \frac{\|r_{k+1}\|^2}{\|p_k\|^2}, \quad p_{k+1} = -\hat{r}_{k+1} = \frac{-r_{k+1} + \beta_{k+1}p_k}{1+\beta_{k+1}}.$$

We summarize our considerations on this version of a conjugate gradient algorithm in the following algorithm. Hestenes called the algorithm *the method of shortest residuals* [100].

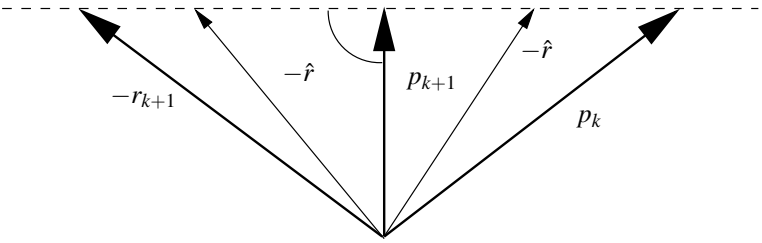

Fig. 1.4 The construction of p_{k+1} in the method of shortest residuals

Algorithm 1.7. (The method of shortest residuals)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$p_1 = -r_1 = -g_1$$

and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : x = x_k + \alpha p_k,\ \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}.$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}. If $\|r_{k+1}\| = 0$ then STOP, otherwise find p_{k+1} as the shortest vector of the form

$$p = \frac{-r_{k+1} + \beta p_k}{1+\beta}$$

by setting

$$\beta = \frac{\|r_{k+1}\|^2}{\|p_k\|^2},$$

4. Increase k by one and go to Step 2.

1.6 Rate of Convergence

We have already shown that a conjugate gradient algorithm terminates in at most n iterations. It has another remarkable feature as far as the convergence is concerned. We will prove that, under some assumptions concerning the eigenvalue structure of a matrix A, the termination can occur in much fewer than n iterations.

To this end observe first that

$$f(x) - f(\bar{x}) = \frac{1}{2}(x-\bar{x})^T A (x-\bar{x}), \tag{1.62}$$

where $\bar{x}$ is the minimum point of f. Indeed, notice that $A\bar{x} = b$ and evaluate the left-hand side of (1.62):

$$\begin{aligned} f(x) - f(\bar{x}) &= \frac{1}{2}x^T Ax - b^T x - \frac{1}{2}\bar{x}^T A\bar{x} + b^T \bar{x} \\ &= \frac{1}{2}x^T Ax - x^T A\bar{x} - \frac{1}{2}\bar{x}^T A\bar{x} + \bar{x}^T A\bar{x} \end{aligned}$$

$$= \frac{1}{2}x^T Ax - x^T A\bar{x} + \frac{1}{2}\bar{x}^T A\bar{x}$$
$$= \frac{1}{2}(x-\bar{x})^T A(x-\bar{x}).$$

We know that x_{k+1} minimizes f on the Krylov subspace $\mathscr{K}(r_1;k)$ so x_{k+1} can be expressed as

$$x_{k+1} = x_1 + \chi_1 r_1 + \chi_2 A r_1 + \chi_3 A^2 r_1 + \ldots + \chi_k A^k r_1 \tag{1.63}$$

for some $\chi_i \in \mathscr{R}$, $i = 1,\ldots,k$. We now define $\bar{P}_k(\cdot)$ to be a polynomial of degree k with coefficients χ_1, χ_2, ..., χ_k, thus

$$\bar{P}_k(A) = \chi_1 I + \chi_2 A + \ldots \chi_k A^k.$$

This enables us to write (1.63) as

$$x_{k+1} = x_1 + \bar{P}_k(A) r_1.$$

We have shown that the unscaled conjugate gradient algorithm at the kth iteration chooses x_{k+1} which minimizes the error function $\mathscr{E}$:

$$\mathscr{E}(x_{k+1}) = \min_{x \in x_1 + \mathscr{K}(r_1;k)} \mathscr{E}(x).$$

The consequence of this is that $\bar{P}_k$ is a minimum polynomial of degree k for the error function $\mathscr{E}$:

$$\|x_1 + \bar{P}_k(A) r_1 - \bar{x}\|_A^2 \leqslant \|x_1 + P_k(A) r_1 - \bar{x}\|_A^2$$

for any polynomial $P_k(\cdot)$ of degree k.

The vector $x_{k+1} - \bar{x}$ can be written as

$$x_{k+1} - \bar{x} = (I + \bar{P}_k(A)A)(x_1 - \bar{x}), \tag{1.64}$$

if we take into account that

$$r_1 = Ax_1 - b = Ax_1 - A\bar{x} = A(x_1 - \bar{x}).$$

Under our assumption that the matrix A is positive definite, its eigenvectors $\{v_i\}_1^n$ are mutually orthogonal and they span the whole space $\mathscr{R}^n$. Thus the vector $x_1 - \bar{x}$ can be expressed as their linear combination

$$x_1 - \bar{x} = \sum_{i+1}^{n} \xi_i v_i. \tag{1.65}$$

To proceed further we need to find eigenvalues and eigenvectors of the matrix $\bar{P}_k(A)$. First, observe that a normalized eigenvector v_i of A with the corresponding

eigenvalue λ_i is also an eigenvector of A^k with the corresponding eigenvalue λ_i^k. Thus, the following equation holds:

$$\begin{aligned}\bar{P}_k(A)v_i &= \left(\chi_1 I + \chi_2 A + \ldots + \chi_k A^k\right) v_i \\ &= \left(\chi_1 I + \chi_2 \lambda_i + \ldots + \chi_k (\lambda_i)^k\right) v_i \\ &= \bar{P}_k(\lambda_i) v_i. \end{aligned} \tag{1.66}$$

It shows that if v_i is an eigenvector of A with the corresponding eigenvalue λ_i, then v_i is also an eigenvector of $\bar{P}_k(A)$ with the corresponding eigenvalue $\bar{P}_k(\lambda_i)$.

If we substitute (1.65) into (1.64) and take into account (1.66) we will get

$$\begin{aligned} x_{k+1} - \bar{x} &= \sum_{i=1}^{n} \left(I + \bar{P}_k(A)A\right) \xi_i v_i \\ &= \sum_{i=1}^{n} \left(\xi_i v_i + \xi_i \bar{P}_k(A) A v_i\right) \\ &= \sum_{i=1}^{n} \left(\xi_i v_i + \xi_i \lambda_i \bar{P}_k(A) v_i\right) \\ &= \sum_{i=1}^{n} \left(1 + \lambda_i \bar{P}_k(\lambda_i)\right) \xi_i v_i. \end{aligned}$$

This together with the orthogonality of the eigenvectors $\{v_i\}_1^n$ provide a new expression for the A-norm of the vector $x_{k+1} - \bar{x}$:

$$\|x_{k+1} - \bar{x}\|_A^2 = \sum_{i=1}^{n} \lambda_i \left(1 + \lambda_i \bar{P}_k(\lambda_i)\right)^2 \xi_i^2.$$

Since $\bar{P}_k(\cdot)$ is the minimum polynomial with respect to the A-norm of $x_{k+1} - \bar{x}$ we can write

$$\|x_{k+1} - \bar{x}\|_A^2 = \min_{P_k} \sum_{i=1}^{n} \lambda_i \left(1 + \lambda_i P_k(\lambda_i)\right)^2 \xi_i^2.$$

We exploit the relation by extracting the term $(1 + \lambda_i P_k(\lambda_i))^2$:

$$\|x_{k+1} - \bar{x}\|_A^2 \leqslant \min_{P_k} \max_{1 \leqslant i \leqslant n} \left(1 + \lambda_i P_k(\lambda_i)\right)^2 \sum_{i+1}^{n} \left(\lambda_i \xi_i^2\right).$$

Since

$$\begin{aligned} \|x_i - \bar{x}\|_A^2 &= (x_1 - \bar{x})^T A (x_1 - \bar{x}) \\ &= \left(\sum_{i+1}^{n} \xi_i v_i\right)^T \sum_{i=1}^{n} \left(\xi_i A v_i\right) \\ &= \sum_{i=1}^{n} \left(\lambda_i \xi_i^2\right) \end{aligned}$$

we finally come to the relation which is the basis for our finite iteration convergence analysis:

$$\|x_{k+1}-\bar{x}\|_A^2 \leqslant \min_{P_k} \max_{1\leqslant i\leqslant n} \left(1+\lambda_i P_k(\lambda_i)\right)^2 \|x_1-\bar{x}\|_A^2. \tag{1.67}$$

The estimation (1.67) will be used in several convergence results for conjugate gradient algorithms.

Theorem 1.11. *Suppose that A has r distinct eigenvalues. Then the conjugate gradient algorithm converges in at most r iterations.*

Proof. To proof the theorem we look for a polynomial of order $r-1$, $\hat{P}_{r-1}$, for which we have

$$\max_{1\leqslant i\leqslant r} \left(1+\lambda_i \hat{P}_{r-1}(\lambda_i)\right)^2 = 0.$$

To this end we take

$$Q_r(\lambda) = \frac{(-1)^r}{\tau_1\tau_2\ldots\tau_r}(\lambda-\tau_1)(\lambda-\tau_2)\cdots(\lambda-\tau_r),$$

where τ_1, τ_2, ..., τ_r are distinct eigenvalues of A. Observe that $Q_r(0)=1$ and that Q_r has r distinct roots – each equal to an eigenvalue of A. Therefore, the polynomial

$$\hat{P}_{r-1}(\lambda) = \frac{Q_r(\lambda)-1}{\lambda}$$

is a polynomial of degree $r-1$. We can use this polynomial in the estimate (1.67) to show that

$$\max_{1\leqslant i\leqslant n} \left(1+\lambda_i \hat{P}_{r-1}(\lambda_i)\right)^2 = \max_{1\leqslant i\leqslant n} \left(Q_r(\lambda_i)\right)^2 = 0$$

since $\tau_1,\ldots,\tau_r$ are among all eigenvalues λ_1, λ_2, ..., λ_n of matrix A. This implies

$$\|x_{r+1}-\bar{x}\|_A^2 \leqslant \max_{1\leqslant i\leqslant n} \left(Q_r(\lambda_i)\right)^2 \|x_1-\bar{x}\|_A^2 = 0$$

which concludes the proof. □

Another result of finite iteration convergence concerns matrices which are rank r modifications of the identity matrix.

Theorem 1.12. *If $A = I_n + B$, where I_n is the identity matrix of dimension n and B is a symmetric positive definite matrix of rank r, then the conjugate gradient algorithm converges in at most $r+1$ iterations.*

Proof. It is straightforward to show that

$$\operatorname{span}\left\{r_1, Ar_1, \ldots, A^{k-1}r_1\right\} = \operatorname{span}\left\{r_1, Br_1, \ldots, B^{k-1}r_1\right\}$$

and since the dimension of these subspaces cannot be greater than $r+1$, the conjugate gradient algorithm must terminate in at most $r+1$ iterations. □

The estimate (1.67) can be used to establish rate of convergence of conjugate gradient algorithms. Under the assumption that eigenvalues of A are clustered the derived rate of convergence indicate the practical convergence in fewer than n iterations.

Consider first the case when A has $n-k$ eigenvalues in the interval $[a,b]$ with the other eigenvalues much greater than b.

Theorem 1.13. *Assume that $n-k$ eigenvalues of A are in the interval $[a,b]$, where $a>0$, and that the others are greater than b. Then, for an arbitrary x_1 the following estimate holds*

$$\|x_{k+1}-\bar{x}\|_A^2 \leqslant \left(\frac{b-a}{b+a}\right)^2 \|x_1-\bar{x}\|_A^2.$$

Proof. To prove the theorem we define $\hat{P}_k(\lambda)$ by

$$1+\lambda\hat{P}_k(\lambda) = \frac{2}{(a+b)\lambda_1\lambda_2\cdots\lambda_k}\left(\frac{a+b}{2}-\lambda\right)\times \\ \times(\lambda_1-\lambda)(\lambda_2-\lambda)\cdots(\lambda_k-\lambda),$$

where λ_1, λ_2, ..., λ_k are eigenvalues greater than b.

As in the proof of Theorem 1.11 we need an upper bound on $(1+\lambda\hat{P}_r(\lambda))^2$ when λ are eigenvalues in $[a,b]$:

$$\max_{a\leqslant\lambda\leqslant b}\left(1+\lambda\hat{P}_k(\lambda)\right)^2 \leqslant \max_{a\leqslant\lambda\leqslant b}\frac{\left(\lambda-\frac{(a+b)}{2}\right)^2}{\left(\frac{a+b}{2}\right)^2} \\ = \left(\frac{b-a}{b+a}\right)^2.$$

This leads to the theorem thesis since

$$\|x_{k+1}-\bar{x}\|_A^2 \leqslant \max_{a\leqslant\lambda\leqslant b}\left(1+\lambda\hat{P}_k(\lambda)\right)^2\|x_1-\bar{x}\|_A^2 \\ = \left(\frac{b-a}{b+a}\right)^2\|x_1-\bar{x}\|_A^2. \qquad \square$$

Suppose now that eigenvalues of A are clustered around k values $0<\mu_1<\mu_2\cdots<\mu_k$. It means that there exists numbers $\delta_i\geqslant 0$ such that $0<\mu_1-\delta_1\leqslant\mu_i+\delta_i\leqslant\mu_{i+1}-\delta_{i+1}$, $i=1,2,\ldots,k-1$. Furthermore, for each eigenvalue λ_j of A, $j=1,2...,n$ there exists the index i with the property

$$\lambda_j\in[\mu_i-\delta_i,\mu_i+\delta_i]. \tag{1.68}$$

Theorem 1.14. *Assume that eigenvalues of matrix A are clustered around points* μ_1, μ_2, ..., μ_k *as described in (1.68). Then the following holds*

$$\|x_{k+1}-\bar{x}\|_A^2 \leqslant M\|x_1-\bar{x}\|_A^2,$$

where

$$M=\max\left\{\frac{\delta_1^2}{\mu_1^2},\frac{\delta_2^2\left(\mu_2+\delta_2-\mu_1\right)^2}{\mu_1^2\mu_2^2},\right.$$
$$\frac{\delta_3^2\left(\mu_3+\delta_3-\mu_1\right)^2\left(\mu_3+\delta_3-\mu_2\right)^2}{\mu_1^2\mu_2^2\mu_3^2},\ldots,$$
$$\left.\frac{\left(\mu_k+\delta_k-\mu_1\right)^2\left(\mu_k+\delta_k-\mu_2\right)^2\cdots\left(\mu_k+\delta_k-\mu_{k-1}\right)^2}{\mu_1^2\mu_2^2\cdots\mu_k^2}\right\}.$$

Proof. Consider the polynomial $\hat{P}_k(\lambda)$ defined as follows

$$1+\lambda\hat{P}_k(\lambda)=\frac{(-1)^k}{\mu_1\mu_2\cdots\mu_k}\left(\lambda-\mu_1\right)\left(\lambda-\mu_2\right)\cdots\left(\lambda-\mu_k\right).$$

Take the ith eigenvalue of A. According to our assumption there exists $j\in\{1,\ldots,k\}$ such that $\lambda_i\in[\mu_j-\delta_j,\mu_j+\delta_j]$. Therefore,

$$\begin{aligned}\left|1+\lambda_i\hat{P}_k(\lambda_i)\right| \leqslant & \frac{\left|\lambda_i-\mu_j\right|}{\mu_j}\frac{\left|\lambda_i-\mu_1\right|}{\mu_1}\frac{\left|\lambda_i-\mu_2\right|}{\mu_2}\times\cdots \\ & \frac{\left|\lambda_i-\mu_{j-1}\right|}{\mu_{j-1}}\frac{\left|\lambda_i-\mu_{j+1}\right|}{\mu_{j+1}}\times\cdots \\ & \frac{\left|\lambda_i-\mu_{k-1}\right|}{\mu_{k-1}}\frac{\left|\lambda_i-\mu_k\right|}{\mu_k}.\end{aligned} \tag{1.69}$$

Observe that

$$\begin{aligned}&\left|\lambda_i-\mu_j\right| \leqslant \delta_j \\ &\left|\lambda_i-\mu_l\right| \leqslant \left|\mu_j+\delta_j-\mu_l\right|,\ \ l=1,2,\ldots,j-1 \\ &\frac{\left|\lambda_i-\mu_l\right|}{\mu_l} \leqslant \left|1-\frac{\lambda_i}{\mu_l}\right| \leqslant 1,\ \ l=j+1,j+2,\ldots,k.\end{aligned} \tag{1.70}$$

Combining (1.70) with (1.69) and

$$\|x_{k+1}-\bar{x}\|_A^2=\min_{P_k}\max_{1\leqslant i\leqslant n}\left(1+\lambda_iP_k(\lambda_i)\right)^2\|x_1-\bar{x}\|_A^2$$

gives the thesis. □

We conclude the section by giving a convergence result provided by Luenberger in [128]. The result is derived from the Kantorovich inequality and for that reason it

is in spirit of the celebrated convergence result of the steepest descent method. The result given by Luenberger is as follows.

Theorem 1.15. *If $\kappa(A)$ is the condition number of matrix A implied by the Euclidean norm, then at the kth iteration of the conjugate gradient algorithm we have*

$$\|x_{k+1}-\bar{x}\|_A \leqslant \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|x_1-\bar{x}\|_A.$$

We remind that the condition number $\kappa(A)$ implied by the Euclidean norm is defined as (cf. Appendix C)

$$\kappa(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\lambda_{max}}{\lambda_{min}}$$

Here, λ_{max}, λ_{min} are the largest and the smallest eigenvalues of A. Furthermore, $\|A\|_2$ is the matrix spectral norm defined as

$$\|A\|_2 = \sup_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2}.$$

where $\|\cdot\|_2$ is the Euclidean vector norm. The relevant convergence result for the steepest descent method is following.

Theorem 1.16. *If $\kappa(A)$ is the condition number of matrix A implied by the Euclidean norm, then at the kth iteration of the steepest descent algorithm we have*

$$\|x_{k+1}-\bar{x}\|_A \leqslant \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^k \|x_1-\bar{x}\|_A.$$

First, based on [189], we prove Theorem 1.16 relying on the Kantorovitch inequality.

Lemma 1.2. *Let B be a symmetric positive definite matrix and λ_{max}, λ_{min} its largest and smallest eigenvalues. Then*

$$\frac{\left(x^TBx\right)\left(x^TB^{-1}x\right)}{\|x\|_2^2} \leqslant \frac{\left(\lambda_{max}+\lambda_{min}\right)^2}{4\lambda_{max}\lambda_{min}}, \quad \forall x\neq 0. \tag{1.71}$$

Proof. Since B is a symmetric positive definite matrix it can be decomposed in the following way

$$B = Q^TDQ,$$

where D is a diagonal matrix with eigenvalues of B on its diagonal: $\lambda_{min}=\lambda_1$, λ_2,..., $\lambda_n=\lambda_{max}$ and Q is an orthogonal matrix. Therefore,

$$\begin{aligned}\left(x^TBx\right)\left(x^TB^{-1}x\right) &= \left(x^TQ^TDQx\right)\left(x^TQ^TD^{-1}Qx\right)\\ &= \left(Qx\right)^T D\left(Qx\right)\left(Qx\right)^T D^{-1}\left(Qx\right).\end{aligned} \tag{1.72}$$

If we introduce $y = Qx$, then

$$\xi = y^T D y = \sum_{i=1}^{n} y_i^2 \lambda_i$$

and

$$\left(x^T B x\right)\left(x^T B^{-1} x\right) = \xi\,\phi(y)$$

with

$$\phi(y) = y^T D^{-1} y = \sum_{i=1}^{n} \frac{y_i^2}{\lambda_i}.$$

In order to prove (1.71) we can assume that $\|x\|_2 = 1$, and because Q is orthogonal we also have $\|y\|_2 = 1$. Therefore,

$$\phi(y) \leqslant \sum_{i=1}^{n} \frac{1}{\lambda_i} = \psi. \tag{1.73}$$

Now, we bound the quantity ψ from the above. It can be shown that

$$\psi \leqslant \frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n}, \tag{1.74}$$

where $\lambda \in [\lambda_1, \lambda_n]$. Equation (1.74) follows from the fact that $1/x$ is a convex function and the linear function on the right-hand side of (1.74) goes through the points $(\lambda_1, 1/\lambda_1)$, $(\lambda_n, 1/\lambda_n)$.

Consider the quadratic function

$$\rho(\lambda) = \lambda\left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n}\right). \tag{1.75}$$

The function is motivated by the fact that ξ can be rephrased by

$$\xi = \sum_{i=1}^{n} w_i \lambda_i \tag{1.76}$$

with $w_i \geqslant 0$, $i = 1, \ldots, n$, $\sum_{i=1}^{n} w_i = 1$ – notice that $\|y\|_2 = 1$.

It achieves its maximum at the point $(\lambda_1 + \lambda_n)/2$ giving

$$\rho\left(\frac{\lambda_1 + \lambda_n}{2}\right) = \frac{\lambda_1 + \lambda_n}{2}\left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda_1 + \lambda_n}{2\lambda_1 \lambda_n}\right).$$

This together with (1.72)–(1.73) imply that

$$\left(x^T B x\right)\left(x^T B^{-1} x\right) \leqslant \frac{\left(\lambda_{min} + \lambda_{max}\right)^2}{4\lambda_{min}\lambda_{max}}$$

which completes the proof. □

Proof. (*of Theorem 1.16*) Consider the steepest descent method iterate

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k r_k \\ \alpha_k &= \frac{r_k^T r_k}{r_k^T A r_k} \end{aligned} \tag{1.77}$$

with (1.77) resulting from the condition $r_{k+1}^T r_k = 0$. Furthermore, we have

$$\begin{aligned} \|x_{k+1} - \bar{x}\|_A^2 &= (x_{k+1} - \bar{x})^T A (x_{k+1} - \bar{x}) \\ &= (Ax_{k+1} - b)^T (x_{k+1} - \bar{x}) \\ &= r_{k+1}^T (x_k - \alpha_k r_k - \bar{x}) \\ &= r_{k+1}^T (x_k - \bar{x}) \\ &= (r_k - \alpha_k A r_k)^T (x_k - \bar{x}) \\ &= r_k^T A^{-1} r_k - \alpha_k r_k^T r_k \\ &= \|x_k - \bar{x}\|_A^2 \left(1 - \frac{r_k^T r_k}{r_k^T A r_k} \frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right) \end{aligned}$$

since

$$\|x_k - \bar{x}\|_A^2 = r_k^T A^{-1} A A^{-1} r_k = r_k^T A^{-1} r_k.$$

Applying the Kantorovitch inequality gives the thesis. □

The proof of Theorem 1.15 relies on the estimate (1.67):

$$\|x_{k+1} - \bar{x}\|_A^2 \leqslant \min_{P_k} \max_{1 \leqslant i \leqslant n} (1 + \lambda_i P_k(\lambda_i))^2 \|x_1 - \bar{x}\|_A^2$$

which can be rephrased as

$$\|x_{k+1} - \bar{x}\|_A^2 \leqslant \min_{Q_k:\ Q_k(0)=1} \max_{\lambda \in [\lambda_{min}, \lambda_{max}]} [Q_k(\lambda)]^2 \|x_1 - \bar{x}\|_A^2. \tag{1.78}$$

It can be further developed by using the following lemma.

Lemma 1.3. *Let $[a,b]$ be a nonempty interval in $\mathscr{R}$ and let γ be any number outside this interval. Then the minimum*

$$\min_{P:\ P(\gamma)=1} \max_{t \in [a,b]} |P(t)|$$

is achieved by the polynomial

$$\hat{C}_k = \frac{C_k\left(1 + 2\frac{t-b}{b-a}\right)}{C_k\left(1 + 2\frac{\gamma-b}{b-a}\right)},$$

where $C_k(\cdot)$ is the Chebyshev polynomial of the first kind and degree k.

Proof. The proof is given in [31]. □

Using Lemma 1.3 and the following lemma proved by Saad (Theorem 6.6 in [189]) (1.78) can be stated in a different form, more convenient for further analysis.

Lemma 1.4. *If x_{k+1} is obtained at the kth iteration of the conjugate gradient algorithm, then*

$$\|x_{k+1}-\bar{x}\|_A \leqslant \frac{\|x_1-\bar{x}\|_A}{C_k(1+2\eta)} \tag{1.79}$$

with

$$\eta = \frac{\lambda_{min}}{\lambda_{max}-\lambda_{min}}.$$

Proof. (*of Theorem 1.15* based on [189]) In order to obtain the thesis we use the inequality

$$\begin{aligned} C_k(t) &= \frac{1}{2}\left[\left(t+\sqrt{t^2-1}\right)^k + \left(t+\sqrt{t^2-1}\right)^{-k}\right] \\ &\geqslant \frac{1}{2}\left(t+\sqrt{t^2-1}\right)^k \end{aligned}$$

which gives

$$\begin{aligned} C_k(1+2\eta) &\geqslant \frac{1}{2}\left(1+2\eta+\sqrt{(1+2\eta)^2-1}\right)^k \\ &\geqslant \frac{1}{2}\left(1+2\eta+2\sqrt{\eta(\eta+1)}\right)^k. \end{aligned}$$

Furthermore,

$$\begin{aligned} 1+2\eta+2\sqrt{\eta(\eta+1)} &= \left(\sqrt{\eta}+\sqrt{\eta+1}\right)^2 \\ &= \frac{\left(\sqrt{\lambda_{min}}+\sqrt{\lambda_{max}}\right)^2}{\lambda_{min}+\lambda_{max}} \\ &= \frac{\sqrt{\lambda_{min}}+\sqrt{\lambda_{max}}}{\sqrt{\lambda_{max}}-\sqrt{\lambda_{min}}} \\ &= \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}, \end{aligned}$$

where κ is the spectral condition number:

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}.$$

Finally, from (1.79) we have the thesis:

$$\|x_{k+1}-\bar{x}\|_A \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_1-\bar{x}\|_A.$$

□

1.7 Relations with a Direct Solver for a Set of Linear Equations

A conjugate gradient algorithm can be used to solve a system of linear equations:

$$Ax = b.$$

Therefore, a natural question is whether the iteration process of the algorithm is related to the matrix A inversion.

Consider the point x_{k+1} of a conjugate gradient algorithm. We have

$$x_{k+1} = x_1 + \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k,$$

where α_i, according to (1.25), can be expressed as

$$\alpha_i = -\frac{c_i}{d_i}, \;\; c_i = p_i^T r_1, \;\; d_i = p_i^T A p_i.$$

Taking this into account we can write

$$\begin{aligned} x_{k+1} = x_1 + \sum_{i=1}^{k} \alpha_i p_i &= x_1 - \sum_{i=1}^{k} \frac{p_i p_i^T r_1}{d_i} \\ &= x_1 - B_k r_1 \end{aligned}$$

with the matrix B_k equal to

$$B_k = \sum_{i+1}^{k} \frac{p_i p_i^T}{d_i},$$

thus B_k is a rank k update of a null matrix.

Since $\{p_i\}_1^n$ are conjugate directions

$$B_k A p_j = p_j, \;\; j \leqslant k \tag{1.80}$$

$$B_k A p_j = 0, \;\; j > k. \tag{1.81}$$

If $j = n$ then

$$(B_n A - I)\, p_j = 0, \;\; j = 1,2,\ldots,n. \tag{1.82}$$

Since p_j, $j = 1,2\ldots,n$ are linearly independent vectors (1.82) can happen only when

$$B_n A - I = 0, \text{ or}$$
$$B_n = A^{-1}.$$

Observe that B_n is also the result of the following iterative process

$$B_0 = 0 \tag{1.83}$$
$$B_k = B_{k-1} + \frac{p_k p_k^T}{d_k},\ k = 1,\ldots,n. \tag{1.84}$$

We summarize our derivations in the following theorem.

Theorem 1.17. *For a set of conjugate directions $\{p_k\}_1^n$ the matrices defined by (1.83)–(1.84) satisfy relations (1.80)–(1.81). Moreover, for an arbitrary x_1, $B_n = A^{-1}$ and the iterative procedure*

$$x_{k+1} = x_1 - B_k r_1,$$

finds the minimum of f in at most n iterations.

Another interesting feature of a conjugate gradient formula is the possibility of expressing p_{k+1} in the form

$$p_{k+1} = -H_{k+1} r_{k+1}$$

with H_{k+1} being a positive definite matrix.

To show that consider

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k, \tag{1.85}$$

where β_{k+1} can be given by one of the expressions:

$$\beta_{k+1} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2} \tag{1.86}$$
$$\beta_{k+1} = \frac{(r_{k+1} - r_k)^T r_{k+1}}{\|r_k\|^2} \tag{1.87}$$
$$\beta_{k+1} = \frac{(r_{k+1} - r_k)^T r_{k+1}}{(r_{k+1} - r_k)^T p_k}. \tag{1.88}$$

The formula (1.87) follows from the fact that $r_{k+1}^T r_k = 0$, as has been already proven, and formula (1.88) from the relation

$$r_k^T p_k = -\|r_k\|^2 \tag{1.89}$$

and the fact that $r_{k+1}^T p_k = 0$.

Using (1.88) and (1.89) we can transform (1.85) into

$$\begin{aligned} p_{k+1} &= -r_{k+1} + \beta_{k+1} p_k \\ &= -H_{k+1} r_{k+1} \\ &= -\left[\left(I_n - \frac{s_k y_k^T}{s_k^T y_k} \right) \left(I_n - \frac{y_k s_k^T}{s_k^T y_k} \right) + \frac{s_k s_k^T}{s_k^T y_k} \right] r_{k+1} \end{aligned} \tag{1.90}$$

with

$$\begin{aligned} s_k &= x_{k+1} - x_k = \alpha_k p_k \\ y_k &= r_{k+1} - r_k. \end{aligned}$$

The matrix H_{k+1} is positive definite, satisfies the secant equation:

$$H_{k+1} y_k = s_k \tag{1.91}$$

and is expressed by the so-called the memoryless BFGS formula (cf. [146]).

The matrix H_{k+1} is an example of a matrix from the family of matrices which are symmetric, positive definite and satisfy the secant equation (1.91). The family is called *the Broyden class* and a particular member of the class is defined recursively by

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \xi_k \rho_k v_k v_k^T \tag{1.92}$$

$$v_k = s_k - \frac{H_k y_k}{\rho_k} \tag{1.93}$$

$$\rho_k = \frac{y_k^T H_k y_k}{s_k^T y_k} \tag{1.94}$$

with $\xi_k \in [0,1]$.

When $\xi_k \equiv 0$ we have the Davidon–Fletcher–Powell (DFP) formula, while with $\xi_k \equiv 1$ we call (1.92)–(1.94) the Broyden–Fletcher–Goldbarb–Shanno (BFGS) formula. Observe that, if we assume that $H_k = I$ and $\xi_k = 1$, then formula (1.92)–(1.94) is equivalent to formula (1.90) which justifies our remark that (1.90) is the memoryless BFGS formula.

Now we prove that the matrix H_{k+1} is positive definite provided that H_k has this property and $p_k = -H_k r_k$. Consider

$$z^T H_{k+1} z = z^T H_k z + \frac{\left(z^T s_k\right)^2}{s_k^T y_k} - \frac{\left(y_k^T H_k z\right)^2}{y_k^T H_k y_k} + \xi_k \rho_k \left(v_k^T z\right)^2. \tag{1.95}$$

If we assume that $a = L_k z$ and $b = L_k y_k$, where $H_k = L_k^T L_k$ (L_k is a factor in the Cholesky decomposition of the matrix H_k – such decomposition is possible since H_k is positive definite, cf. Appendix C), then (1.95) can be written as

$$z^T H_{k+1} z = \frac{\|a\|^2\|b\|^2 - \langle a,b\rangle}{\|b\|^2} + \frac{\left(z^T s_k\right)^2}{s_k^T y_k} + \xi_k \rho_k \left(v_k^T z\right)^2.$$

Taking into account (1.94), $\xi_k \in [0,1]$ and the fact that

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k H_k r_k \\ s_k^T y_k &= \alpha_k p_k (r_{k+1} - r_k) = \alpha_k r_k^T H_k r_k > 0, \end{aligned} \tag{1.96}$$

we can show that

$$z^T H_{k+1} z > 0, \ \ \forall z \neq 0. \tag{1.97}$$

To this end refer to the Hölder inequality (cf. Appendix A), notice that $\alpha_k > 0$ and that $\|a\|^2\|b\|^2 - a^T b = 0$ if $a = \gamma b$ for some $\gamma \neq 0$ but then $z = \gamma y_k$. If at the same time $z^T s_k = 0$ then we must have $s_k^T y_k = 0$ but this is impossible due to (1.96). This proves (1.97).

Theorem 1.18. *Suppose that the sequence $\{x_k\}_1^n$ with an arbitrary x_1 is generated according to the rules*

$$x_{k+1} = x_k - \alpha_k H_k r_k \tag{1.98}$$

$$\alpha_k = \operatorname{argmin}_{\alpha>0} f(x_k - \alpha H_k r_k), \tag{1.99}$$

where the sequence $\{H_k\}_1^n$ is defined by (1.92)–(1.94). Then directions $p_k = -H_k r_k$, $k = 1,2,\ldots,n$ are conjugate:

$$p_i^T A p_j = 0, \ \ \forall i \neq j, \ \ i,j = 1,2,\ldots,n \tag{1.100}$$

and

$$H_n = A^{-1}. \tag{1.101}$$

Proof. According to (1.92) we write

$$\begin{aligned} H_{k+1} y_k &= H_k y_k + \frac{s_k s_k^T y_k}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k s_k}{s_k^T H_k s_k} + \xi_k \rho_k v_k v_k^T y_k \\ &= s_k + \xi_k \rho_k v_k v_k^T y_k. \end{aligned} \tag{1.102}$$

It is rather straightforward to prove that $v_k^T y_k = 0$ and this together with (1.102) lead to the conclusion that for any k the secant equation holds:

$$H_{k+1} y_k = H_{k+1} A s_k = s_k.$$

Now assume that (1.100) is valid for any i, j such that $i \leqslant k$, $j \leqslant k$. Moreover, assume that

$$H_{k+1} y_i = H_{k+1} A s_i = s_i, \quad 1 \leqslant i \leqslant k. \tag{1.103}$$

We will show that (1.100) and (1.103) are also valid for $k+1$. Taking into account

$$r_{k+1} = r_{i+1} + A\left(s_{i+1} + s_{i+2} + \ldots + s_k\right),\ i < k$$

and the fact that

$$s_i^T r_{i+1} = 0,\ 1 \leqslant i \leqslant k+1,$$

from our induction hypothesis we obtain

$$s_i^T r_{k+1} = s_i^T r_{i+1} = 0,\ \ 1 \leqslant i < k+1.$$

These equalities together with (1.103) enable us to write

$$s_i^T A H_{k+1} r_{k+1} = 0,\ \ 1 \leqslant i < k+1$$

and since $s_i = \alpha_i p_i$, $\alpha_i \neq 0$, $p_{k+1} = -H_{k+1} r_{k+1}$ we have

$$p_i^T A p_{k+1} = 0,\ \ 1 \leqslant i < k+1$$

which means that (1.100) is valid for $k+1$.

According to our induction hypothesis we also have

$$\begin{aligned} y_{k+1}^T H_{k+1} y_i &= y_{k+1}^T H_{k+1} A s_i \\ &= y_{k+1}^T s_i \\ &= s_{k+1}^T A s_i \\ &= s_{k+1}^T y_i = 0,\ \ 1 \leqslant i \leqslant k. \end{aligned}$$

From that, the definition of H_{k+1} and (1.103) it follows that

$$\begin{aligned} H_{k+2} y_i &= H_{k+1} y_i + \frac{s_{k+1} s_{k+1}^T y_i}{s_{k+1}^T y_{k+1}} - \\ &\quad - \frac{H_{k+1} y_{k+1} y_{k+1}^T H_{k+1} y_i}{y_{k+1}^T H_{k+1} y_{k+1}} + \xi_{k+1} \rho_{k+1} v_{k+1} v_{k+1}^T y_i \\ &= H_{k+1} y_i = s_i \end{aligned}$$

and that proves (1.103) for $k+1$.

Since p_1, p_2, ..., p_n are conjugate they are also linearly independent. Moreover, the following holds

$$H_n y_i = H_n A s_i = s_i,\ \ i = 1, 2, \ldots, n$$

and this can only happen when $H_n A = I$, thus $H_n = A^{-1}$ and this completes the proof. □

Another interesting feature of the algorithm based on the iteration (1.98)–(1.99), with H_k defined by (1.92)–(1.94) and an arbitrary symmetric positive definite H_1 is that it generates the same points x_k, $k = 1, 2, \ldots, n$ as a conjugate gradient algorithm with scaling.

Consider the following version of a conjugate gradient algorithm which uses the scaling matrix H.

Algorithm 1.8. (The preconditioned conjugate gradient algorithm with fixed scaling matrix)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$ and a symmetric positive definite matrix H. Set

$$p_1 = -Hr_1 = -Hg_1$$

and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : x = x_k + \alpha p_k,\ \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}.$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate $\hat{\beta}_{k+1}$ according to

$$\hat{\beta}_{k+1} = \frac{r_{k+1}^T H r_{k+1}}{r_k^T H r_k}$$

and p_{k+1} according to

$$p_{k+1} = -Hr_{k+1} + \hat{\beta}_{k+1} p_k. \tag{1.104}$$

4. Increase k by one and go to Step 2.

The conjugate gradient algorithm with scaling, called by Hestenes (in [100]) as *a generalized conjugate gradient algorithm*, is a conjugate gradient algorithm which solves the problem

$$\min_{y \in \mathscr{R}^n} \left[h(y) = f(Dy) = y^T D^T A D y - b^T D y \right], \tag{1.105}$$

where D is the Cholesky factor of the matrix H: $H = DD^T$. D is a nonsingular matrix which transforms the initial problem in x variables to problem (1.105) by the transformation $x = Dy$.

It is straightforward to verify that

$$\nabla h(y) = D^T g(x) \tag{1.106}$$

and if we define $\hat{p}$ by

$$D\hat{p} = p$$

then the rule (1.104) stated in y variables can be expressed by

$$\hat{p}_{k+1} = -\nabla h(y_k) + \hat{\beta}_{k+1}\hat{p}_k$$

with

$$\hat{\beta}_{k+1} = \frac{\nabla h(y_{k+1})^T \nabla h(y_{k+1})}{\nabla h(y_k)^T \nabla h(y_k)}.$$

Then

$$y_{k+1} = y_k + \alpha_k \tilde{p}_k$$

and $\nabla h(y_k)$, $k = 1,2,\ldots,n$ are mutually orthogonal which together with (1.106) imply that

$$r_k^T DD^T r_j = r_k^T H r_j = r_k^T p_j = 0,\ j = 1,2,\ldots,k-1. \tag{1.107}$$

The condition (1.107) states that x_{k+1} is the minimum point of f on the subspace

$$\begin{aligned}\mathscr{P}_k &= \left\{ x \in \mathscr{R}^n :\ x = x_1 + \sum_{i=1}^{k} \gamma_i p_i,\ \ \gamma_i \in \mathscr{R}\ \ i = 1,2,\ldots,k \right\} \\ &= \left\{ x \in \mathscr{R}^n :\ x = x_1 + \sum_{i=1}^{k} \gamma_i H r_i,\ \ \gamma_i \in \mathscr{R},\ \ i = 1,2,\ldots,k \right\}.\end{aligned} \tag{1.108}$$

The relation of a generalized conjugate gradient to the iterative process (1.98)–(1.99) with H_k specified by (1.92)–(1.94) and H_1 an arbitrary positive definite matrix is based on the observation that both methods generate points x_{k+1} which minimize f on the subspace (1.108) with $H = H_1$.

To this end suppose first that $H_1 = I$. Then, by induction arguments, we can show that

$$H_k = I + \sum_{i=2}^{k} \sum_{j=1}^{k} \zeta_{ij}^k r_i r_J^T$$

where ζ_{ij}^k are some numbers. It follows that

$$p_k = -H_k r_k = \sum_{i=1}^{k} \theta_i^k r_i$$

with some numbers θ_i^k. This implies that $x_{k+1} \in \mathscr{P}_k$ with $H = H_1$. Furthermore, p_i, $i = 1, 2 \ldots, n$ are conjugate thus x_{k+1} is the minimum point of f on $\mathscr{P}_k$. The general case of $H_1 \neq I$ can be proven similarly but in the space of variables y which are the result of the linear transformation of x which changes H_1 to the identity matrix, i.e. $x = Dy$ where $DD^T = H_1$.

It is worthwhile to mention that irrespective of the coefficient values ξ_k any algorithm from the Broyden's class generates the same sequence of points.

1.8 Limited Memory Quasi-Newton Algorithms

The major drawback of the algorithm specified in the previous section, when applied to problems with large n, is the amount of memory required to store a preconditioning matrix H. In [144] Nocedal proposed the preconditioned conjugate gradient algorithm which requires a user controlled amount of storage.

There two ways of expressing scaling matrices H_k discussed in the previous section. The first was used in the definition (1.92)–(1.94). The second was stated in [144]:

$$\begin{aligned} H_{k+1} &= \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T \\ &= V_k^T H_k V_k + \rho_k s_k s_k^T \end{aligned} \tag{1.109}$$

with

$$\rho_k = \frac{1}{y_k^T s_k} \tag{1.110}$$

$$V_k = \left(I - \rho_k y_k s_k^T\right) \tag{1.111}$$

(here presented for the BFGS formula – $\xi_k \equiv 1$). Applying the matrix update m times from the initial matrix H_1^0 (We use special notation for H_1: H_1^0 to emphasize the special role the matrix plays in the subsequent analysis.) we will end up with the matrix

$$\begin{aligned} H_{m+1} = {} & V_m^T V_{m-1}^T \cdots V_1^T H_1^0 V_1 \cdots V_{m-1} V_m + \\ & V_m^T V_{m-1}^T \cdots V_2^T \rho_1 s_1 s_1^T V_2 \cdots V_{m-1} V_m + \\ & \vdots \\ & V_m^T V_{m-1}^T \rho_{m-2} s_{m-2} s_{m-2}^T V_{m-1} V_m + \\ & V_m^T \rho_{m-1} s_{m-1} s_{m-1}^T V_m + \\ & \rho_m s_m s_m^T. \end{aligned} \tag{1.112}$$

Now suppose that we want to evaluate H_{m+2} but with the same amount of storage. Assuming that we start from the same matrix H_1^0 the matrix H_{m+2} would be as

follows

$$\begin{aligned} H_{m+2} = {} & V_{m+1}^T V_m^T \cdots V_2^T H_1^0 V_2 \cdots V_m V_{m+1} + \\ & V_{m+1}^T V_m^T \cdots V_3^T \rho_2 s_2 s_2^T V_3 \cdots V_m V_{m+1} + \\ & \vdots \\ & V_{m+1}^T V_m^T \rho_{m-1} s_{m-1} s_{m-1}^T V_m V_{m+1} + \\ & V_{m+1}^T \rho_m s_m s_m^T V_{m+1} + \\ & \rho_{m+1} s_{m+1} s_{m+1}^T. \end{aligned} \tag{1.113}$$

In general, when we evaluate H_k using vectors $\{s_i, y_i\}_{k-m}^{k-1}$ we have

$$\begin{aligned} H_k = {} & V_{k-1}^T V_{k-2}^T \cdots V_{k-m}^T H_1^0 V_{k-m} \cdots V_{k-2} V_{k-1} + \\ & V_{k-1}^T V_{k-2}^T \cdots V_{k-m+1}^T \rho_{k-m} s_{k-m} s_{k-m}^T V_{k-m+1} \cdots V_{k-2} V_{k-1} + \\ & \vdots \\ & V_{k-1}^T V_{k-2}^T \rho_{k-3} s_{k-3} s_{k-3}^T V_{k-2} V_{k-1} + \\ & V_{k-1}^T \rho_{k-2} s_{k-2} s_{k-2}^T V_{k-1} + \\ & \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{1.114}$$

The matrix update scheme (1.114) is called the limited memory BFGS update (L-BFGS update).

Notice that instead of storing $\{s_i, y_i\}_1^m$ (in the case of formula (1.112)) we have to keep $\{s_i, y_i\}_2^{m+1}$ (formula (1.113)) and that, if H_1^0 is positive definite then also H_m is positive definite (which is easy to verify) under the condition that $s_i^T y_i > 0$ for $i = 1, \ldots, m-1$. Even though (1.112) and (1.113) look very similar, there is an important difference between them – matrix H_{m+1} satisfies the secant equation, if all H_k, where H_k is defined by (1.112) with $m+1$ replaced by k, are used to define p_k according to $p_k = -H_k r_k$. The same cannot be said about H_{m+2} since in formula (1.113) we use H_1^0 instead of H_2. However, the following holds [144].

Lemma 1.5. *Suppose that H_k is defined by (1.114), $p_k = -H_k r_k$ and assume that $\{p_i\}_1^k$ are conjugate, i.e.*

$$p_i^T A p_j = 0, \quad i \neq j,\ i, j = 1, \ldots, k.$$

Then

$$H_k y_j = s_j, \quad j = k-1, k-2, \ldots, k-m\ (k > m). \tag{1.115}$$

Proof. Notice that

$$V_i y_i = \left(I - \rho_i y_i s_i^T\right) y_i = y_i - y_i = 0 \tag{1.116}$$

for $i = k, k-1, \ldots, k-m+1$ and

$$V_i y_j = \left(I - \rho_i y_i s_i^T\right) y_j = y_j - \rho_i y_i s_i^T A s_j = y_j \tag{1.117}$$

for $i > j$, which together with formula (1.114) and (1.116) prove (1.115). □

The matrices H_k defined by (1.114) can be used in preconditioned conjugate gradient algorithms, or alternatively in quasi-Newton methods. Consider a version of the preconditioned conjugate gradient algorithm proposed in [140] (see also [139], [19], [141] and [142]).

Algorithm 1.9. (The preconditioned conjugate gradient algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$p_1 = -H_1^0 g_1 = -H_1^0 r_1$$

and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : x = x_k + \alpha p_k, \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$

and substitute $x_k + \alpha_k p_k$ for x_{k+1}.

3. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate p_{k+1} according to the rule

$$p_{k+1} = -H_k r_{k+1} + \hat{\beta}_{k+1} p_k$$

where

$$\hat{\beta}_{k+1} = \frac{y_k^T H_k r_{k+1}}{y_k^T p_k}$$

and H_k is given by (1.114).

4. Increase k by one and go to Step 2.

It is important to note that the stated version of a preconditioned conjugate gradient algorithm uses not recently defined scaling matrix but the one from the previous iteration. Notice that the minimal requirements on $\hat{\beta}_{k+1}$ is to have p_{k+1} and p_k as conjugate directions, i.e.

$$p_{k+1}^T A p_k = 0$$

which holds since

$$\begin{aligned} r_{k+1} &= A x_{k+1} - b = A\left(x_k + \alpha_k p_k\right) - b \\ &= r_k + \alpha_k A p_k \end{aligned} \tag{1.118}$$

which implies $y_k = \alpha_k A p_k$ and

$$p_{k+1}^T A p_k = -\frac{r_{k+1}^T H_k y_k}{\alpha_k} + \frac{y_k^T H_k r_{k+1}}{y_k^T p_k} \cdot \frac{p_k^T y_k}{\alpha_k} = 0.$$

Our aim is to compare the above algorithm to a version of the quasi-Newton method which also uses H_k stated by (1.114).

Algorithm 1.10. (The limited memory quasi-Newton algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$p_1 = -H_1^0 g_1 = -H_1^0 r_1$$

and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \ x = x_k + \alpha p_k, \ \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$

and substitute $x_k + \alpha_k p_k$ for x_{k+1}.

3. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate p_{k+1} according to the rule

$$p_{k+1} = -H_{k+1} r_{k+1},$$

where H_{k+1} is given by (1.114).

4. Increase k by one and go to Step (2).

It is rather surprising that both the preconditioned conjugate gradient and the quasi-Newton algorithms are equivalent to the preconditioned conjugate gradient algorithm with the fixed scaling matrix $H_k \equiv H_1^0$.

Theorem 1.19. *Suppose that p_k are generated by* Algorithm 1.9 *with H_k defined by (1.114) with $H_1 = H_1^0$ and fixed m, and $\tilde{p}_k$ by* Algorithm 1.8 *with the fixed scaling matrix H_1^0. Then*

$$p_k = \tilde{p}_k, \quad \text{for } k = 1, 2, \ldots.$$

Proof. First we state some properties of the preconditioned conjugate gradient algorithm applied with the fixed scaling matrix H_1^0. Suppose that $\{\tilde{p}_i\}$ are directions generating by the algorithm. First, we show that:

$$\operatorname{span}\{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_i\} = \operatorname{span}\left\{H_1^0 \tilde{r}_1, H_1^0 \tilde{r}_2, \ldots, H_1^0 \tilde{r}_i\right\}. \tag{1.119}$$

The proof of (1.119) is by induction. It holds for $k = 1$, thus assuming it is true for $k > 1$, we have to show (1.119) for $i = k + 1$. Indeed,

$$\tilde{p}_{k+1} = -H_1^0 \tilde{r}_{k+1} + \tilde{\beta}_{k+1} \tilde{p}_k$$

which means that

$$\operatorname{span}\{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_{k+1}\} \subset \operatorname{span}\left\{H_1^0 \tilde{r}_1, H_1^0 \tilde{r}_2, \ldots, H_1^0 \tilde{r}_{k+1}\right\}$$

and the opposite inclusion could be derived similarly.

Next, notice that

$$H_1^0 A p_i = \frac{1}{\alpha_i} H_1^0 (\tilde{r}_{i+1} - \tilde{r}_i)$$

which says, due to (1.119), that

$$H_1^0 A \tilde{p}_i \in \operatorname{span}\{\tilde{p}_1, \ldots, \tilde{p}_{i+1}\}. \tag{1.120}$$

In the next step of the proof we show that directions $\tilde{p}_1, \ldots, \tilde{p}_k$ are conjugate. The proof is again by induction. It is obviously true for $k = 1$, we have to show that

$$\tilde{p}_{k+1}^T A \tilde{p}_i = 0, \;\; i = 1, \ldots, k.$$

Observe that if $\{\tilde{p}_i\}_1^k$ are conjugate then the following holds

$$\tilde{r}_{k+1}^T \tilde{p}_i = 0, \;\; i = 1, \ldots, k \tag{1.121}$$

and by taking into account (1.120) we also have the relations

$$\tilde{r}_{k+1}^T H_1^0 A \tilde{p}_i = 0, \;\; i = 1, \ldots, k-1. \tag{1.122}$$

By the definition of $\tilde{\beta}_{k+1}$ (1.121) holds for $i = k$. Notice now that we have

$$\tilde{p}_{k+1}^T A \tilde{p}_i = -\tilde{r}_{k+1}^T H_1^0 A \tilde{p}_i + \tilde{\beta}_{k+1} \tilde{p}_k^T A \tilde{p}_i. \tag{1.123}$$

The second term on the right-hand side of (1.123) is equal to zero (since $\tilde{p}_i$, $\tilde{p}_k$ are conjugate according to the induction assumption), thus (1.122) leads to

$$\tilde{p}_{k+1}^T A \tilde{p}_i = 0, \;\; i = 1, \ldots, k-1,$$

and thus we have proven that $\{\tilde{p}_1, \ldots, \tilde{p}_{k+1}\}$ are conjugate.

Next, we want to prove that

$$\tilde{r}_k^T H_1^0 \tilde{r}_i = 0, \;\; i = 1, \ldots, k-1. \tag{1.124}$$

In order to show (1.124) note that

$$H_1^0 \tilde{r}_{i+1} = -\tilde{p}_{i+1} + \tilde{\beta}_{i+1} \tilde{p}_i \in \operatorname{span}\{\tilde{p}_i, \tilde{p}_{i+1}\} \tag{1.125}$$

for $i \leqslant k-1$ and (1.121) proves (1.124).

Relations (1.121), (1.124) hold also for Algorithm 1.9. Again it can be proved by induction – if it is true for k we have

$$r_{k+1}^T p_i = 0, \;\; i = 1, \ldots, k \tag{1.126}$$

$$r_k^T H_1^0 r_i = 0, \;\; i = 1, \ldots, k-1. \tag{1.127}$$

In addition we assume that

$$H_k r_{k+1} = H_1^0 r_{k+1} \tag{1.128}$$

$$p_i^T y_j = 0, \; i \neq j, \; i, j = 1, \ldots, k \tag{1.129}$$

which are obviously satisfied for $k = 1$.

Having

$$\begin{aligned} H_k = {} & V_{k-1}^T V_{k-2}^T \cdots V_1^T H_1^0 V_1 \cdots V_{k-2} V_{k-1} + \\ & V_{k-1}^T V_{k-2}^T \cdots V_2^T \rho_1 s_1 s_1^T V_2 \cdots V_{k-2} V_{k-1} + \\ & \vdots \qquad\quad \vdots \\ & V_{k-1}^T V_{k-2}^T \rho_{k-3} s_{k-3} s_{k-3}^T V_{k-2} V_{k-1} + \\ & V_{k-1}^T \rho_{k-2} s_{k-2} s_{k-2}^T V_{k-1} + \\ & \rho_{k-1} s_{k-1} s_{k-1}^T, \end{aligned}$$

with $k \leqslant m+1$, we compute p_{k+1}:

$$p_{k+1} = -H_k r_{k+1} + \hat{\beta}_{k+1} p_k$$

with

$$\hat{\beta}_{k+1} = \frac{y_k^T H_k r_{k+1}}{y_k^T p_k}.$$

Applying (1.126)–(1.129) results in

$$H_k r_{k+1} = H_1^0 r_{k+1} \tag{1.130}$$

$$y_{k+1}^T H_k r_{k+1} = y_{k+1}^T H_1^0 r_{k+1} \tag{1.131}$$

which show that $p_{k+1} = \tilde{p}_{k+1}$ and the relations (1.126)–(1.129) are also true for $k+1$.

For $k = m+2$ we use the different formula to define H_{m+2}:

$$\begin{aligned} H_{m+2} = {} & V_{m+1}^T V_m^T \cdots V_2^T H_1^0 V_2 \cdots V_m V_{m+1} + \\ & V_{m+1}^T V_m^T \cdots V_3^T \rho_2 s_2 s_2^T V_3 \cdots V_m V_{m+1} + \\ & \vdots \end{aligned}$$

$$V_{m+1}^T V_m^T \rho_{m-1} s_{m-1} s_{m-1}^T V_m V_{m+1} +$$
$$V_{m+1}^T \rho_m s_m s_m^T V_{m+1} +$$
$$\rho_{m+1} s_{m+1} s_{m+1}^T.$$

However, also in this case, as for any $k > m+2$, we can prove (1.126)–(1.129), (1.130)–(1.131), the relations which lead to $p_k = \tilde{p}_k$. □

Algorithm 1.10 also generates directions of the preconditioned conjugate gradient algorithm with the fixed scaling matrix H_1^0.

Theorem 1.20. *Suppose that p_k are generated by* Algorithm 1.10 *with H_k defined by (1.114) with $H_1 = H_1^0$ and fixed m, and $\tilde{p}_k$ by* Algorithm 1.8 *with the fixed scaling matrix H_1^0. Then*

$$p_k = \tilde{p}_k, \quad \text{for } k = 1, 2, \ldots.$$

Proof. In order to proof the theorem it is sufficient to notice that formula (1.109) can be stated as

$$H_{k+1} = -\left(H_k + \frac{y_k^T H_k p_k}{p_k^T y_k}\right)$$

if we assume that the directional minimization is exact, i.e. $g_{k+1}^T p_k = 0$. It means that both Algorithm 1.9 and Algorithm 1.10 are equivalent since both are based on the exact line search rule. Theorem 1.19 implies the thesis. □

The results of Theorem 1.20 can be extended to quasi-Newton algorithms which use preconditioning matrices based on more general updating schemes. Following [111] consider Algorithm 1.20 with matrices H_k defined by the rule

$$H_{k+1} = \gamma_k P_k^T H_1^0 Q_k + \sum_{i=1}^{m_k} w_{ik} z_{ik}^T. \tag{1.132}$$

Here,

1. $H_1^0 \in \mathscr{R}^{n\times n}$ is a symmetric positive definite matrix that remains constant for all k and γ_k is a nonzero scalar that can be thought of as an iterative rescaling of H_1^0;
2. $P_{k+1} \in \mathscr{R}^{n\times n}$ is a matrix that is the product of projection matrices of the form

$$I - \frac{uv^T}{u^T v} \tag{1.133}$$

   where $u \in \text{span}\{y_1, \ldots, y_k\}$, $v_k \in \text{span}\{s_1, \ldots, s_k\}$ and $Q_{k+1} \in \mathscr{R}^{n\times n}$ is the product of projection matrices of the same form where u is any n-vector and $v \in \text{span}\{s_1, \ldots, s_k\}$;
3. m_k is a nonnegative integer, w_{ik}, $i = 1, \ldots, m_k$, is any n–vector, and z_{ik+1}, $i = 1, \ldots, m_{k+1}$, is any vector in $\text{span}\{s_1, \ldots, s_k\}$.

The general scheme (1.132) coupled with $d_{k+1} = -H_{k+1}g_{k+1}$ fits many known quasi-Newton methods including the Broyden family and the L-BFGS method. In particular, when

$$\gamma_k = 1,\ m_k = 0,\ P_k = I - \frac{y_k s_k^T}{s_k^T y_k},\ Q_k = I,$$

then we have the preconditioned conjugate gradient algorithm with the fixed scaling matrix H_1^0; when

$$\begin{aligned}
\gamma_k &= 1\\
m_k &= \min\{k+1, m\}\\
V_{i,k} &= \prod_{j=i}^{k}\left(I - \frac{y_i s_i^T}{s_i^T y_i}\right)\\
P_k &= Q_k = V_{k-m_k+1,k}\\
w_{ik} &= z_{ik} = \frac{\left(V_{k-m_k+i+1,k}\right)^T\left(s_{k-m_k+i}\right)}{\sqrt{\left(s_{k-m_k+i}\right)^T\left(y_{k-m_k+i}\right)}},
\end{aligned}$$

where $m > 0$ is a given integer number, then we have the limited memory BFGS method. The question is under which conditions Algorithm 1.10 with matrices H_k given by (1.132) terminates in n steps. It appears that the crucial condition is

$$P_j y_i \in \operatorname{span}\left\{y_1, \ldots, y_j\right\},\ i = 1, \ldots, j,\ j = 1, \ldots, k-1. \tag{1.134}$$

The following theorem is proved in [111].

Theorem 1.21. *Suppose that* Algorithm 1.10 *is applied with H_k defined by (1.132). Then, for each k before termination:*

$$\begin{aligned}
r_{k+1}^T s_j &= 0,\ j = 1, \ldots, k\\
s_{k+1}^T A s_j &= 0,\ j = 1, \ldots, k\\
\operatorname{span}\{s_1, \ldots, s_{k+1}\} &= \operatorname{span}\left\{H_1^0 r_1, \ldots, H_1^0 r_{k+1}\right\}
\end{aligned} \tag{1.135}$$

if and only if (1.134) hold.

Notice that (1.134) are satisfied for the L-BFGS method (cf. (1.116)–(1.117)). The immediate consequence of Theorem 1.21 is the following corollary (also proved in [111]).

Corollary 1.2. *Suppose that the assumptions of* Theorem 1.21 *are satisfied. Suppose further that $H_k r_{k+1} \neq 0$ whenever $r_{k+1} \neq 0$. Then, directions p_k generated by* Algorithm 1.10 *with H_k defined by (1.132) are equal to directions produced by the preconditioned conjugate gradient algorithm with the fixed scaling matrix H_1^0.*

1.9 Reduced-Hessian Quasi-Newton Algorithms

Quasi-Newton algorithms based on the BFGS updates when applied to strongly convex quadratic functions generate conjugate directions. Furthermore, direction p_k at the kth iteration belongs to the space $\text{span}\{H_1^0 r_1, H_1^0 r_2, \ldots, H_1^0 r_k\}$ and is the same as the direction generated by the preconditioned conjugate gradient algorithm with the fixed scaling matrix H_1^0. Suppose now that $H_1^0 = 1/\sigma I$ with $\sigma > 0$, then $p_k \in \mathscr{G}_k = \text{span}\{r_1, r_2, \ldots, r_k\}$ and it can be expressed by the formula

$$p_k = -\frac{1}{\sigma} r_k + \beta_k p_{k-1}$$

with $\beta_k = \|r_k\|^2 / \|r_{k-1}\|^2$. Assume that $\bar{p}_k$ are directions generated by the conjugate gradient algorithm, i.e. such that $\sigma = 1$. Then, we have

$$\sigma p_k = -r_k + \beta_k (\sigma p_{k-1}) \tag{1.136}$$

and since $p_1 = -1/\sigma r_1 = -1/\sigma \bar{p}_1$ the following holds

$$\bar{p}_k = \sigma p_k$$

for all k before termination occurs. Thus the sequence of points $\{x_1, \ldots, x_n\}$ evaluated by Algorithm 1.10 with H_k defined by the BFGS scheme is such that $r_{k+1} \in \mathscr{G}_k^{\perp}$ where $\mathscr{G}_k^{\perp}$ is the orthogonal complement of $\mathscr{G}_k$. Since it holds for any k the normalized gradients $r_1/\|r_1\|, \ldots, r_k/\|r_k\|$ form the orthogonal basis for $\mathscr{G}_k$.

Based on the above analysis and Lemma 12.2 (which is stated in Chap. 12 since it applies to a general nonlinear function f) we can introduce the reduced-Hessian algorithm which differs from Algorithm 1.10 only by linear algebra operations needed to evaluate p_k. Suppose that the matrix $G_k \in \mathscr{R}^{n \times k}$ has columns composed from gradients generated up to iteration k. Since $\text{rank}(G_k) = k$ there exist matrices $Q_k \in \mathscr{R}^{n \times n}$ and $R_k \in \mathscr{R}^{n \times k}$ such that Q_k is orthogonal matrix, R_k is nonsingular upper triangular matrix and the following holds [87] – cf. Appendix C.

$$G_k = Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix}.$$

If we divide Q_k into full rank matrices Z_k and W_k: $Q_k = [Z_k \; W_k]$ and transform the space of x variables to the space of x^Q by $x = Q_k x^Q$ then the matrix $B_k = H_k^{-1}$ in the new space is stated as

$$Q_k^T B_k Q_k = \begin{bmatrix} Z_k^T B_k Z_k & 0 \\ 0 & \sigma I_{n-k} \end{bmatrix}$$

and the gradient r_k as

$$Q_k^T r_k = \begin{bmatrix} Z_k^T r_k \\ 0 \end{bmatrix}.$$

$Z_k B_k Z_k$ is known in the literature (cf. e.g. [146]) as the reduced approximate Hessian while $Z_k^T r_k$ as the reduced gradient.

In the space of x^Q variables the quasi-Newton iterate has the direction $\bar{q}_k = Q_k^T p_k$ (notice that $Q_k^T = Q_k$) defined by the equation

$$\left(Q_k^T B_k Q_k\right) \bar{q}_k = \left(Q_k^T B_k Q_k\right) Q_k^T p_k = Q_k^T r_k.$$

Equations (12.32)–(12.34) imply that $q_k \in \mathscr{R}^k$ related to p_k by $p_k = Z_k q_k$ satisfies

$$Z_k^T B_k Z_k q_k = -Z_k^T r_k. \tag{1.137}$$

Equation (1.137) can be solved by doing the Cholesky factorization of $Z_k^T B_k Z_k = C_k^T C_k$ and then by solving two sets of equations

$$C_k^T z = -Z_k^T r_k \tag{1.138}$$

$$C_k q_k = z. \tag{1.139}$$

Equations (1.138) are solved with respect to z and then q_k is calculated from (1.139) by backward substitution.

The following result is proved in [65].

Lemma 1.6. *Suppose that* Algorithm 1.10 *is applied with matrices H_k defined by the BFGS scheme and with $H_1^0 = 1/\sigma I$. Then,*

$$Z_k = \left[\frac{r_1}{\|r_1\|} \; \frac{r_2}{\|r_2\|} \; \cdots \; \frac{r_k}{\|r_k\|}\right]$$

and

$$C_k = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ & a_2 & b_2 & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & a_{k-1} & b_{k-1} \\ & & & & \sqrt{\sigma} \end{bmatrix},$$

where

$$a_i = \frac{\|r_i\|^2}{\sqrt{s_i^T y_i}}, \;\; b_i = -\frac{\|r_{i+1}\|^2}{\sqrt{s_i^T y_i}}, \; i = 1, \ldots, k-1.$$

Lemma 1.6 is crucial in proving that f is minimized on a sequence of expanding linear manifolds if $\{x_k\}$ is generated by Algorithm 1.10 applied with matrices H_k defined by the BFGS scheme and with $H_1^0 = 1/\sigma I$. At the start of iteration k the curvature of f is exact on the manifold $\mathscr{M}(\mathscr{G}_k) = \{x_1 + z : \; z \in \mathscr{G}_k\}$.

Theorem 1.22. *Suppose that* Algorithm 1.10 *is applied with matrices H_k defined by the BFGS scheme and with $H_1^0 = 1/\sigma I$. At the start of iteration k we have:*

(i) x_k minimizes f on $\mathscr{M}(\mathscr{G}_{k-1})$,
(ii) the curvature of the quadratic model is exact on the subspace $\mathscr{G}_{k-1}$: $z^T B_k z = z^T A z$ for all $z \in \mathscr{G}_{k-1}$.

Proof. We follow the proof stated in [80]. In order to prove *(i)* notice that $r_k \in \mathscr{G}_{k-1}$.
We introduce the diagonal matrix

$$D_k = \sigma \mathrm{diag}\left(\frac{\sqrt{s_1^T y_1}}{\|r_1\|^2}, \frac{\sqrt{s_2^T y_2}}{\|r_2\|^2}, \ldots, \frac{\sqrt{s_{k-1}^T y_{k-1}}}{\|r_{k-1}\|^2}, \frac{1}{\sqrt{\sigma}\|r_k\|} \right),$$

then

$$D_k C_k = \begin{bmatrix} \frac{\sigma}{\|r_1\|} & -\frac{\sigma\|r_2\|}{\|r_1\|^2} & 0 & \cdots & 0 \\ & \frac{\sigma}{\|r_2\|} & -\frac{\sigma\|r_3\|}{\|r_2\|^2} & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & \frac{\sigma}{\|r_{k-1}\|} & -\frac{\sigma\|r_k\|}{\|r_{k-1}\|^2} \\ & & & & \frac{\sigma}{\|r_k\|} \end{bmatrix},$$

and if we denote by P_k the matrix whose columns are directions $p_1, p_2, \ldots, p_k$, then

$$U_k = -P_k D_k C_k$$

has columns equal to

$$u_1 = \frac{\sigma p_1}{\|r_1\|}$$
$$u_l = \frac{\sigma}{\|r_l\|}(p_l - \beta_l p_{l-1}), \quad l = 2, \ldots, k.$$

Since $r_k = -\sigma(p_k - \beta_k p_{k-1})$ we have $Z_k = -P_k D_k R_k$ and the reduced Hessian can be expressed as follows

$$Z_k^T A Z_k = C_k^T D_k P_k^T A P_k D_k C_k = C_k^T \bar{D}_k C_k$$

with

$$\bar{D}_k = \sigma^2 \mathrm{diag}\,(h_1, h_2, \ldots, h_k) \tag{1.140}$$
$$h_i = \frac{s_i^T y_i}{\|r_i\|^4} p_i^T A p_i, \quad i = 1, \ldots, k-1 \tag{1.141}$$
$$h_k = \frac{p_k^T A p_k}{\sigma \|r_k\|^2}.$$

The expression for $\bar{D}_k$ can be simplified. The exact line search rule is applied, thus $\alpha_i = -r_i^T p_i / p_i^T A p_i$ which together with $r_{i+1}^T p_i = 0$ imply that

$$s_i^T y_i = \alpha_i p_i^T (r_{i+1} - r_i) = -\alpha_i r_i^T p_i = \frac{\left(r_i^T p_i\right)^2}{p_i^T A p_i}. \tag{1.142}$$

Furthermore, from (1.136),

$$r_i^T p_i = -\frac{\|r_i\|^2}{\sigma}$$

which together with (1.142) lead to

$$\bar{D}_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \frac{1}{\alpha_k} \end{bmatrix}.$$

Notice now that the $(k-1) \times (k-1)$ submatrix of C_k built from the first $(k-1)$ rows and columns of C_k is identical to the corresponding $(k-1) \times (k-1)$ submatrix of $\bar{D}_k C_k$, and since C_k is upper triangular the remark applies also to the corresponding $(k-1) \times (k-1)$ submatrices of $C_k^T C_k$ and $C_k^T \bar{D}_k C_k$. But $C_k^T C_k = Z_k^T B_k Z_k$ and $C_k^T \bar{D}_k C_k = Z_k^T A Z_k$ which together with the representation of $Z_k = [Z_{k-1} \; r_k / \|r_k\|]$ say that $Z_{k-1}^T B_k Z_{k-1} = Z_{k-1}^T A Z_{k-1}$. Since for any z, $Z_{k-1} z \in \mathscr{G}_{k-1}$ we have proven the thesis. □

The presented outline of the reduced-Hessian BFGS method in exact arithmetic generates the same sequence of vectors x_k as Algorithm 1.10 applied with matrices H_k defined by the BFGS scheme and with $H_1^0 = 1/\sigma I$. Thus Theorem 1.22 is also applicable to Algorithm 1.10.

1.10 Limited Memory Reduced-Hessian Quasi-Newton Algorithms

The reduced-Hessian BFGS method has also its limited memory version. It is based on the transformation of the space variables into the space of x^Q variables. Unlike in the method described in the previous section the transformation matrix Q_k, which is used at every iteration of an algorithm, corresponds to the space spanned by the directions generated by an algorithm – $\mathscr{P}_k = \text{span}\{p_1, p_2, \ldots, p_k\}$. According to Lemma 12.3, stated in Chap. 12 for a general nonlinear case, the spaces $\mathscr{P}_k$ and $\mathscr{G}_k$ are identical. Furthermore, due to Theorem 12.2 the matrix Z_k is also the orthogonal basis for the space $\mathscr{P}_k$, i.e. there exists an upper triangular matrix $\bar{R}_k$ such that

$$P_k = Z_k \bar{R}_k.$$

It is shown in [80] that if we want to work with subspaces of the limited dimension, say m, then it is necessary to use the subspace $\mathscr{P}_k$ instead of $\mathscr{G}_k$. Only then we

can guarantee that the limited memory version of the reduced-Hessian BFGS converges to a solution in at most n iterations provided that the method with the exact line search is applied to a strongly convex function.

Although the limited memory version of the reduced-Hessian method is based on the subspaces $\mathscr{P}_k$ the most important role in the method is played by the subspaces spanned by the columns of the matrices

$$P_k^b = [p_l \; p_{l+1} \; \cdots p_{k-1} \; g_k],$$

where $l = k - m + 1$ (we assume that $k > m + 1$). We use the matrix P_k^b instead of P_k at the kth iteration to be able to utilize the most recent curvature information while defining the direction p_k. In analogy to (1.138) – (1.139) the direction of descent p_k is obtained, first by solving the following set of linear equations with respect to q_k

$$\bar{C}_k^T z = -\bar{Z}_k^T g_k \tag{1.143}$$

$$\bar{C}_k q_k = z, \tag{1.144}$$

and then by evaluating $p_k = \bar{Z}_k q_k$. $\bar{Z}_k$ is the orthogonal basis for P_k^b while $\bar{C}_k$ is the Cholesky factor of $\bar{Z}_k^T B_k \bar{Z}_k$, i.e.

$$P_k^b = \bar{Z}_k \bar{R}_k$$
$$\bar{C}_k^T \bar{C}_k = \bar{Z}_k^T B_k \bar{Z}_k.$$

Once p_k is calculated it replaces g_k in the matrix P_k^b, the first column of P_k^b is dropped and the new gradient g_{k+1} is added as the last column of the transformed P_k^b giving the basis for the next iteration – P_{k+1}^b. The details of all these steps, including the updates of the relevant factorizations, are presented in Sect. 12.4 where a general nonlinear case is considered.

Algorithm 1.11. (The limited memory reduced-Hessian quasi-Newton algorithm)

Parameters: the sequence $\{\sigma_k\}_1^n$ of positive numbers, positive integer number m.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set $P_1^b = [r_1]$, $\bar{Z}_1 = [r_1/\|r_1\|]$, $\bar{C}_1 = [1]$, $l = 1$ and $k = 1$.
2. Solve (1.143)–(1.144) and set $p_k = \bar{Z}_k q_k$.
3. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \; x = x_k + \alpha p_k, \; \alpha \in \mathscr{R}\}$. Substitute $x_k + \alpha_k p_k$ for x_{k+1}.
4. If $\|r_{k+1}\| = 0$ then STOP.
   Otherwise replace r_k in P_k^b with p_k.
   If $l = m + 1$ build P_{k+1}^b by dropping the first column from P_k^b and adding r_{k+1} as the last column to P_k^b.

   If $l < m + 1$ build P_{k+1}^b by adding r_{k+1} as the last column to P_k^b. Increase l by one.
   Update $\bar{Z}_k$ to $\bar{Z}_{k+1}$ and $\bar{C}_k$ to $\bar{C}_{k+1}$ according to the BFGS rule.
   Increase k by one and go to Step 2.

Algorithm 1.11 uses the sequence $\{\sigma_k\}_1^n$ of positive numbers that provide by $\sigma_k I_{n-r}$ the Hessian approximation in the subspace orthogonal to that spanned by the columns of $\bar{Z}_k$ – see Chap. 12 for the further explanation.

The following result is proved in [121].

Lemma 1.7. *Suppose that* $\sigma_1 = 1$ *in* Algorithm 1.11, *then*

$$\bar{C}_k = \begin{bmatrix} a_l/h_l & b_l & 0 & \cdots & 0 \\ & a_{l+1} & b_{l+1} & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & a_{k-1} & b_{k-1} \\ & & & & \sqrt{\sigma_k} \end{bmatrix}$$

and

$$\bar{R}_k = \begin{bmatrix} h_l d_l & h_l t_{l,l+1} & h_l t_{l,l+2} & \cdots & h_l t_{l,k-1} & 0 \\ & d_{l+1} & t_{l+1,l+2} & \cdots & t_{l+1,k-1} & 0 \\ & & \ddots & & \vdots & \vdots \\ & & & & d_{k-1} & 0 \\ & & & & & \|r_k\| \end{bmatrix},$$

where $l = \max[0, k-m+1]$ *and the scalars* a_j, b_j, $t_{i,j}$ *and* d_j *are given by*

$$a_j = \frac{\|r_j\|^2}{\sqrt{s_j^T y_j}}, \quad b_j = -\frac{\|r_{j+1}\|^2}{\sqrt{s_j^T y_j}}, \; t_{i,j} = -\frac{\|r_j\|^2}{\sigma_j \|r_i\|}$$

$$d_j = -\frac{\|r_j\|}{\sigma_j}, \; h_j = \delta_j \frac{\|p_j\| \sigma_j}{\|r_j\|},$$

with $\delta_j = 1$ *if* $j = 0$ *and* $\delta_j = -1$ *otherwise. Moreover, the search directions* p_k *satisfy*

$$p_1 = -r_1 \tag{1.145}$$

$$p_k = -\frac{1}{\sigma_k} r_k + \beta_k p_{k-1}, \; \beta_k = \frac{\sigma_{k-1}}{\sigma_k} \frac{\|r_k\|^2}{\|r_{k-1}\|^2}, \; k > 1. \tag{1.146}$$

Lemma 1.7 can be used to prove that Algorithm 1.11 converges in at most n iterations when applied to a strongly convex quadratic function.

Theorem 1.23. *If* $\sigma_1 = 1$ Algorithm 1.11 *finds the solution in at most n iterations.*

Proof. According to (1.145)–(1.146) directions p_k can be represented as

$$p_k = \frac{1}{\sigma_k} \tilde{p}_k$$

where $\tilde{p}_k$ are constructed by the conjugate gradient algorithm

$$\tilde{p}_1 = -r_1$$
$$\tilde{p}_k = -r_k + \beta_k \tilde{p}_{k-1},\ \beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}.$$

Since $\sigma_k > 0$ the directions p_k are scaled directions of the conjugate gradient algorithm thus after at most n iterations the minimum point is reached. □

Lemma 1.7 is very important since it gives the justification for using limited memory version of the reduced-Hessian BFGS method for general nonlinear problems. It is argued in [81] that the limited memory reduced-Hessian method has to be based on the matrix P_k^b, the use of G_k wouldn't guarantee the finite termination in the case of quadratics.

1.11 Conjugate Non-Gradient Algorithm

We end this chapter by considering the modifications we would have to introduce to the conjugate gradient algorithm if we allow p_1 to be an arbitrary nonzero vector non necessarily equal to $-g_1$. We will call such a method the *conjugate non-gradient method*. The need to consider this scheme follows from some experience with conjugate gradient algorithm applied to problems with nonconvex functions. It has been found that if a conjugate gradient algorithm is used with restarts then it is beneficial to use as a restarting direction a vector different from a current gradient [6]. The restarted conjugate direction method is described in more details in the next chapter.

A conjugate non-gradient algorithm is as follows.

Algorithm 1.12. (The conjugate non-gradient algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set as p_1 an arbitrary nonzero vector and $k = 1$.
2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \ x = x_k + \alpha p_k, \alpha \in \mathscr{R}\}$:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}.$$

3. Substitute $x_k + \alpha_k p_k$ for x_{k+1}. If $\|r_{k+1}\| = 0$ then STOP, otherwise calculate p_{k+1} according to

$$p_{k+1} = -r_{k+1} + \beta_{k+1}^1 p_k + \beta_{k+1}^2 p_1,$$

where

$$\beta_{k+1}^1 = \frac{r_{k+1}^T y_k}{p_k^T y_k},$$
$$\beta_{k+1}^2 = \frac{r_{k+1}^T y_1}{p_1^T y_1}.$$

4. Increase k by one and go to Step 2.

Theorem 1.24. *Suppose that the point x_k generated by the conjugate non-gradient algorithm is not the minimum point of f. Then the following hold*

$$\operatorname{span}\{p_1, p_2, \ldots, p_{k+1}\} = \mathscr{K}(p_1; k) \tag{1.147}$$
$$p_k^T A p_i = 0, \quad i = 1, 2, \ldots, k-1. \tag{1.148}$$

Proof. Equation (1.147) can be proved in the same way as the analogous relations stated in Theorem 1.7. The proof of (1.148) is based on induction. Since

$$p_2 = -r_2 + \beta_2^1 p_1$$

and

$$y_1 = \alpha_1 A p_1$$

we have

$$\begin{aligned} p_2^T A p_1 &= \left(-r_2 + \beta_2^1 p_1\right)^T A p_1 \\ &= -r_2^T A p_1 + \beta_2^1 p_1^T A p_1 \\ &= -r_2^T A p_1 + \frac{r_2^T A p_1}{p_1^T A p_1} p_1^T A p_1 = 0 \end{aligned}$$

thus (1.148) is valid for $k = 1$. Assume now that it holds for some k, we show that for $k+1$.

First, we show that p_{k+1} and p_k are conjugate:

$$\begin{aligned} p_{k+1}^T A p_k &= \left(-r_{k+1} + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1\right)^T A p_k \\ &= -r_{k+1}^T A p_k + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k^T A p_k + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1^T A p_k. \end{aligned}$$

Due to our induction hypothesis the last term is equal to zero. The sum of the first two terms is also equal to zero according to the formula for β_{k+1}^1.

Now, we consider vectors p_{k+1} and p_1. We have

$$p_{k+1}^T A p_1 = \left(-r_{k+1} + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1 \right)^T A p_1$$
$$= -r_{k+1}^T A p_1 + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1^T A p_1 + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k^T A p_1.$$

Since p_k and p_1 are conjugate the last term is equal to zero and, due to the definition of β_{k+1}^2, the first two terms sum to zero as well. (Notice that $r_{k+1}^T A p_1 = r_{k+1}^T y_1 / \alpha_1$ and $y_1^T p_1 = \alpha_1 p_1^T A p_1$.)

It remains to show that p_{k+1}, p_i, for any $1 < i < k$ are also conjugate. In order to prove that we will refer to (1.147):

$$p_{k+1}^T A p_i = \left(-r_{k+1} + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1 \right)^T A p_i$$
$$= -r_{k+1}^T A p_i + \frac{r_{k+1}^T y_k}{y_k^T p_k} p_k^T A p_i + \frac{r_{k+1}^T y_1}{y_1^T p_1} p_1^T A p_i.$$

Again the last two terms are equal to zero because $p_1, \ldots, p_k$ are conjugate. We have to show that

$$r_{k+1}^T A p_i = 0, \;\; 1 < i < k. \tag{1.149}$$

From (1.147) the following follows

$$A p_i \in \text{span}\left\{ A p_1, A^2 p_1, \ldots, A^i p_i \right\}$$
$$\subset \text{span}\left\{ p_1, p_2, \ldots, p_{i+1} \right\}. \tag{1.150}$$

But

$$r_{k+1}^T p_i = 0, \;\; i = 1, \ldots, k \tag{1.151}$$

because $p_1, \ldots, p_k$ are conjugate. Combining (1.150) and (1.151) leads to (1.149). This completes the proof. □

1.12 Notes

Conjugate direction methods were developed as iterative procedures for solving a set of linear equations. It was done in the early fifties when high-speed digital computing machines were being developed. At that time it was clear that several computational techniques had to be re-examined – even Gauss elimination method was not fully understood from a machine point of view. Therefore, the research

on computational methods for solving a set of linear equations was split into two branches. On one hand so called direct methods were being developed – several matrix factorizations such as LU, or QR were proposed together with their applications to solving linear equations. On the other hand, iterative procedures such as the method by Lanczos, or a conjugate gradient algorithm by Hestenes and Stiefel were developed at the Institute for Numerical Analysis at the University of California at Los Angeles. At that time the main interest of researches at the institute was concentrated on methods of solving simultaneous equations and the determination of eigenvalues. The application of a conjugate gradient algorithm to minimization of quadratic functions was not emphasized in Hestenes and Stiefel paper [101] – the concept conjugacy was mainly used by Hestenes to developed a general theory of quadratic forms in Hilbert spaces. Many more versions than those presented in the chapter are discussed in [100]. Papers [75], [113], [95], [74], [200], [201], [39], [96], [115], [66], [97], [98], [202], [203], [5], [60], [165], [15], [67], [16], [211], [99], [126], [48], [106], [127], [159], [56], [130], [107], [1], [131], [156], [69] and [53] discuss conjugate gradient and quasi-Newton methods, but much more comprehensive bibliography on the early development of conjugate direction methods is given by Golub and O'Leary in [86] (see also [87]).

Our presentation of conjugate gradient algorithms is based to much extent on that given by Hestenes in [100] – in particular we often refer to the minimization of f on the $\mathscr{P}_k$ plane as introduced by Hestenes. We also discuss conjugate gradient residual algorithms as proposed in [100]. The proof of Theorem 1.7 is given along the lines of the proof of the analogous theorem stated in [146]. Similar proving technique is also applied in Sect. 8 in the context of Theorem 1.19 though in the proof we also refer to the proof of the relevant theorem presented in [144].

The analysis of convergence rates of conjugate gradient algorithms is of great importance when solving problems with many variables. The monograph by Saad [189] treats iterative solvers for large, not necessarily symmetric, systems of linear equations. It pays special attention to preconditioning and its influence on the rate of convergence. Large parts of Sect. 6 borrow from [189] – the book gives also very extensive literature on modern iterative solvers for linear equations with sparse matrices.

Chapter 2
Conjugate Gradient Methods for Nonconvex Problems

2.1 Introduction

It is worthwhile to notice that when interests in conjugate gradient algorithms for quadratic problems subsided their versions for nonconvex differentiable problems were proposed. These propositions relied on the simplicity of their counterparts for quadratic problems. As we have shown in the previous chapter a conjugate gradient algorithm is an iterative process which requires at each iteration the current gradient and the previous direction. The simple scheme for calculating the current direction was easy to extend to a nonquadratic problem

$$\min_{x\in\mathscr{R}^n} f(x). \tag{2.1}$$

Since we assume that f is a nonconvex function the extension of a conjugate gradient algorithm to these functions needs the specification of the optimality conditions which would serve as a stopping criterion in new methods. Throughout Chap. 1 we dealt with quadratic functions defined by symmetric positive definite matrices. For these quadratic functions the condition that the gradient is equal to zero are both necessary and sufficient conditions of optimality of a global solution. In the case of general nonconvex functions the considered extensions of a conjugate gradient algorithm are aimed at finding a local solution to the problem (2.1). A stopping criterion of these methods refers to the necessary optimality conditions of the problem (2.1). If at points fulfilling a stopping criterion the sufficient optimality conditions are satisfied these methods find also local minimizers. The formal statement of these conditions requires several definitions beginning from the definition of a local solution to the problem (2.1) and ending up with the sufficient optimality conditions for a local solution.

Definition 2.1. The point $\bar{x}\in\mathscr{R}^n$ is a local solution to the problem (2.1) if there exists a neighborhood $\mathscr{N}$ of $\bar{x}$ such that

$$f(x)\geqslant f(\bar{x})$$

for all $x\in\mathscr{N}$. If $f(x)>f(\bar{x}), \forall x\in\mathscr{N}$ with $x\neq\bar{x}$, then $\bar{x}$ is a strict local solution.

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

The necessary optimality conditions for a local solution $\bar{x}$ are given in the following lemma.

Lemma 2.1. *Suppose that f is a continuously differentiable function in some neighborhood of $\bar{x}$ which solves the problem (2.1), then*

$$g(\bar{x}) = 0. \tag{2.2}$$

Proof. The proof of the lemma follows directly from the Taylor's expansion theorem. For any $d \in \mathscr{R}^n$ there exists $\alpha \in (0,1)$ such that (cf. Appendix A)

$$f(\bar{x}+d) = f(\bar{x}) + g(\bar{x}+\alpha d)^T d. \tag{2.3}$$

If $g(\bar{x}) \neq 0$ then we can assume $\bar{d} = -g(\bar{x})$ and due to the continuous differentiability of f we can take some small $\bar{\alpha} \in (0,1)$ such that

$$g(\bar{x}+\bar{\alpha}\bar{d})^T \bar{d} < 0 \tag{2.4}$$

and (by substituting $\bar{\alpha}\bar{d}$ for d in (2.3))

$$f(\bar{x}+\bar{\alpha}\bar{d}) = f(\bar{x}) + g(\bar{x}+\tilde{\alpha}\bar{d})^T \bar{d} \tag{2.5}$$

for some $\tilde{\alpha} \in (0,\bar{\alpha})$. Equation (2.5) together with (2.4) contradict our assumption that $\bar{x}$ is a local minimizer of f. □

A point $\bar{x}$ satisfying (2.2) is called a *stationary point*. If at a stationary point $\bar{x}$ the Hessian matrix of f is positive definite, or in other words that *sufficient optimality conditions* are satisfied:

$$g(\bar{x}) = 0 \tag{2.6}$$

$$z^T \nabla^2 f(\bar{x}) z > 0, \quad \forall z \in \mathscr{R}^n \quad \text{with } z \neq 0, \tag{2.7}$$

then $\bar{x}$ is a local solution.

Lemma 2.2. *Suppose that f is twice continuously differentiable and at a point $\bar{x}$ (2.6)–(2.7) hold. Then $\bar{x}$ is a strict local solution of the problem (2.1).*

Proof. Since f is twice continuously differentiable there exists a neighborhood of $\bar{x} - \mathscr{N}$ such that $\nabla^2 f(x)$ is positive definite for all $x \in \mathscr{N}$. There exists a positive number α such that for any normalized direction d we have $\bar{x}+\alpha d \in \mathscr{N}$. Using the Taylor's expansion theorem, for any $\bar{\alpha} \in [0,\alpha]$, we can write

$$f(\bar{x}+\bar{\alpha}d) = f(\bar{x}) + \frac{1}{2} d^T \nabla^2 f(z) d > f(\bar{x}) \tag{2.8}$$

since $z \in \mathscr{N}$. This implies that $\bar{x}$ is a local solution to the problem (2.1). □

Having the necessary optimality conditions (2.2) the straightforward extension of a conjugate gradient algorithm, stated in Chap. 1 for the quadratic problem, would be as follows.

Algorithm 2.1. (The conjugate gradient algorithm with exact line search)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set
$$d_1 = -g_1 \tag{2.9}$$
and $k = 1$.
2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \ x = x_k + \alpha d_k, \ \alpha \in (0,\infty)\}$. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise calculate d_{k+1} according to
$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k.$$
4. Increase k by one and go to Step 2.

The above algorithm is an example of an iterative optimization algorithm which generates the sequence $\{x_k\}$ in the following way

$$x_{k+1} = x_k + \alpha_k d_k$$

where α_k is chosen to decrease value of f:

$$f(x_{k+1}) < f(x_k). \tag{2.10}$$

We will demand from an iterative optimization procedure that either at some point x_l $g_l = 0$, or there exists a subsequence $\{x_{k_l}\}$ such that

$$\lim_{k_l \to \infty} g(x_{k_l}) = 0.$$

In Step 2 condition (2.10) is satisfied by requiring that α_k is a minimum of f along a direction d_k. Contrary to the quadratic case this is not a simple algebraic problem. In general we have to apply an iterative procedure to find an approximate solution to the problem

$$\min_{\alpha > 0} [\phi(\alpha) = f(x_k + \alpha d_k)]. \tag{2.11}$$

2.2 Line Search Methods

In order to avoid solving problem (2.11) several rules for finding α_k, which guarantee the global convergence of an optimization algorithm, have been proposed. These rules related to the problem (2.11) are called the *directional minimization rules*.

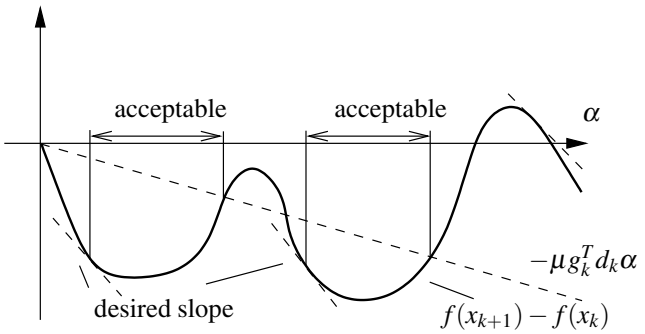

Fig. 2.1 The Wolfe line search rule

The most popular rule is expressed by the Wolfe conditions [207], [208]:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k \tag{2.12}$$

$$g(x_k + \alpha_k d_k)^T d_k \geqslant \eta g_k^T d_k, \tag{2.13}$$

where $0 < \mu < \eta < 1$ (cf. Fig. 2.1).

Condition (2.12) is sometimes referred as *Armijo condition* although this name is often reserved for another directional minimization rule which will be discussed later. Condition (2.12) stipulates a decrease of f along d_k. Notice that if

$$g_k^T d_k < 0, \tag{2.14}$$

i.e. d_k is the *direction of descent*, then there exists, under some conditions imposed on f, $\tilde{\alpha}$ such that for all $\alpha \in [0, \tilde{\alpha})$ (2.12) is satisfied:

$$f(x_k + \alpha d_k) - f(x_k) \leqslant \mu \alpha g_k^T d_k$$

since $0 < \mu < 1$.

Condition (2.13) is called the *curvature condition* and its role is to force α_k to be sufficiently far away from zero which could happen if only condition (2.12) were to be used. Notice that there exists $\hat{\alpha} > 0$ such that for any $\alpha \in [0, \hat{\alpha}]$ (2.13) will not be satisfied:

$$g(x_k + \alpha d_k)^T d_k < \eta g_k^T d_k$$

for any $\alpha \in [0, \hat{\alpha}]$ since $0 < \eta < 1$.

If we wish to find a point α_k, which is closer to a solution of problem (2.11) than a point satisfying (2.12)–(2.13), we can impose on α_k the *strong Wolfe conditions*:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k \tag{2.15}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \leqslant \eta |g_k^T d_k| \tag{2.16}$$

with $0 < \mu < \eta < 1$. In contrast to the Wolfe conditions $g(x_k + \alpha_k d_k)^T d_k$ cannot be arbitrarily large.

The directional minimization rule which does not refer to the curvature condition, called the *Armijo rule*, (the corresponding procedure is often called *backtracking line search*) is defined by the following algorithm:

Algorithm 2.2. (The Armijo line search algorithm)

Parameters: $s > 0$, $\mu,\ \beta \in (0,1)$.

1. Set $\alpha = s$.
2. If

$$f(x_k + \alpha d_k) - f(x_k) \leqslant \mu \alpha g_k^T d_k \tag{2.17}$$

   substitute α for α_k and STOP.
3. Substitute $\beta\alpha$ for α, go to Step 2.

The essence of the Armijo rule is that α_k is accepted only if α_k/β does not satisfy the Armijo condition therefore we avoid too small values of α_k which could hamper the global convergence. As we will show later, at each iteration initial s can assume different values, provided they lie in some set $[s_{min}, s_{max}]$ and $0 < s_{min} < s_{max}$.

The Wolfe conditions can be modified by taking, as the accuracy reference in the Armijo and the curvature conditions, $-\|d_k\|^2$ instead of $g_k^T d_k$. Assume that

$$g_k^T d_k \leqslant -\|d_k\|^2$$

then the *modified Wolfe conditions* are

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu \alpha_k \|d_k\|^2 \tag{2.18}$$

$$g(x_k + \alpha_k d_k)^T d_k \geqslant -\eta \|d_k\|^2, \tag{2.19}$$

where $0 < \mu < \eta < 1$. Notice that condition (2.18) is weaker in comparison to the Armijo condition while condition (2.19) is stronger than the curvature condition. The equivalence of these conditions holds when $g_k^T d_k = -\|d_k\|^2$.

The above mentioned line search methods have any practical meaning if we can guarantee finding α_k satisfying them in a finite number of operations. However, first of all we have to show that such α_k exist.

Lemma 2.3. *Assume that f is continuously differentiable and that it is bounded from below along the line $\mathscr{L}_k = \{x \in \mathscr{R}^n :\ x = x_k + \alpha d_k,\ \alpha \in (0,\infty)\}$. Suppose also that d_k is a direction of descent ((2.14) is satisfied). If $0 < \mu < \eta < 1$ then there exist nonempty intervals of step lengths satisfying the Wolfe and the strong Wolfe conditions.*

Proof. The proof is standard one – we follow that given in [146]. Consider the line $l(\alpha) = f(x_k) + \mu\alpha g_k^T d_k$. Since $0 < \mu < 1$ it must intersect the graph of $\phi(\alpha)$. Suppose that $\tilde{\alpha}$ is the smallest intersection point. Therefore, we have

$$f(x_k + \tilde{\alpha}d_k) = f(x_k) + \mu\tilde{\alpha}g_k^T d_k.$$

Furthermore, from the mean value theorem there exists $\hat{\alpha} \in (0, \tilde{\alpha})$ such that

$$f(x_k + \tilde{\alpha}d_k) = f(x_k) + \tilde{\alpha}g(x_k + \hat{\alpha}d_k)^T d_k$$

and since $0 < \mu < \eta < 1$ we also have

$$g(x_k + \hat{\alpha}d_k)^T d_k = \mu g_k^T d_k > \eta g_k^T d_k.$$

Due to our smoothness assumption (2.12)–(2.13) are satisfied by all α in some neighborhood of $\hat{\alpha}$. The interval on which the strong Wolfe conditions hold is the same since $f(x_k + \tilde{\alpha}d_k) - f(x_k) < 0$ and

$$|g(x_k + \hat{\alpha}d_k)^T d_k| < \eta |g_k^T d_k|.$$

It completes the proof. □

In order to show that there exists a nonempty interval of step lengths satisfying the modified Wolfe conditions we will provide a procedure which determines α_k in a finite number of operations.

Lemma 2.4. *Assume that f is continuously differentiable and that it is bounded from below along the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \ x = x_k + \alpha d_k, \ \alpha \in (0, \infty)\}$. Suppose also that d_k is a direction of descent ((2.14) is satisfied). If $0 < \mu < \eta < 1$ then there exists a procedure which finds α_k satisfying (2.18)–(2.19) in a finite number of operations.*

Proof. The proof is based on the paper [133]. Let $\varepsilon > 0$ be such that $\eta > \mu + \varepsilon$. Let α^0 be an arbitrary positive number and set $\alpha_N^0 = \infty, \ \alpha_\mu^0 = 0$. The procedure, which we propose, produces a sequence of points $\{\alpha^l\}$ constructed in the following way. If

$$f(x_k + \alpha^l d_k) - f(x_k) \leqslant -(\mu + \varepsilon)\alpha^l \parallel d_k \parallel^2 \tag{2.20}$$

we set $\alpha_\mu^{l+1} = \alpha^l, \ \alpha_N^{l+1} = \alpha_N^l$ and $\alpha_\mu^{l+1} = \alpha_\mu^l, \ \alpha_N^{l+1} = \alpha^l$ otherwise. Next, we substitute α^{l+1} by $(\alpha_\mu^l + \alpha_N^l)/2$ if $\alpha_N^{l+1} < \infty$, or by $2\alpha^{l+1}$ if $\alpha_N^{l+1} = \infty$.

If for every α^l generated by the procedure we have that α^l satisfies (2.20) then α_N^l will remain equal to infinity and $f(x_k + \alpha^l d_k) \to -\infty$. Therefore, let us suppose that there exists α^l such that (2.20) is not fulfilled. This means that at some iteration doubling has started and we have the sequences $\{\alpha_N^l\}, \ \{\alpha_\mu^l\}$ such that $\alpha_N^l - \alpha_\mu^l \to 0, \ \alpha_N^l \to \hat{\alpha}$, because either α_N^{l+1} or α_μ^{l+1} is set to $(\alpha_N^l + \alpha_\mu^l)/2$. Moreover, let us assume that for every l, α^l does not satisfy (2.19) and (2.20) (otherwise

we have found the desired coefficient α_k). In this case $\alpha_\mu^l \to \hat{\alpha}$ and $\hat{\alpha}$ satisfies (2.20). If we have (2.19) for α^l infinitely often then the procedure will terminate after finite number of iterations, because $\alpha^l \to \hat{\alpha}$, f is continuous and (2.18) will have to be satisfied.

Now, let us suppose that (2.19) holds only for the finite number of α^l, thus

$$g(x_k + \alpha^l d_k)^T d_k < -\eta \|d_k\|^2$$

for infinitely many times. This leads to

$$g(x_k + \hat{\alpha} d_k)^T d_k \leqslant -\eta \|d_k\|^2.$$

Because

$$f(x_k + \alpha_N^l d_k) - f(x_k) > -(\mu + \varepsilon)\alpha_N^l \|d_k\|^2, \tag{2.21}$$

thus

$$f(x_k + \alpha_N^l d_k) - f(x_k + \hat{\alpha} d_k) > -(\mu + \varepsilon)(\alpha_N^l - \hat{\alpha})\|d_k\|^2 \tag{2.22}$$

and

$$\begin{aligned} -\eta \|d_k\|^2 &\geqslant \lim_{l \to \infty} g(x_k + \alpha^l d_k)^T d_k = g(x_k + \hat{\alpha} d_k)^T d_k \\ &= \lim_{l \to \infty} \frac{f(x_k + \alpha_N^l d_k) - f(x_k + \hat{\alpha} d_k)}{\alpha_N^l - \hat{\alpha}} \geqslant -(\mu + \varepsilon)\|d_k\|^2, \end{aligned} \tag{2.23}$$

but this is impossible since $\eta > \mu + \varepsilon$. □

Since the Wolfe conditions are a special case of the modified Wolfe conditions we have also shown how to find α_k, in a finite number of operations, which satisfies (2.12)–(2.13).

Following the first part of the proof of Lemma 2.3 we can easily show that the backtracking procedure will find an α_k which fulfills the Armijo condition in a finite number of operations provided that d_k is a direction of descent.

We have specified the main elements of algorithms which belong to the class of line search methods. An algorithm from the class can be stated as follows.

Algorithm 2.3. (The line search method)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$ and a descent direction d_1. Set $k = 1$.
2. Find $\alpha_k > 0$ which approximately solves the problem

$$\min_{\alpha > 0} f(x_k + \alpha d_k).$$

   Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise find d_{k+1}, a direction of descent.
4. Increase k by one and go to Step 2.

2.3 General Convergence Results

The main convergence result, which will be extensively exploited in the context of several algorithms, is due to Zoutendijk [214]. It refers to the angle θ_k between d_k and the *steepest descent direction* $-g_k$ defined by

$$\cos\theta_k = \frac{-g_k^T d_k}{\|g_k\|\|d_k\|}. \tag{2.24}$$

Essentially the Zoutendijk's result states that if the θ_k is always less than π then the algorithm is globally convergent in the sense that

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{2.25}$$

Equation (2.25) is a straightforward consequence of the following theorem.

Theorem 2.1. *Suppose that* $\{x_k\}$ *is generated by* Algorithm 2.3 *and*

(i) α_k *satisfies the Wolfe conditions,*
(ii) f *is bounded below in* $\mathscr{R}^n$,
(iii) f *is continuously differentiable in an open set* $\mathscr{N}$ *containing the level set* $\mathscr{M} = \{x \in \mathscr{R}^n : f(x) \leqslant f(x_1)\}$,
(iv) $g(x)$ *is Lipschitz continuous – there exists* $L > 0$ *such that*

$$\|g(y) - g(x)\| \leqslant L\|y - x\| \tag{2.26}$$

for all $x,\ y \in \mathscr{N}$.

Then

$$\sum_{k\geqslant 0} \cos^2\theta_k \|g_k\|^2 < \infty.$$

Proof. From the curvature condition

$$(g_{k+1} - g_k)^T d_k \geqslant (\eta - 1)\, g_k^T d_k \tag{2.27}$$

and since g is Lipschitz continuous we also have

$$|\,(g_{k+1} - g_k)^T d_k| \leqslant \alpha_k L\|d_k\|^2. \tag{2.28}$$

Combining (2.27) with (2.28) we obtain

$$\alpha_k \geqslant \frac{\eta - 1}{L\|d_k\|^2} g_k^T d_k \tag{2.29}$$

which together with the Armijo condition results in

$$f(x_{k+1}) - f(x_k) \leqslant -\mu \frac{1-\eta}{L} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2}. \tag{2.30}$$

Taking into account (2.24) we rewrite (2.30) as

$$f(x_{k+1}) - f(x_k) \leqslant -c\cos^2\theta_k \|g_k\|^2$$

with $c = \mu(1-\eta)/L$. After summing this expression over all indices from one to k we come to

$$f(x_{k+1}) - f(x_1) \leqslant -c\sum_{l=1}^{k} \cos^2\theta_l \|g_l\|^2.$$

Since f is bounded below we must have

$$\sum_{k=1}^{\infty} \cos^2\theta_k \|g_k\|^2 < \infty. \tag{2.31}$$

□

Observe that if there exists $m > 0$ such that

$$\cos\theta_k \geqslant m \tag{2.32}$$

for all k then from (2.31) we have

$$\lim_{k\to\infty} \cos^2\theta_k \|g_k\|^2 = 0$$

and

$$\lim_{k\to\infty} \|g_k\| = 0$$

which guarantees global convergence of the algorithm as we have already stated.

If the modified Wolfe conditions or the Armijo rule are used in general only weaker convergence results can be established.

Theorem 2.2. *Suppose that assumptions of* Theorem 2.1 *hold and* α_k *satisfies the modified Wolfe conditions. Then*

$$\lim_{k\to\infty} \|d_k\| = 0.$$

Proof. From the modified curvature condition we have

$$(g_{k+1} - g_k)^T d_k \geqslant -(\eta - 1)\|d_k\|^2. \tag{2.33}$$

On the other hand, from the Lipschitz condition, we come to

$$|(g_{k+1} - g_k)^T d_k| \leqslant \alpha_k L \|d_k\|^2$$

which combined with (2.33) give the lower bound on α_k:

$$\alpha_k \geqslant \frac{1-\eta}{L}$$

for all k.

Using arguments similar to those in the proof of Theorem 2.1 we can show that

$$f(x_{k+1}) - f(x_1) \leqslant -\mu c \sum_{l=1}^{k} \|d_l\|^2$$

with $c = (1-\eta)/L$. Since f is bounded below we must have

$$\sum_{k=1}^{\infty} \|d_k\|^2 < \infty$$

from which it follows

$$\lim_{k\to\infty} \|d_k\| = 0.$$

□

Theorem 2.2 is useful in proving global convergence if we can show that there exists $M < \infty$ such that

$$\|g_k\| \leqslant M\|d_k\|$$

for all k. Suppose, for example, that

$$d_k = -B_k^{-1} g_k \tag{2.34}$$

and there exists $M < \infty$ such that

$$\|B_k\|_2 \leqslant M$$

where $\|\cdot\|_2$ is the matrix norm induced by the Euclidean norm. Then, we will have

$$\|g_k\| \leqslant M\|d_k\|$$

and from Theorem 2.2 global convergence follows

$$\lim_{k\to\infty} \|g_k\| = 0.$$

Yet another convergence result can be provided for algorithms with the Armijo rule (cf. [9]), or the *modified Armijo rule* that is defined by Algorithm 2.2 in which the condition (2.17) is replaced by the condition

$$f(x_k + \alpha d_k) - f(x_k) \leqslant -\mu\alpha\|d_k\|^2 \tag{2.35}$$

and it is assumed that

$$g_k^T d_k \leqslant -\|d_k\|^2. \tag{2.36}$$

Theorem 2.3. *Suppose that assumptions (ii)–(iii) of* Theorem 2.1 *are satisfied and* α_k *is determined by the Armijo rule, or the modified Armijo rule. If for any convergent subsequence* $\{x_k\}_{k\in K}$ *such that*

$$\lim_{k\to\infty,\ k\in K} g_k \neq 0$$

we have

$$\limsup_{k\to\infty,k\in K} \|d_k\| < \infty,$$
$$\liminf_{k\to\infty,k\in K} \left|g_k^T d_k\right| > 0$$

then,

$$\lim_{k\to\infty,k\in K} g_k^T d_k = 0 \tag{2.37}$$

in the case of the Armijo rule, or

$$\lim_{k\to\infty,k\in K} \|d_k\| = 0 \tag{2.38}$$

in the case of the modified Armijo rule.

Proof. The proof for the Armijo rule is given in [9], therefore we concentrate on the proof of the part of theorem related to the modified Armijo rule.

As in the proof of the previous theorem we can show that

$$f(x_{k+1}) - f(x_1) \leqslant -\mu \sum_{l=1}^{k} \alpha_l \|d_l\|^2, \tag{2.39}$$

in the case of the modified Armijo rule. On the basis on the assumption *(iii)*, (2.39) implies that

$$\lim_{k\to\infty} \alpha_k \|d_k\|^2 = 0. \tag{2.40}$$

Consider now any convergent subsequence $\{x_k\}_{k\in K}$ and let its accumulation point be $\bar{x}$. Suppose also that (2.38) is not satisfied, therefore, from (2.39), we must have

$$\lim_{k\to\infty,k\in K} \alpha_k = 0. \tag{2.41}$$

The above means that there exists $k_1 \in K$ such that for all $k \in K,\ k \geqslant k_1$

$$f(x_k + (\alpha_k/\beta)d_k) - f(x_k) > -\mu(\alpha_k/\beta)\|d_k\|^2 \tag{2.42}$$

since otherwise we would have $\alpha_k = s$ for k sufficiently large contradicting (2.41).

Since the sequence $\{d_k\}_{k\in K}$ is bounded there exists convergent subsequence of $\{d_k\}_{k\in K}$ which for the simplicity of presentation will be denoted in the same way. Now (2.42) can be rewritten as

$$\frac{f(x_k+\bar{\alpha}_k d_k)-f(x_k)}{\bar{\alpha}_k} > -\mu\|d_k\|^2$$

where $\bar{\alpha}_k = \alpha_k/\beta$.

If we take limits of both sides of the above expression and notice that f is continuously differentiable we obtain

$$g(\bar{x})^T\bar{d} \geqslant -\mu\|\bar{d}\|^2, \tag{2.43}$$

where

$$\lim_{k\to\infty, k\in K} d_k = \bar{d}.$$

On the other hand, d_k satisfies (2.36) thus

$$g(\bar{x})^T\bar{d} \leqslant -\|\bar{d}\|^2$$

which contradicts (2.43). This means that our assumption (2.41) has been wrong and the proof of (2.38) is concluded. □

As the possible application of Theorem 2.3 consider the algorithm with the direction defined by (2.34). However, now we assume that the condition numbers of B_k are bounded (cf. Appendix C):

$$\kappa(B_k) = \|B_k\|_2\|B_k^{-1}\|_2 \leqslant M < \infty$$

for all k. In this case eigenvalues of B_k^{-1} lie in the subinterval $[\gamma,\Gamma]$ where γ, Γ are smallest and largest eigenvalues of B_k^{-1}. We have

$$-g_k^T d_k = g_k^T B_k^{-1} g_k \geqslant \gamma\|g_k\|^2$$
$$\|d_k\| \leqslant \|B_k^{-1}\|\|g_k\| \leqslant \Gamma\|g_k\|.$$

and all assumptions of Theorem 2.3 are satisfied. Therefore, for any convergent subsequence $\{x_k\}_{k\in K}$ its limit point $\bar{x}$ satisfies $g(\bar{x}) = 0$.

All convergence theorems stated so far refer only to stationary points. In order to show global convergence to local minimum points we need the assumption that every stationary point satisfies sufficient optimality conditions.

The following theorem is due to Bertsekas [9] – its extension to constrained problems is given in [176].

Theorem 2.4. *Assume that*

(i) f is twice continuously differentiable,
(ii) the sequence $\{x_k\}$ constructed by Algorithm 2.3 *is such that*

$$\begin{aligned} f(x_{k+1}) &\leqslant f(x_k) \\ \alpha_k &\leqslant s \\ \|d_k\| &\leqslant c\|g_k\| \end{aligned}$$

for all k, where $s > 0$, $c > 0$,
(iii) every convergent subsequence of $\{x_k\}$ is convergent to a stationary point.

Then, for every local minimum point $\bar{x}$ of f at which $\nabla^2 f(\bar{x})$ is positive definite there exists a neighborhood of $\bar{x}$ – $\mathscr{N}$ such that, if $x_{\bar{k}} \in \mathscr{N}$ for some $\bar{k}$ then $x_k \in \mathscr{N}$ for all $k \geqslant \bar{k}$ and $x_k \to_{k\to\infty} \bar{x}$.

Proof. We follow the proof stated in [9]. Since $\nabla^2 f(\bar{x})$ is positive definite exists $\bar{\varepsilon} > 0$ such that $\nabla^2 f(x)$ is positive definite for all x satisfying $\|x - \bar{x}\| \leqslant \bar{\varepsilon}$. Introduce

$$\gamma = \min_{\|z\|=1,\ \|x-\bar{x}\|\leqslant\bar{\varepsilon}} z^T \nabla^2 f(x) z$$

$$\Gamma = \max_{\|z\|=1,\ \|x-\bar{x}\|\leqslant\bar{\varepsilon}} z^T \nabla^2 f(x) z$$

which will satisfy $\Gamma \geqslant \gamma > 0$.

Define now an open set

$$\mathscr{N} = \left\{ x \in \mathscr{R}^n :\ \|x - \bar{x}\| < \bar{\varepsilon},\ f(x) - f(\bar{x}) < \frac{1}{2}\gamma \left(\frac{\bar{\varepsilon}}{1 + sc\Gamma} \right)^2 \right\}.$$

We will show that the set $\mathscr{N}$ is the one stated in the thesis, i.e. if $x_{\bar{k}} \in \mathscr{N}$ then $x_k \in \mathscr{N}$ for all $k \geqslant \bar{k}$.

Assume that $x_{\bar{k}} \in \mathscr{N}$, then from the mean value theorem (cf. Appendix A) the following relation holds

$$\frac{1}{2}\gamma\|x_{\bar{k}} - \bar{x}\| \leqslant f(x_{\bar{k}}) - f(\bar{x}) \leqslant \frac{1}{2}\gamma \left(\frac{\bar{\varepsilon}}{1 + sc\Gamma} \right)^2$$

which implies that

$$\|x_{\bar{k}} - \bar{x}\| \leqslant \frac{\bar{\varepsilon}}{1 + sc\Gamma}. \tag{2.44}$$

Furthermore, from the iterative rule

$$\begin{aligned} \|x_{\bar{k}+1} - \bar{x}\| &= \|x_{\bar{k}} - \alpha_{\bar{k}} d_{\bar{k}} - \bar{x}\| \\ &\leqslant \|x_{\bar{k}} - \bar{x}\| + sc\|g_{\bar{k}}\|. \end{aligned}$$

Using again the mean value theorem but for g we can write

$$\|x_{\bar{k}+1} - \bar{x}\| \leqslant (1 + sc\Gamma)\,\|x_{\bar{k}} - \bar{x}\| \tag{2.45}$$

since $g(\bar{x}) = 0$.

Taking into account (2.44) and (2.45) we conclude that

$$\|x_{\bar{k}+1} - \bar{x}\| < \bar{\varepsilon}$$

and since

$$f(x_{\bar{k}+1}) \leqslant f(x_{\bar{k}}) < \frac{1}{2}\gamma\left(\frac{\bar{\varepsilon}}{1 + sc\Gamma}\right)^2$$

thus we also have $x_{\bar{k}+1} \in \mathscr{L}$.

If we denote by $\bar{\mathscr{N}}$ the closure of $\mathscr{N}$ then $x_k \in \bar{\mathscr{N}}$ for all $k \geqslant \bar{k}$. $\bar{\mathscr{N}}$ is a compact set thus $\{x_k\}_{k \geqslant \bar{k}}$ has at least one accumulation point which, by the assumption, must be the stationary point. Since $\bar{x}$ is the only stationary point in $\bar{\mathscr{N}}$ we have shown that the sequence $\{x_k\}_{k \geqslant \bar{k}}$ converges to $\bar{x}$. □

Theorem 2.4 has many applications. We can use it to prove that the steepest descent method with the Armijo rule is convergent to local minimum points if all of them satisfy sufficient optimality conditions. Moreover, the quasi-Newton methods defined by

$$d_k = -D_k g_k$$

with bounded positive definite matrices D_k and with the Wolfe rules are also globally convergent to local minimum points provided that the step selection procedure determines bounded α_k. As we will see in the next section practical step selection algorithms usually fulfill this requirement.

2.4 Moré–Thuente Step-Length Selection Algorithm

In this section we discuss a procedure which finds a step-length parameter α_k, in a finite number of iterations, that fulfills the strong Wolfe conditions. The strong Wolfe conditions guarantee global convergence of several line search methods. They are employed in quasi-Newton methods (cf. [29, 166]) and in conjugate gradient methods (cf. [2, 43, 78, 125]).

From mathematical point of view we aim at the approximate minimization of the function

$$\phi(\alpha) = f(x_k + \alpha d_k) : [0, \infty) \to \mathscr{R}.$$

By an approximate minimizing point of $\phi(\cdot)$ we mean an $\alpha > 0$ such that

$$\phi(\alpha) \leqslant \phi(0) + \mu\dot{\phi}(0)\alpha \tag{2.46}$$

$$|\dot{\phi}(\alpha)| \leqslant \eta|\dot{\phi}(0)|. \tag{2.47}$$

Since we assume that d_k is a direction of descent we also have that $\dot{\phi}(0) < 0$. The condition (2.47) is called the curvature condition since it implies that the average curvature of ϕ on $[0, \alpha]$ is positive

$$\dot{\phi}(\alpha) - \dot{\phi}(0) \geqslant (1-\eta)|\dot{\phi}(0)|.$$

In addition to conditions (2.46)–(2.47) additional requirements are imposed on α:

$$0 < \alpha_{min} \leqslant \alpha \leqslant \alpha_{max} < \infty.$$

The main reason for requiring a lower bound α_{min} is to avoid too small values of α which could cause some numerical problems. The upper bound α_{max} is needed if the function ϕ is unbounded below.

In this section we present the step-length selection procedure proposed by Moré and Thuente [136]. We describe this procedure since it has been verified in professional optimization software, it has guaranteed convergence in a finite number of operations. As far as the other step-length algorithms are concerned we have to mention that suggested by Fletcher [70] which led to algorithms developed by Al-Baali and Fletcher [3], Moré and Sorensen [135]. Furthermore, we could recommend implementations of algorithms by Dennis and Schnabel [55], by Lemaréchal [120] and by Fletcher [71]. In the next section we introduce to a new line search procedure proposed by Hager and Zhang in [91]. The interesting features of the procedure are the precautions taken to cope with a finite precision calculations.

The procedure given by Moré and Thuente requires that $0 < \mu < \eta < 1$ although value of $\mu < 1/2$ is often assumed, because if ϕ is a quadratic function with $\dot{\phi}(0) < 0$ and $\ddot{\phi}(0) > 0$, then the global minimizer $\bar{\alpha}$ of ϕ satisfies

$$\phi(\bar{\alpha}) = \phi(0) + \frac{1}{2}\bar{\alpha}\ddot{\phi}(0) \tag{2.48}$$

which means that $\bar{\alpha}$ satisfies (2.46) only if $\mu \leqslant 1/2$. Furthermore, if $\mu < 1/2$ is used $\alpha = 1$ is accepted at final iterations of the Newton and quasi-Newton methods.

The procedure by Moré and Thuente finds a point α which belongs to the set

$$T(\mu) = \left\{\alpha :\ \phi(\alpha) \leqslant \phi(0) + \alpha\mu\dot{\phi}(0),\ |\dot{\phi}(\alpha)| \leqslant \mu|\dot{\phi}(0)|\right\}$$

thus the parameter η is used only in the convergence test.

The iteration of the procedure consists of three major steps.

Algorithm 2.4. (Iteration of the Moré–Thuente step-length selection algorithm)

Input and output data: $I_i \subset [0,\infty]$.

1. Choose a trial point $\alpha_t \in I_i \bigcap [\alpha_{min}, \alpha_{max}]$.
2. Test for convergence.
3. Update I_i.

Initially we assume that $I_0 = [0,\infty]$, and then we find $I_i = [\alpha_l^i, \alpha_u^i]$ such that α_l^i, α_u^i satisfy

$$\psi(\alpha_l^i) \leqslant \psi(\alpha_u^i),\ \psi(\alpha_l^i) \leqslant 0,\ \dot{\psi}(\alpha_l^i)\left(\alpha_u^i - \alpha_l^i\right) < 0, \tag{2.49}$$

where

$$\psi(\alpha) = \phi(\alpha) - \phi(0) - \mu\dot{\phi}(0)\alpha.$$

The endpoints of I_i are chosen in such a way that I_i always contains α from $T(\mu)$ – the following lemma gives basis for that [136].

Lemma 2.5. *Let I be a closed interval with endpoints $[\alpha_l,\ \alpha_u]$ which satisfy (2.49). Then there exists an $\bar{\alpha}$ in I with $\psi(\bar{\alpha}) \leqslant \psi(\alpha_l)$ and $\dot{\psi}(\bar{\alpha}) = 0$, in particular $\bar{\alpha} \in I \bigcap T(\mu)$.*

The updated interval I_{i+1} is determined, on the basis of the trial point α_t^i, by the following procedure.

1. If $\psi(\alpha_t^i) > \psi(\alpha_l^i)$, then $\alpha_l^{i+1} = \alpha_l^i$, $\alpha_u^{i+1} = \alpha_t^i$.
2. If $\psi(\alpha_t^i) \leqslant \psi(\alpha_l^i)$ and $\psi^{'}(\alpha_t^i)\left(\alpha_l^i - \alpha_t^i\right) > 0$, then $\alpha_l^{i+1} = \alpha_t^i$ and $\alpha_u^{i+1} == \alpha_u^i$.
3. If $\psi(\alpha_t^i) \leqslant \psi(\alpha_l^i)$ and $\psi^{'}(\alpha_t^i)\left(\alpha_l^i - \alpha_t^i\right) < 0$, then $\alpha_l^{i+1} = \alpha_t^i$ and $\alpha_u^{i+1} = \alpha_l^i$.
4. Substitute α_t^i for α_i.

The aim of the updating procedure is to choose I_i in such a way that $I_i \bigcap T(\mu)$ is not empty and then to move to I_{i+1} guaranteeing that $I_{i+1} \bigcap T(\mu)$ is also non empty.

In general, with the exception of some very rare degenerate cases (cf. [136]) the above step-length selection procedure either terminates at α_{min}, α_{max} or an interval $I_i \subset [\alpha_{min}, \alpha_{max}]$ is generated.

We can show that the procedure terminates at α_{min} if the sequence $\{\alpha_i\}$ is decreasing and the following holds

$$\psi(\alpha_i) > 0 \quad \text{or} \quad \dot{\psi}(\alpha_i) \geqslant 0,\ \ k = 0,1,\ldots. \tag{2.50}$$

On the other hand $\{\alpha_i\}$ terminates at α_{max} if

$$\psi(\alpha_i) \leqslant 0 \quad \text{and} \quad \dot{\psi}(\alpha_i) < 0,\ \ k = 0,1,\ldots, \tag{2.51}$$

– then the sequence of trial points is increasing.

Therefore, the finite operation convergence of the above step-length selection procedure requires assumptions imposed on α_{min} and α_{max}. Namely, we require that

$$\psi(\alpha_{min}) \leqslant 0 \ \text{ and } \ \dot{\psi}(\alpha_{max}) < 0, \tag{2.52}$$

$$\psi(\alpha_{max}) > 0 \ \text{ and } \ \dot{\psi}(\alpha_{max}) \geqslant 0. \tag{2.53}$$

Under these conditions we can prove the following convergence result.

Theorem 2.5. *If the bounds α_{min} and α_{max} satisfy (2.52)–(2.53), then the step-length selection algorithm terminates in a finite number of iterations with $\alpha_i \in T(\mu)$, or the sequence $\{\alpha_i\}$ converges to some $\bar{\alpha} \in T(\mu)$ with $\dot{\psi}(\bar{\alpha}) = 0$. If the search algorithm does not terminate in a finite number of steps, then there is an index i_0 such that the endpoints of the interval I_i satisfy $\alpha_l^i < \bar{\alpha} < \alpha_u^i$. Moreover, if $\psi(\bar{\alpha}) \geqslant 0$, then $\dot{\psi}$ changes sign on $\left[\alpha_l^i, \bar{\alpha}\right]$ for all $i \geqslant i_0$, while if $\psi(\bar{\alpha}) < 0$, then $\dot{\psi}$ changes sign on $\left[\alpha_l^i, \bar{\alpha}\right]$ or $\left[\bar{\alpha}, \alpha_u^i\right]$ for all $i \geqslant i_0$.*

The algorithm for the selection of a trial point α_t contains several safeguards. In order to guarantee termination at α_{min}, provided that (2.50) holds, we use

$$\alpha_t^{i+1} \in \left[\alpha_{min}, \max\left[\delta_{min}\alpha_t^i, \alpha_{min}\right]\right]$$

for some factor $\delta_{min} < 1$.

If (2.51) holds then taking

$$\alpha_t^{i+1} \in \left[\min\left[\delta_{max}\alpha_t^i, \alpha_{max}\right]\right]$$

for some $\delta_{max} > 1$, ensures that eventually $\alpha_t = \alpha_{max}$.

In the other case α_t is determined by using quadratic or cubic interpolation – the details are given in [136].

The step-length selection procedure is especially important for conjugate gradient algorithms where good approximation of a minimum point of ϕ, obtained through few function and gradient evaluations, is needed. In general we cannot expect that $\alpha_k = 1$ is accepted close to a minimum point of the problem (2.1). In this situation the first trial point α_t^0 can have big influence on the whole efficiency of the line search method.

In practice two choices for α_t^0 are employed. In the first approach we assume that the first-order change in the function at iterate x_k will be the same as that obtained at the previous step. This implies that $\alpha_{k-1} g_{k-1}^T d_{k-1} = \alpha_t^0 g_k^T d_k$, therefore

$$\alpha_t^0 = \alpha_{k-1} \frac{g_{k-1}^T d_{k-1}}{g_k^T d_k}.$$

Another choice is to take as α_t^0 the minimum point of the quadratic defined by $f(x_{k-1})$, $f(x_k)$ and $\dot{\phi}(0) = g_k^T d_k$. This leads to

$$\alpha_t^0 = \frac{2\left(f(x_k) - f(x_{k-1})\right)}{\dot{\phi}(0)}. \tag{2.54}$$

One advantage of using (2.54) is that if $x_k \to \bar{x}$ superlinearly (cf. Sect. 2 in Chap. 4) then the ratio in (2.54) converges to 1. Therefore, taking

$$\alpha_t^0 = \min[1, 1.01\alpha_t^0]$$

ensures that $\alpha_k = 1$ will be tried and accepted for large k.

2.5 Hager–Zhang Step-Length Selection Algorithm

The step-length selection procedure proposed by Hager and Zhang [91] is also intended for optimization algorithms which use the Wolfe conditions. The procedure aims at finding the point α which satisfies both (2.46) and (2.47). The drawback of Moré–Thuente step-length selection algorithm is that it is designed for computer with an infinite precision arithmetic. If the method is used on computers with a finite precision arithmetic then the algorithm can fail to find, in a finite number of operations (or in any number of operations), a point α satisfying (2.46)–(2.47).

Hager and Zhang give in [91] the following example. Suppose that we want to minimize the function $f(x) = 1 - 2x + x^2$ in a neighborhood of the point $x = 1$. In Fig. 2.2 we present the graph generated by a Matlab using an Intel equipped PC. The solid line is obtained by generating 10,000 values of x on the interval $[1.0 - 2.5 \times 10^{-8}, 1 + 2.5 \times 10^{-8}]$ and then by connecting computed values of f by straight lines.

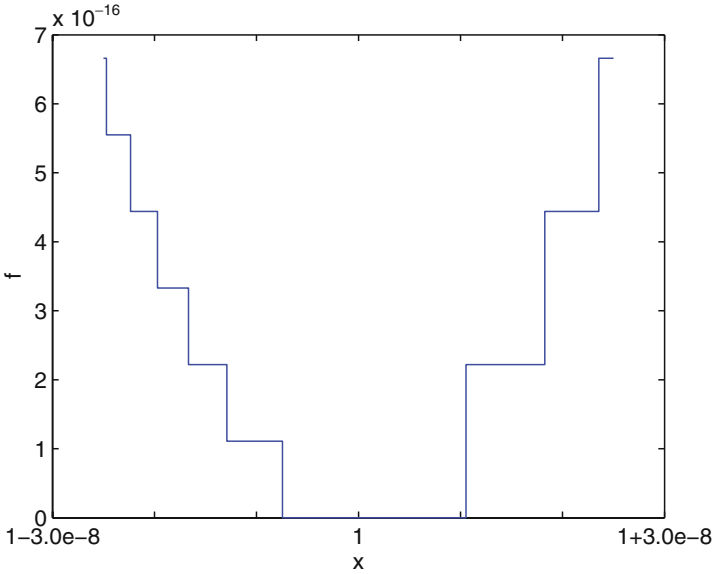

Fig. 2.2 Plot of $f(x) = 1 - 2x + x^2$

The striking feature of the solid line is a flat section on the subinterval $[1.0 - 0.9 \times 10^{-8}, 1 + 0.9 \times 10^{-8}]$ to which a zero value is assigned. Notice that its width can be estimated as 1.8×10^{-8} which is much wider then the machine precision $\varepsilon = 2.2 \times 10^{-16}$. The phenomenon is easily explained by a well-known effect of the reduction of significant digits. Near $x = 1$, both $1 - 2x$ and x^2 have almost equal values close to 1 in absolute terms but with opposite signs. As the result in order to evaluate f at the points close to 1 we have to subtract numbers of almost equal values. The error of these operations is equal to the square root of the machine precision [90].

The same phenomenon does not occur for the derivative of f: $\dot{f} = 2(x - 1)$ in which case a flat section near the point $x = 1$ has the width of 1.6×10^{-16} (cf. Fig. 2.3). It implies that if we base our line search procedure on the derivative of f then the accuracy of determining its minimum point is close to the arithmetic precision 2.2×10^{-16} while the accuracy of the minimum value of f with the order of the square root of the arithmetic precision.

This suggested to Hager and Zhang to use instead of (2.46) the criterion based on the derivative of ϕ. This can be achieved by using a quadratic approximation of ϕ. The quadratic interpolating polynomial q that matches $\phi(\alpha)$ at $\alpha = 0$, and $\dot{\phi}(\alpha)$ at $\alpha = 0$ and $\alpha = \alpha_k$ (which is unknown) is given by

$$q(\alpha) = \phi(0) + \dot{\phi}(0)\alpha + \frac{\dot{\phi}(\alpha_k) - \dot{\phi}(0)}{2\alpha_k}\alpha^2.$$

Now, if we replace ϕ by q in the first Wolfe condition (2.46) we will obtain

$$q(\alpha_k) - q(0) \leqslant \mu \dot{q}(\alpha_k)$$

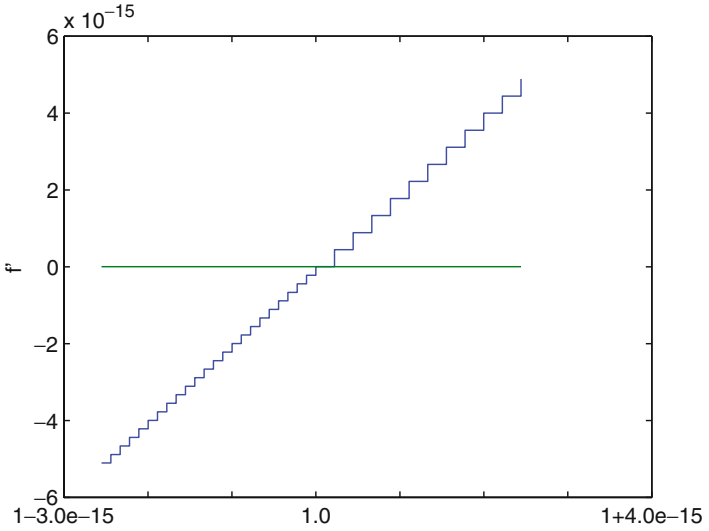

Fig. 2.3 Plot of the derivative $\dot{f}(x) = 2(x - 1)$

which is translated into the condition

$$\frac{\dot{\phi}(\alpha_k)-\dot{\phi}(0)}{2}\alpha_k+\dot{\phi}(0)\alpha_k \leqslant \mu\alpha_k\dot{\phi}(0)$$

which can also be restated as

$$\dot{\phi}(\alpha_k) \leqslant (2\delta-1)\dot{\phi}(0) \tag{2.55}$$

where δ must satisfy $\delta \in (0, \min[0.5, \eta]]$.

The procedure aims at finding α_k which either satisfies the Wolfe conditions: (2.46) and

$$\dot{\phi}(\alpha) \geqslant \eta\dot{\phi}(0)$$

– we call these criteria **LSR1**; or the condition (2.55) together with

$$\begin{aligned}\phi(\alpha) &\leqslant \phi(0)+\varepsilon_k\\ \varepsilon_k &= \varepsilon|f(x_k)|\end{aligned}$$

with ε to be some small positive values – we call these criteria **LSR2**.

The major iterations of the procedure are as follows.

Algorithm 2.5. (The Hager–Zhang step-length selection algorithm)

Parameter: $\gamma \in (0,1)$.

1. Choose $[a_0, b_0]$ and set $k = 0$.
2. If either **LSR1**, or **LSR2** are satisfied at a_k then STOP.
3. Define a new interval $[a,b]$ by using *secant* procedure: $[a,b] = secant(a_k, b_k)$.
4. If $b-a > \gamma(b_k-a_k)$, then $c = (a+b)/2$ and use the *update* procedure: $[a,b] = update(a,b,c)$.
5. Increase k by one, substitute $[a,b]$ for $[a_k,b_k]$, go to Step 2.

The essential part of the algorithm is the *update* function which changes the current bracketing interval $[a,b]$ into a new one $[\bar{a},\bar{b}]$ with help of the additional point which is either obtained by a bisection step, or a secant step.

Algorithm 2.6. (The *update* procedure)

Parameters: $\theta \in (0,1)$, ε_k.
Input data: a, b, c.
Output data: $\bar{a}$, $\bar{b}$.

1. If $c \notin (a,b)$, then $\bar{a} = a$, $\bar{b} = b$ and STOP.
2. If $\dot{\phi}(c) \geqslant 0$, then $\bar{a} = a$, $\bar{b} = c$ and STOP.
3. If $\dot{\phi}(c) < 0$ and $\phi(c) \leqslant \phi(0)+\varepsilon_k$, then $\bar{a} = c$, $\bar{b} = b$ and STOP.

4. If $\dot{\phi}(c) < 0$ and $\phi(c) > \phi(0) + \varepsilon_k$, then $\bar{a} = a$, $\bar{b} = c$ and perform the steps:
   (a) Calculate $d = (1-\theta)\bar{a} + \theta\bar{b}$; if $\dot{\phi}(d) \geqslant 0$, then set $\bar{b} = d$ and STOP.
   (b) If $\dot{\phi}(d) < 0$ and $\phi(d) \leqslant \phi(0) + \varepsilon_k$, then set $\bar{a} = d$ and go to step (a).
   (c) If $\dot{\phi}(d) < 0$ and $\phi(d) > \phi(0) + \varepsilon_k$, then set $\bar{b} = d$ and go to step (a).

The *update* procedure finds the interval $[a,b]$ such that

$$\phi(a) < \phi(0) + \varepsilon_k, \;\; \dot{\phi}(a) < 0, \;\; \phi^{'}(b) \geqslant 0. \tag{2.56}$$

Eventually, a nested sequence of intervals $[a_k, b_k]$ is constructed which converges to the point satisfying either **LSR1**, or **LSR2**.

Another part of the Hager–Zhang algorithm is the *secant* procedure which updates interval with the help of secant steps. Suppose that $\dot{\phi}(\alpha)$ is a convex function on the interval $[a,b]$ with the property: $\dot{\phi}(a) < 0$, $\dot{\phi}(b) > 0$. Then in order to approximate a point at which $\dot{\phi}$ vanishes we construct a linear approximation of $\dot{\phi}$ on $[a,b]$:

$$\dot{\phi}_{la}(\alpha) = \frac{\dot{\phi}(b) - \dot{\phi}(a)}{b-a}(\alpha - a) + \dot{\phi}(a).$$

Under the assumption that $\dot{\phi}$ is convex, the point at which $\dot{\phi}_{la}$ is equal to zero lies on the right of a and is given by formula:

$$\alpha_0 = \frac{a\dot{\phi}(b) - b\dot{\phi}(a)}{\dot{\phi}(b) - \dot{\phi}(a)}.$$

Once α_0 is obtained $\bar{a}$ is set to α_0 and α_0 is used to perform the second secant step this time based on a and $\bar{\alpha}$ yielding a point $\bar{b}$ which must lie on the left of b provided that $\dot{\phi}$ is convex. Similarly secant steps are found in the case of concave $\dot{\phi}$ – details and the full description of the *secant* procedure is given in [91].

The Hang–Zhang line search procedure finds α_k fulfilling either **LSR1** or **LSR2** in a finite number of operations as stated in the following theorem proved in [91].

Theorem 2.6. *Suppose that ϕ is continuously differentiable on an interval $[a_0, b_0]$, where (2.56) holds. If $\delta \in (0, 0.5)$, then the Hager–Zhang step-length selection algorithm terminates at a point satisfying either* **LSR1***, or* **LSR2**.

In [91] it is also shown that under some additional assumptions the interval width $|b_k - a_k|$ tends to zero with root convergence order $1 + \sqrt{2}$.

The Hager–Zhang step-length selection procedure was successfully applied, among others, in the conjugate gradient algorithm which is discussed in Sect. 10.

2.6 Nonlinear Conjugate Gradient Algorithms

The first conjugate gradient algorithm for nonconvex problems was proposed by Fletcher and Reeves in 1964 [73]. Its direction of descent is

$$d_k = -g_k + \beta_k d_{k-1} \tag{2.57}$$

with

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}. \tag{2.58}$$

The algorithm is a straightforward extension of a conjugate gradient algorithm for the quadratic with the particular choice of a coefficient β_k. We know that the other equivalents for β_k are

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{\|g_{k-1}\|^2}, \tag{2.59}$$

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{d_{k-1}^T (g_k - g_{k-1})}, \tag{2.60}$$

$$\beta_k = \frac{\|g_k\|^2}{d_{k-1}^T (g_k - g_{k-1})}. \tag{2.61}$$

Although these formulae are equivalent for the quadratic the performance of a nonlinear conjugate gradient algorithm strongly depends on coefficients β_k. In general, formula (2.59) proposed by Polak and Ribière [161], is preferred over formula (2.58). On the other hand, formula (2.60) which is equivalent to formula (2.59) when directional minimization is exact, is used in several, more sophisticated conjugate gradient algorithms which will be discussed later.

Yet, the Fletcher–Reeves formula has an attractive property as far as convergence analysis is concerned. It appeared, after several failed attempts to prove global convergence of any conjugate gradient algorithm, that positiveness of a coefficient β_k is crucial as far as convergence is concerned.

Before presenting global convergence of the Fletcher–Reeves algorithm with inexact directional minimization and for nonconvex problems we provide the analysis of the behavior of conjugate gradient algorithms with different coefficients β_k. We follow Powell [168] who provided for the first time systematic arguments in favor of the Polak–Ribière algorithm.

Consider a conjugate gradient algorithm in which α_k is determined according to the exact directional minimization rule

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha d_k) \tag{2.62}$$

with d_k defined by (2.57). If we denote by θ_k the angle between d_k and $-g_k$ then the following relation holds

$$\|d_k\| = \sec\theta_k \|g_k\|. \tag{2.63}$$

Furthermore, using the fact that $g_{k+1}^T d_k = 0$ we can state

$$\beta_{k+1}\|d_k\| = \tan\theta_{k+1}\|g_{k+1}\| \tag{2.64}$$

and combining these two relations we have

$$\begin{aligned}\tan\theta_{k+1} &= \sec\theta_k \frac{\|g_{k+1}\|}{\|g_k\|} \\ &> \tan\theta_k \frac{\|g_{k+1}\|}{\|g_k\|}\end{aligned} \tag{2.65}$$

if β_k is defined by (2.58) since $\sec\theta > \tan\theta$ for $\theta \in [0, \pi/2)$.

Powell'a arguments are as follows. If θ_k is close to $\pi/2$ then d_k is not a good direction of descent and the iteration takes very small step which implies that $g_{k+1} - g_k$ is small. This results in $\|g_{k+1}\|/\|g_k\|$ close to one and, from (2.65), θ_{k+1} is also close to $\pi/2$. Thus, the Fletcher–Reeves algorithm does not have self-correcting mechanism with this respect.

Consider now the Polak–Ribière version of a conjugate gradient algorithm. If, due to a small step, $g_{k+1} - g_k$ is close to zero, then according to rule (2.59) the coefficient β_{k+1} is also close to zero

$$\beta_{k+1} \leqslant \frac{\|g_{k+1}\|\|g_{k+1} - g_k\|}{\|g_k\|^2}$$

and the next direction is taken as the steepest direction: $d_{k+1} = -g_{k+1}$ since

$$\tan\theta_{k+1} \leqslant \sec\theta_k \frac{\|g_{k+1} - g_k\|}{\|g_k\|} \tag{2.66}$$

from (2.63)–(2.64). This self-correcting behavior of the Polak–Ribière algorithm makes it more efficient method for most problems. This analysis suggests the following restarting criteria in any conjugate gradient algorithms. If

$$|g_{k+1}^T g_k| \geqslant 0.2\|g_{k+1}\|^2 \tag{2.67}$$

holds then an algorithm should assume $d_{k+1} = -g_{k+1}$ [168].

Powell's observation led him to one of the first global convergence result on a conjugate gradient algorithm for a general nonconvex function. Although he assumed that the directional minimization is exact his result is important nonetheless, since he noticed that the prerequisite for the global convergence of the Polak–Ribière algorithm is

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{2.68}$$

This condition will appear in the analysis of the global convergence of the Polak–Ribière versions of a conjugate gradient algorithm.

Theorem 2.7. *Assume that*

(i) the set

$$\mathscr{M} = \{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\} \tag{2.69}$$

is bounded,
(ii) g is a Lipschitz continuous function,
(iii) condition (2.68) is satisfied.

Then, the Polak–Ribière version of Algorithm 2.1 *(β_k defined by (2.59)) generates the sequence $\{x_k\}$ such that*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{2.70}$$

Proof. Assume the opposite to (2.70) – there exists a positive ε and an integer k_1 such that

$$\|g_k\| \geqslant \varepsilon \tag{2.71}$$

for all $k \geqslant k_1$. Furthermore, due to (2.68), there exists $k_2 \geqslant k_1$ such that

$$\|g_{k+1} - g_k\| \leqslant \frac{1}{2}\varepsilon \tag{2.72}$$

for all $k \geqslant k_2$. Combining (2.66), (2.71)–(2.72) and the fact that

$$\sec\theta \leqslant 1 + \tan\theta$$

holds for all $\theta \in [0, \pi/2)$, we come to the relation

$$\tan\theta_{k+1} \leqslant \frac{1}{2}\,(1 + \tan\theta_k) \tag{2.73}$$

for all $k \geqslant k_2$.

Applying (2.73) recursively m times results in

$$\begin{aligned}\tan\theta_{k+1} &\leqslant \frac{1}{2} + \frac{1}{4} + \cdots \left(\frac{1}{2}\right)^{k_2+m+1} \left(1 + \tan\theta_{k_2}\right) \\ &\leqslant 1 + \tan\theta_{k_2}.\end{aligned}$$

This means that θ_k is bounded away from $\pi/2$. If we now apply Theorem 2.1 we come to the contradiction with (2.71). This completes the proof. □

The conjugate gradient algorithms considered so far have relied on the exact directional minimization, from now on we will concentrate on the convergence analysis of the methods which require only approximate minimization of f on the line $\mathscr{L}_k$.

Algorithm 2.7. (The standard conjugate gradient algorithm for nonconvex problems)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -g_1$$

   and $k = 1$.
2. Find α_k which satisfies the Wolfe conditions (2.12)–(2.13), or the strong Wolfe conditions (2.15)–(2.16). Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise evaluate β_{k+1} and calculate d_{k+1} according to

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k.$$

4. Increase k by one and go to Step 2.

2.7 Global Convergence of the Fletcher–Reeves Algorithm

Although, as explained in the previous section, the Fletcher–Reeves algorithm is inferior to the Polak–Ribière version of a conjugate gradient procedure, it was the method for which the global convergence properties were established first. The credit goes to Al-Baali who using ingenious analysis showed that the Fletcher–Reeves method is globally convergent for general nonconvex problems even if the directional minimization is not exact. His arguments are given below since they influenced research on nonlinear conjugate gradients algorithms for some time suggesting that the global convergence of conjugate gradient algorithms requires some assumptions on parameters in the step-length selection procedure based on the strong Wolfe conditions.

The global convergence analysis which we present in this section follows some unified pattern. First of all we use Theorem 2.1 to obtain a final conclusion about convergence. Observe that if a directional minimization is exact, i.e.

$$g_k^T d_{k-1} = 0,$$

then we can easily show that

$$\cos\theta_k = \frac{\|g_k\|}{\|d_k\|}.$$

This implies, from Theorem 2.1, that

$$\sum_{k=1}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty \tag{2.74}$$

and, if one can show that $\{\cos\theta_k\}$ is bounded from zero, that

$$\lim_{k\to\infty}\|g_k\| = 0. \tag{2.75}$$

Unfortunately, this can be shown only under strong assumptions concerning function f, for example when f is a strictly convex function (cf. [161]). If we cannot bound the sequence $\{\|g_k\|/\|d_k\|\|\}$ from zero then we are content with the result weaker than (2.75), namely

$$\liminf_{k\to\infty}\|g_k\| = 0. \tag{2.76}$$

The standard way of reasoning to obtain (2.76) is to assume that (2.76) does not hold, i.e. we suppose that there exists $\varepsilon > 0$ such that

$$\|g_k\| \geqslant \varepsilon \tag{2.77}$$

for all k. Then, under this assumption we show that we can find $c > 0$ with the property

$$\cos\theta_k \geqslant c\frac{\|g_k\|}{\|d_k\|} \tag{2.78}$$

for all k. Equation (2.74) together with (2.77) and (2.78) imply that

$$\sum_{k=1}^{\infty}\frac{1}{\|d_k\|^2} < \infty. \tag{2.79}$$

Eventually, if one can show that $\|d_k\|^2$ grows at most linearly, i.e.

$$\|d_k\|^2 \leqslant Mk$$

for some constant $M < \infty$, then we come to the contradiction with (2.79).

It is interesting to observe that (2.78) is, due to the definition of θ_k, equivalent to the condition

$$g_k^T d_k \leqslant -c\|d_k\|^2 \tag{2.80}$$

for all k.

The main difficulty in proving global convergence of a conjugate gradient algorithm is to show that d_k is a direction of descent under mild assumption on the directional minimization. If we assume that the directional minimization is exact, i.e.

$$g_k^T d_{k-1} = 0 \tag{2.81}$$

for all k, then d_{k+1} is the steepest descent direction since

$$\begin{aligned} g_k^T d_k &= -\|g_k\|^2 + \beta_k g_k^T d_{k-1} \\ &= -\|g_k\|^2. \end{aligned}$$

Powell, using this property and assuming that the set (2.69) is bounded, and the function f is twice continuously differentiable, showed that the Fletcher–Reeves method generates a sequence $\{x_k\}$ satisfying (2.76).

Al-Baali showed that the descent property holds for all k if α_k is determined by using the strong Wolfe conditions. On that basis he then proved the method fulfills (2.76).

Theorem 2.8. *Assume that α_k is calculated in such a way that (2.13) is satisfied with $\eta \in (0, \frac{1}{2})$ for all k with $g_k \neq 0$. Then the descent property of the Fletcher–Reeves method holds for all k.*

Proof. Al-Baali proved by induction that the following inequalities hold for all k with $g_k \neq 0$:

$$-\sum_{j=0}^{k-1} \eta^j \leqslant \frac{g_k^T d_k}{\|g_k\|^2} \leqslant -2 + \sum_{j=0}^{k-1} \eta^j. \tag{2.82}$$

Equation (2.82) is obviously satisfied for $k = 1$, we assume that (2.82) is valid for $k > 1$ – we must show it for $k+1$. We have

$$\sum_{j=0}^{k-1} \eta^j < \sum_{j=0}^{\infty} \eta^j = \frac{1}{1-\eta},$$

thus, from (2.82), d_k is a direction of descent for $\eta \in (0, \frac{1}{2})$. Moreover, from (2.58) we also have

$$\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} = -1 + \frac{g_{k+1}^T d_k}{\|g_k\|^2}$$

hence from the strong Wolfe curvature condition

$$-1 - \eta \frac{g_k^T d_k}{\|g_k\|^2} \leqslant \frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} \leqslant -1 + \eta \frac{g_k^T d_k}{\|g_k\|^2}.$$

Applying the induction hypothesis (2.82) we eventually get

$$\begin{aligned} -\sum_{j=0}^{k} \eta^j = -1 - \eta \sum_{j=0}^{k-1} \eta^j &\leqslant \frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} \\ &\leqslant -1 + \eta \left(-2 + \sum_{j=0}^{k-1} \eta^j \right) \\ &\leqslant -2 + \sum_{j=0}^{k} \eta^j \end{aligned}$$

which completes the proof of (2.82). Since

$$-2+\sum_{j=0}^{\infty}\eta^j=-2+\frac{1}{(1-\eta)}, \tag{2.83}$$

thus, from (2.82), d_k is a direction of descent. □

The main convergence result given by Al-Baali refers to Theorem 2.1 by using the fact that d_k is a direction of descent.

Theorem 2.9. *Assume that* $\{x_k\}$ *is generated by* Algorithm 2.7 *with* β_k *defined by the Fletcher–Reeves formula (2.58). Furthermore, suppose that the assumptions (ii)–(iv) of* Theorem 2.1 *are satisfied and* α_k *is determined by the strong Wolfe conditions with* $0<\mu<\eta<\frac{1}{2}$. *Then, the sequence* $\{x_k\}$ *is such that*

$$\liminf_{k\to\infty}\|g_k\|=0. \tag{2.84}$$

Proof. Notice that

$$\begin{aligned}\|d_k\|^2 &= \|g_k\|^2-2\beta_k g_k^T d_{k-1}+\beta_k^2\|d_{k-1}\|^2\\ &\leqslant \|g_k\|^2-2\eta\beta_k g_k^T d_{k-1}+\beta_k^2\|d_{k-1}\|^2\end{aligned} \tag{2.85}$$

from the curvature condition. However, form (2.82), we also have

$$\begin{aligned}\eta g_{k-1}^T d_{k-1}\leqslant g_k^T d_{k-1} &\leqslant -\eta g_k^T d_{k-1}\\ &\leqslant \frac{\eta}{(1-\eta)}\|g_k\|^2\end{aligned}$$

which together with (2.85) imply

$$\|d_k\|^2\leqslant\frac{(1+\eta)}{(1-\eta)}\|g_k\|^2+\beta_k^2\|d_{k-1}\|^2. \tag{2.86}$$

Notice that

$$\begin{aligned}\|d_k\|^2 &\leqslant \frac{(1+\eta)}{(1-\eta)}\|g_k\|^2+\beta_k^2\left(\frac{(1+\eta)}{(1-\eta)}\|g_{k-1}\|^2+\beta_{k-1}^2\|d_{k-2}\|^2\right)\\ &= \frac{(1+\eta)}{(1-\eta)}\|g_k\|^2+\frac{(1+\eta)}{(1-\eta)}\frac{\|g_k\|^4}{\|g_{k-1}\|^4}\|g_{k-1}\|^2+\\ &\quad \beta_{k-1}^2\beta_{k-2}^2\|d_{k-2}\|^2\\ &= \frac{(1+\eta)}{(1-\eta)}\frac{\|g_k\|^4}{\|g_k\|^2}+\frac{(1+\eta)}{(1-\eta)}\frac{\|g_k\|^4}{\|g_{k-1}\|^2}+\\ &\quad \frac{\|g_k\|^4}{\|g_{k-2}\|^2}\|d_{k-2}\|^2\end{aligned} \tag{2.87}$$

which, when applied recursively, gives us

$$\|d_k\|^2 \leqslant \frac{(1+\eta)}{(1-\eta)}\|g_k\|^4 \sum_{j=1}^{k} \|g_j\|^{-2}. \tag{2.88}$$

The rest of the proof is typical. We consider θ_k, the angle between $-g_k$ and d_k. Assuming that the thesis is not valid we will show that θ_k are bounded away from $\pi/2$ and this contradicts Theorem 2.1.

Notice that from (2.82), (2.83) we have

$$\cos\theta_k \geqslant \frac{(1-2\eta)}{(1-\eta)}\frac{\|g_k\|}{\|d_k\|}. \tag{2.89}$$

Furthermore, since x_k belong to a bounded set and g is continuous, there exists a positive number c such that $c < \infty$ and

$$\|d_k\|^2 \leqslant ck \tag{2.90}$$

(from (2.88)). Assume now that the thesis is false, thus we can find a positive number c_1 such that

$$\|g_k\|^2 \geqslant c_1 \tag{2.91}$$

for all k. Taking into account (2.89)–(2.91) we come to the relation

$$\cos^2\theta_k \geqslant \frac{c_1}{ck}.$$

However, this also means that

$$\sum_{k=1}^{\infty} \cos^2\theta_k \geqslant \frac{c_1}{c}\sum_{k=1}^{\infty}\frac{1}{k} = \infty. \tag{2.92}$$

From Theorem 2.1 we know that if (2.92) holds we cannot have (2.91). This concludes the proof. □

Al-Baali introduces the condition on η under which the Fletcher–Reeves conjugate gradient algorithm is globally convergent. Assuming that $\eta \in (0, \frac{1}{2})$ imposes strict curvature condition therefore it is not surprising that there were efforts to weaken this condition. Here, we state some results given by Dai and Yuan [42] (see also [44]).

Dai and Yuan showed that the condition on η can be weakened. They propose the following curvature condition

$$\eta_1 g_k^T d_k \leqslant g(x_k + \alpha_k d_k)^T d_k \leqslant -\eta_2 g_k^T d_k \tag{2.93}$$

with $0 < \mu < \eta_1 < 1$ and $\eta_2 > 0$. If $\eta_1 = \eta_2 \in (0, \frac{1}{2})$ then we have the curvature condition considered so far. Furthermore, (2.93) is the Wolfe curvature condition if $\eta_1 = \eta \in (0,1)$ and $\eta_2 = \infty$.

Their step-length conditions are general but according to them the global convergence of the Fletcher–Reeves algorithm holds if

$$\eta_1 + \eta_2 \leqslant 1. \tag{2.94}$$

Theorem 2.10. *Assume that* $\{x_k\}$ *is generated by* Algorithm 2.7 *with* β_k *defined by the Fletcher–Reeves formula (2.58). Furthermore, suppose that the assumptions (ii)–(iv) of* Theorem 2.1 *are satisfied,* α_k *is determined by conditions (2.12), (2.93) with* $0 < \mu < \eta_1 < 1$, $\eta_2 > 0$ *and gives a direction of descent. Then, the sequence* $\{x_k\}$ *is such that*

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Proof. Due to the definition of the Fletcher–Reeves algorithm

$$-\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} = 1 - \frac{g_{k+1}^T d_k}{\|g_k\|^2} \tag{2.95}$$

which together with (2.93) leads to

$$1 - \eta_2\rho_k \leqslant \rho_{k+1} \leqslant 1 + \eta_1\rho_k, \tag{2.96}$$

where

$$\rho_k = -\frac{g_k^T d_k}{\|g_k\|^2}. \tag{2.97}$$

The proof stated in [42] (which we follow) is similar to that proposed by Al-Baali in [2]. The main ingredients of the proof are steps which show that $\|d_k\|^2$ have a linear growth rate and that it leads to the contradiction with the thesis of Theorem 2.1. In order to persuade the first step assume that

$$g_k^T d_k \leqslant 0 \tag{2.98}$$

for all k. In order to guarantee the feasibility of the step-length selection procedure we have to consider the case: $g_k^T d_k = 0$. We assume that in this case we take $\alpha_k = 0$ which implies $x_{k+1} = x_k$ and $d_{k+1} = -g_{k+1} + d_k$. Therefore, at the point x_{k+1} (2.98) holds with the strict inequality.

From the first inequality in (2.96) we deduce that

$$\eta_2\rho_k + \rho_{k+1} \geqslant 1 \tag{2.99}$$

for all k, which with the help of the Hölder inequality (cf. Appendix A) leads to

$$\rho_k^2 + \rho_{k+1}^2 \geqslant \frac{1}{1+\eta_2^2} = c_1 \tag{2.100}$$

for all k. On the other hand, the second inequality in (2.96) gives us

$$\rho_{k+1} \leqslant 1 + \eta_1 \rho_k \leqslant 1 + \eta_1 (1 + \eta_1 \rho_{k-1})$$

thus

$$\rho_k < \frac{1}{1 - \eta_1} \tag{2.101}$$

for all k.

Before proceeding further we have to show that condition (2.98) is satisfied for all k. Since for $k = 1$ the condition is obviously fulfilled we assume that it holds for all $k = 1, \ldots, \bar{k}$. If this happens then (2.96), (2.101) are true, and

$$\begin{aligned} \rho_{k+1} &\geqslant 1 - \eta_2 \rho_k > 1 - \eta_2 \frac{1}{1 - \eta_1} \\ &= \frac{1 - \eta_1 - \eta_2}{1 - \eta_1} \geqslant 0 \end{aligned}$$

which means that (2.98) also holds for $k = \bar{k} + 1$ and by induction arguments we have shown that d_k satisfy (2.98).

As in the proof of Al-Baali's theorem now we look for the upper bound on $\|d_k\|^2$. From the Fletcher–Reeves rule we have

$$\begin{aligned} \|d_{k+1}\|^2 &= \|g_{k+1}\|^2 - 2\beta_{k+1} g_{k+1}^T d_k + \beta_{k+1}^2 \|d_k\|^2 \\ &\leqslant \|g_{k+1}\|^2 + \frac{\|g_{k+1}\|^4}{\|g_k\|^4} \|d_k\|^2 + 2\eta_1 \rho_k \|g_{k+1}\|^2 \\ &\leqslant \frac{\|g_{k+1}\|^4}{\|g_k\|^4} \|d_k\|^2 + \frac{1 + 2\eta_1}{1 - \eta_1} \|g_{k+1}\|^2. \end{aligned} \tag{2.102}$$

Introducing

$$t_k = \frac{\|d_k\|^2}{\|g_k\|^4}$$

we can transform (2.102) into

$$t_{k+1} \leqslant t_k + \frac{1 + 2\eta_1}{1 - \eta_1} \frac{1}{\|g_{k+1}\|^2} \tag{2.103}$$

Now, if the thesis is not satisfied, there exists $\varepsilon > 0$ such that

$$\|g_k\|^2 \geqslant \varepsilon \tag{2.104}$$

for all k. Taking into account (2.103)–(2.104) we come to

$$t_{k+1} \leqslant t_1 + c_2 k \tag{2.105}$$

with

$$c_2 = \frac{1}{\varepsilon}\frac{1+2\eta_1}{1-\eta_1}.$$

Next the contradiction to the thesis of Theorem 2.1 has to be established. It is not so straightforward as in the proof of Al-Baali's theorem due to more general step-length conditions. We consider separately odd and even cases of k. From (2.105) we have

$$\frac{1}{t_{2k}} \geqslant \frac{1}{c_2(2k-1)+t_1}$$
$$\frac{1}{t_{2k-1}} \geqslant \frac{1}{c_2(2k-2)+t_1} > \frac{1}{c_2(2k-1)+t_1}.$$

On that basis

$$\begin{aligned}\sum_{k=1}^{\infty} cos^2\theta_k \|g_k\|^2 &= \sum_{k=1}^{\infty} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} \\ &= \sum_{k=1}^{\infty} \frac{\rho_k^2}{t_k} = \sum_{k=1}^{\infty}\left(\frac{\rho_{2k-1}^2}{t_{2k-1}} + \frac{\rho_{2k}^2}{t_{2k}}\right) \\ &\geqslant \sum_{k=1}^{\infty} \frac{\rho_{2k-1}^2+\rho_{2k}^2}{c_2(2k-1)+t_1} \\ &\geqslant \sum_{k=1}^{\infty} \frac{c_1}{c_2(2k-1)+t_1} = \infty\end{aligned}$$

which is impossible. □

If condition (2.94) is not satisfied then the Fletcher–Reeves method with line search rules (2.12) may not converge. Dai and Yuan construct the counter-example for the case

$$\eta_1 + \eta_2 > 1. \tag{2.106}$$

Suppose that the curvature condition (2.93) is satisfied with

$$g_{k+1}^T d_k = \eta_1 g_k^T d_k, \;\; k = 1,\ldots,N$$
$$g_{N+1}^T d_N = -\eta_2 g_N^T d_N.$$

Equation (2.95) implies that for the corresponding sequence $\{\rho_k\}$ the following holds

$$\rho_{k+1} = 1 + \eta_1\rho_k, \;\; k = 2,\ldots,N$$
$$\rho_{N+1} = 1 - \eta_2\rho_N.$$

By taking into account $\rho_1 = 1$ this leads to

$$\rho_k = \frac{1-\eta_1^k}{1-\eta_1}, \quad k = 2,\ldots,N \tag{2.107}$$

$$\rho_{N+1} = 1 - \eta_2 \frac{1-\eta_1^N}{1-\eta_1} = \frac{1-\eta_1-\eta_2+\eta_2\eta_1^N}{1-\eta_1}. \tag{2.108}$$

Since we have assumed (2.106) we can choose N in such a way that

$$1 + \eta_2\eta_1^N < \eta_1 + \eta_2 \tag{2.109}$$

which together with (2.108) lead to the conclusion that $\rho_{N+1} < 0$ which, from (2.97), means that d_{N+1} is not a direction of descent. Therefore, we have shown that under (2.106) applying step-length selection procedure based on (2.12), (2.93) does not prevent us from increasing value of f.

The example is based on the function

$$f(x) = \frac{1}{2}x^2 \tag{2.110}$$

where $x \in \mathscr{R}$. We claim that it is possible to find step-sizes α_k which satisfy (2.12), (2.93), they are generated by the Fletcher–Reeves algorithm and lead to the sequence

$$x_k = \eta_1^k x_1, \quad k = 1,\ldots,N, \tag{2.111}$$

$$x_{N+1} = -\eta_2\eta_1^N x_1. \tag{2.112}$$

If $\{x_k\}$ is defined by (2.111)–(2.112) and f by (2.110) then

$$d_k = -\frac{1-\eta_1^k}{1-\eta_1} g_k, \quad k = 1,\ldots,N. \tag{2.113}$$

Equation (2.113) is obviously true for $k = 1$. Now assume that it holds for $k > 1$, we show that it is valid also for $k+1$. Indeed,

$$\begin{aligned} d_{k+1} &= -g_{k+1} + \eta_1^2 d_k = -g_{k+1} - \eta_1^2 \frac{1-\eta_1^k}{1-\eta_1} g_k \\ &= -g_{k+1} - \eta_1 \frac{1-\eta_1^k}{1-\eta_1} g_{k+1} \\ &= -\frac{1-\eta_1^{k+1}}{1-\eta_1} g_{k+1}. \end{aligned}$$

It remains to show that $\{x_k\}$ is generated in such a way that conditions (2.12), (2.93) are satisfied. First, we verify condition (2.12). Due to the fact that $g_k = x_k$ we can write

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= \frac{1}{2}(x_{k+1} - x_k)(x_{k+1} - x_k) \\ &= \frac{1}{2}\frac{x_k + x_{k+1}}{g_k} \alpha_k d_k g_k \end{aligned}$$

$$= \begin{cases} \frac{1}{2}(1+\eta_1)\alpha_k d_k g_k & \text{if } k = 1,\ldots,N-1 \\ \frac{1}{2}(1-\eta_2)\alpha_k d_k g_k & \text{if } k = N \end{cases}$$
$$\leqslant \frac{1}{2}(1-\eta_2)\alpha_k d_k g_k \leqslant \mu\,\alpha_k d_k g_k.$$

Furthermore, from (2.111)–(2.113) we have

$$g_{k+1}d_k = \eta_1 d_k g_k, \quad k = 1,\ldots,N-1$$
$$g_{k+1}d_k = -\eta_2 d_k g_k.$$

thus conditions (2.93) are also fulfilled. Therefore, from (2.108)–(2.109) we conclude that d_{N+1} is a direction of ascent.

2.8 Other Versions of the Fletcher–Reeves Algorithm

The Fletcher–Reeves algorithm has one of the strongest convergence properties among all standard conjugate gradient algorithms. Yet it requires the strong Wolfe conditions for the directional minimization. The question is whether we can provide a version of the Fletcher–Reeves algorithm which is globally convergent under the Wolfe conditions. In some sense such a method has been proposed by Dai and Yuan [43].

First, Dai and Yuan look at the Fletcher–Reeves and Hestenes–Stiefel versions of conjugate gradient algorithms defined by the coefficients β_{k+1}:

$$\beta_{k+1}^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}$$
$$\beta_{k+1}^{HS} = \frac{g_k^T y_k}{d_k^T y_k}$$

where $y_k = g_{k+1} - g_k$. Observe that if the directional minimization is exact and f is a quadratic function then these two formulae lead to the same iterative algorithm.

Now, consider β_{k+1} defined by

$$\beta_{k+1}^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T y_k} \tag{2.114}$$

as proposed by Dai and Yuan. It is also equivalent to β_{k+1}^{FR} and β_{k+1}^{HS} if f is quadratic. However, notice that if the directional minimization is exact then $\beta_{k+1}^{FR} = \beta_{k+1}^{DY}$ for all k since in this case $d_k^T(g_{k+1} - g_k) = \|g_k\|^2$. Therefore, the method discussed can be regarded as a version of the Fletcher–Reeves algorithm.

Before stating the main convergence result of this section notice that β_{k+1}^{YD} can be expressed differently:

$$\beta_{k+1}^{YD} = \frac{g_{k+1}^T d_{k+1}}{g_k^T d_k}, \tag{2.115}$$

which follows from the relation

$$\begin{aligned} g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 + \beta_{k+1}^{YD} g_{k+1}^T d_k \\ &= \beta_{k+1}^{YD} \left(-y_k^T d_k + g_{k+1}^T d_k\right) \\ &= \beta_{k+1}^{YD} g_k^T d_k. \end{aligned} \tag{2.116}$$

Theorem 2.11. *Suppose that the assumptions (ii)–(iv) of* Theorem 2.1 *are satisfied and the step-length α_k is determined by the Wolfe conditions with $0 < \mu < \eta < 1$. Then,* Algorithm 2.7 *with β_k defined by (2.114) generates the sequence $\{x_k\}$ such that*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{2.117}$$

Proof. Assume that (2.117) is not satisfied, thus there exists $\varepsilon > 0$ such that

$$\|g_k\| \geqslant \varepsilon \tag{2.118}$$

for all k.

First, we show that d_k is a direction of descent. Since α_k satisfies the Wolfe curvature condition we have

$$\left(g_{k+1} - g_k\right)^T d_k \geqslant (\eta - 1) g_k^T d_k.$$

Because

$$g_k^T d_k < 0 \tag{2.119}$$

holds for $k = 1$, assuming that (2.119) is true for $k > 1$ we have to show that it is also satisfied for $k + 1$. From (2.116) and (2.119) we have that d_{k+1} is also a direction of descent.

In order to complete the proof we aim at showing that (2.118) leads to the contradiction with Theorem 2.1. Indeed, the rule for d_{k+1} can be written as

$$d_{k+1} + g_{k+1} = \beta_{k+1}^{DY} d_k.$$

Squaring it gives us

$$\|d_{k+1}\|^2 = \left(\beta_{k+1}^{DY}\right)^2 \|d_k\|^2 - 2 g_{k+1}^T d_{k+1} - \|g_{k+1}\|^2. \tag{2.120}$$

This result is then used to show that

$$\sum_{k=0}^{\infty} \frac{\left(g_{k+1}^T d_{k+1}\right)^2}{\|d_{k+1}\|^2} = \infty$$

which contradicts the thesis of Theorem 2.1.

Due to (2.115) and (2.120) we have

$$\begin{aligned}\frac{\|d_{k+1}\|^2}{\left(g_{k+1}^T d_{k+1}\right)^2} &= \frac{\|d_k\|^2}{\left(g_k^T d_k\right)^2} - \frac{2}{g_{k+1}^T d_{k+1}} \\ &\quad - \frac{\|g_{k+1}\|^2}{\left(g_{k+1}^T d_{k+1}\right)^2} \\ &= \frac{\|d_k\|^2}{\left(g_k^T d_k\right)^2} - \left(\frac{1}{\|g_{k+1}\|} + \frac{\|g_{k+1}\|}{g_{k+1}^T d_{k+1}}\right)^2 \\ &\quad + \frac{1}{\|g_{k+1}\|^2} \\ &\leqslant \frac{\|d_k\|^2}{\left(g_k^T d_k\right)^2} + \frac{1}{\|g_{k+1}\|^2}. \end{aligned} \tag{2.121}$$

From (2.121) follows immediately that

$$\frac{\|d_k\|^2}{\left(g_k^T d_k\right)^2} \leqslant \sum_{i=1}^{k} \frac{1}{\|g_k\|^2}$$

and, since (2.118) holds,

$$\frac{\|d_k\|^2}{\left(g_k^T d_k\right)^2} \leqslant \frac{k}{\varepsilon^2}$$

which leads to

$$\sum_{k=1}^{\infty} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} = \infty.$$

Due to Theorem 2.1 this completes the proof. □

Eventually, we have provided a version of the Fletcher–Reeves algorithm which is globally convergent under the Wolfe conditions in the directional minimization and without assuming that $\eta < \frac{1}{2}$.

2.9 Global Convergence of the Polak–Ribière Algorithm

The work of Al-Baali inspired other researches to provide new versions of conjugate gradient algorithms. Gilbert and Nocedal [78] consider a conjugate gradient algorithm with coefficients β_k defined by

$$|\beta_k| \leqslant \beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}. \tag{2.122}$$

The example of the sequence (2.122) is the following one

$$\beta_k = \begin{cases} -\beta_k^{FR} & \text{if } \beta_k^{PR} < -\beta_k^{FR} \\ \beta_k^{PR} & \text{if } |\beta_k^{PR}| \leqslant \beta_k^{FR} \\ \beta_k^{FR} & \text{if } \beta_k^{PR} > \beta_k^{FR} \end{cases}$$

where

$$\beta_k^{PR} = \frac{(g_k - g_{k-1})^T g_k}{\|g_{k-1}\|^2}. \tag{2.123}$$

Observe that β_k defined in that way leads to a conjugate gradient algorithm which does not have the drawback of the Fletcher–Reeves algorithm mentioned in Sect. 2.6. If $x_{k+1} \approx x_k$ then $\beta_k \approx 0$ instead of being close to one.

The global convergence of the conjugate gradient algorithm based on (2.123) follows from the familiar Al-Baali's analysis. First, we have to show that d_k is always a direction of descent if the strong Wolfe conditions are applied with $\eta < \frac{1}{2}$.

Lemma 2.6. *Suppose that the assumptions (ii)–(iv) of* Theorem 2.1 *hold and the step-length α_k satisfies the strong Wolfe conditions with $0 < \eta < \frac{1}{2}$ Then,* Algorithm 2.7 *with β_k defined by (2.122) generates directions d_k with the property*

$$-\frac{1}{1-\eta} \leqslant \frac{g_k^T d_k}{\|g_k\|^2} \leqslant \frac{2\eta - 1}{1-\eta} \tag{2.124}$$

for all k.

Proof. Equation (2.124) obviously holds for $k = 1$. Assume now that (2.124) is true for some $k > 1$. In order to apply the induction arguments we have to show that it also holds for $k + 1$.

First, observe that since $0 < \eta < \frac{1}{2}$ we have

$$g_k^T d_k < 0.$$

Furthermore, the following holds

$$\begin{aligned} \frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} &= -1 + \beta_{k+1} \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2} \\ &= -1 + \frac{\beta_{k+1}}{\beta_{k+1}^{FR}} \frac{g_{k+1}^T d_k}{\|g_k)\|^2}. \end{aligned} \tag{2.125}$$

From the line search conditions we come to

$$|\beta_{k+1} g_{k+1}^T d_k| \leqslant -\eta |\beta_{k+1}| g_k^T d_k$$

which combined with (2.125) imply

$$-1+\eta\frac{|\beta_{k+1}|}{\beta_{k+1}^{FR}}\frac{g_k^T d_k}{\|d_k\|^2}\leqslant\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2}$$
$$\leqslant -1-\eta\frac{|\beta_{k+1}|}{\beta_{k+1}^{FR}}\frac{g_k^T d_k}{\|g_k\|^2}.$$

The induction hypothesis leads to

$$-1+\frac{|\beta_{k+1}|}{\beta_{k+1}^{FR}}\frac{\eta}{1-\eta}\leqslant\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2}$$
$$\leqslant -1-\frac{|\beta_{k+1}|}{\beta_{k+1}^{FR}}\frac{\eta}{1-\eta}$$

which, due to (2.122), proves (2.124) for $k+1$. □

Lemma 2.6 shows that (2.78) holds with $c=(1-2\eta)/(1-\eta)$ and that

$$c_1\frac{\|g_k\|}{\|d_k\|}\leqslant\cos\theta_k\leqslant c_2\frac{\|g_k\|}{\|d_k\|} \tag{2.126}$$

for all k with

$$c_1=\frac{1-2\eta}{1-\eta},\quad c_2=\frac{1}{1-\eta}. \tag{2.127}$$

Equation (2.127) is essential in proving the global convergence of conjugate gradient algorithm if we can show that $\|d_k\|^2$ grows at most linearly when the gradients g_k are not too small.

Theorem 2.12. *Suppose that assumptions of* Lemma 2.6 *are fulfilled. then*

$$\liminf_{k\to\infty}\|g_k\|=0.$$

Proof. From Lemma 2.6 and the strong Wolfe curvature condition we deduce that

$$|g_k^T d_{k-1}|\leqslant-\eta g_{k-1}^T d_{k-1}\leqslant\frac{\eta}{1-\eta}\|g_{k-1}\|^2.$$

As in the proof of Theorem 2.9 we can show that (cf. (2.86))

$$\|d_k\|^2\leqslant\left(\frac{1+\eta}{1-\eta}\right)\|g_k\|^2+\beta_k^2\|d_{k-1}\|^2.$$

Using this, as in (2.87)–(2.88), we come to the relation

$$\|d_k\|^2 \leqslant \gamma \|g_k\|^4 \sum_{j=1}^{k} \|g_j\|^{-2}$$

with $\gamma = (1+\eta)/(1-\eta)$. If we assume now that there exists $\varepsilon > 0$ such that

$$\|g_k\| \geqslant \varepsilon$$

for all k, then

$$\|d_k\|^2 \leqslant \gamma \varepsilon^2 k. \tag{2.128}$$

The rest of the proof follows the appropriate lines of the proof of Theorem 2.9. □

2.10 Hestenes–Stiefel Versions of the Standard Conjugate Gradient Algorithm

Another globally convergent algorithm was proposed in [91]. The direction d_k is determined according to the formula:

$$d_{k+1} = -g_{k+1} + \beta_{k+1}^{HZ} d_k \tag{2.129}$$

$$\beta_{k+1}^{HZ} = \frac{1}{d_k^T y_k} \left(y_k - 2d_k \frac{\|y_k\|^2}{d_k^T y_k} \right)^T g_{k+1}. \tag{2.130}$$

The formula is related to the Hestenes–Stiefel version of the standard conjugate gradient algorithm. Assume that the line search is exact. Then we have $d_k^T g_{k+1} = 0$ and β_{k+1}^{HZ} becomes

$$\beta_{k+1}^{HZ} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$$

which is the coefficient in the Hestenes–Stiefel formula (cf. (2.60)).

The formula (2.130) is also related to the formerly proposed scheme by Dai and Liao [41]:

$$\beta_{k+1}^{DL} = \frac{1}{d_k^T y_k} (y_k - t s_k)^T g_{k+1} \tag{2.131}$$

with $t > 0$ as a constant parameter. Equation (2.130) can be regarded as an adaptive version of (2.131) with $t = 2\|y_k\|^2 / s_k^T d_k$.

More interesting is that the conjugate gradient algorithm defined by (2.129)–(2.130) can be treated as a version of the Perry–Shanno memoryless quasi-Newton

method as discussed in the next chapter (cf. (3.19)). If d_{k+1}^{PS} is a direction according to the Perry–Shanno formula and d_{k+1}^{HZ} is given by (2.129)–(2.130), then

$$\begin{aligned} d_{k+1}^{PS} &= -\gamma_k g_{k+1} - \left[\left(1+\gamma_k\frac{\|y_k\|^2}{d_k^T y_k}\right)\frac{d_k^T g_{k+1}}{d_k^T y_k} - \gamma_k\frac{y_k^T g_{k+1}}{d_k^T y_k}\right]^T d_k \\ &\quad +\gamma_k\frac{d_k^T g_{k+1}}{d_k^T y_k}y_k \\ \gamma_k &= \frac{d_k^T y_k}{\|y_k\|^2} \end{aligned}$$

and

$$d_{k+1}^{HZ} = \gamma_k\left(d_{k+1}^{PS} + \frac{d_k^T g_{k+1}}{d_k^T y_k}y_k\right). \tag{2.132}$$

Indeed we have to show that

$$\begin{aligned} &-\gamma_k g_{k+1} + \gamma_k\left[\frac{1}{d_k^T y_k}\left(y_k - 2d_k\frac{1}{\gamma_k}\right)^T g_{k+1}\right]^T d_k + \gamma_k\frac{d_k^T g_{k+1}}{d_k^T y_k} \\ &= -\gamma_k g_{k+1} - \left[\left(1+\gamma_k\frac{1}{\gamma_k}\right)\frac{d_k^T g_{k+1}}{d_k^T y_k} - \gamma_k\frac{y_k^T g_{k+1}}{d_k^T y_k}\right]^T d_k + \gamma_k\frac{d_k^T g_{k+1}}{p_k^T y_k} \end{aligned}$$

which obviously holds.

Thus, if the second term in the bracket of (2.132) can be neglected then the direction d_k^{HZ} is a multiple of d_k^{PS}. Hager and Zhang show in [91] that this is the case if f is strongly convex and the cosine of the angle between g_{k+1} and d_k is small.

The direction d_{k+1} is a direction of descent if instead of β_{k+1}^{HZ} in (2.129) we use $\hat{\beta}_{k+1}^{HZ}$ that avoids low nonnegative values of β_k^{HZ} – the new algorithm is as follows:

$$d_{k+1} = -g_{k+1} + \hat{\beta}_{k+1}^{HZ} d_k \tag{2.133}$$

$$\hat{\beta}_{k+1}^{HZ} = \max\left[\beta_{k+1}^{HZ}, \eta_k\right], \quad \eta_k = \frac{-1}{\|d_k\|\min\left[\eta, \|g_k\|\right]} \tag{2.134}$$

where $\eta > 0$. Notice that when $\{d_k\}$ is bounded and $\|g_k\|$ assume small values then η_k have big negative values and we regard $\hat{\beta}_{k+1}^{HZ}$ and β_{k+1}^{HZ} as equal.

The introduction of $\hat{\beta}_{k+1}^{HZ}$ is motivated in [91] by the need of having the scheme which guarantees global convergence properties of algorithm (2.130), (2.133)–(2.134) with the Wolfe line search rules. The convergence analysis given in [91] breaks down when β_{k+1}^{HZ} assume negative values. We could overcome the problem by taking

$$\hat{\beta}_{k+1}^{HZ} = \max[\beta_{k+1}^{HZ}, 0]$$

however that would impair the efficiency of the method. The other formula which comes to mind is

$$\hat{\beta}_{k+1}^{HZ} = |\beta_{k+1}^{HZ}| \tag{2.135}$$

but in this case the essential result in [91]–Theorem 2.13 stated below couldn't be proved. Notice that this choice for coefficients β_k is natural since coefficients β_k are always positive when f is quadratic and nonlinear conjugate gradient algorithms rely on the quadratic approximation of f near a solution. The presentation of the Hager–Zhang algorithm is based on the formula (2.134).

The global convergence properties of algorithm (2.130), (2.133)–(2.134) refer to the important property of d_k being always a direction of descent.

Theorem 2.13. *Suppose that* $d_k^T y_k \neq 0$ *and*

$$d_{k+1} = -g_{k+1} + \tau d_k, \;\; d_1 = -g_1,$$

for any $\tau \in \left[\beta_{k+1}^{HZ}, \max\left[\beta_{k+1}^{HZ}, 0\right]\right]$, *then*

$$g_{k+1}^T d_{k+1} \leqslant -\frac{7}{8}\|g_{k+1}\|^2. \tag{2.136}$$

Proof. We follow the proof stated in [91]. Suppose first that $\tau = \beta_{k+1}^{HZ}$. Since for $k = 1$ (2.136) is obviously satisfied, assume that for $k > 1$. Then, we have

$$\begin{aligned} g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 + \beta_{k+1}^{HZ} g_{k+1}^T d_k \\ &= -\|g_{k+1}\|^2 + g_{k+1}^T d_k \left(\frac{y_k^T g_{k+1}}{d_k^T y_k} - 2\frac{\|y_k\|^2 g_{k+1}^T d_k}{\left(d_k^T y_k\right)^2} \right) \\ &= \frac{y_k^T g_{k+1} \left(d_k^T y_k\right) \left(g_{k+1}^T d_k\right) - \|g_{k+1}\|^2 \left(d_k^T y_k\right)^2 - 2\|y_k\|^2 \left(g_{k+1}^T d_k\right)^2}{\left(d_k^T y_k\right)^2} \\ &= \frac{u^T v - 4\|u\|^2 - \frac{1}{2}\|v\|^2}{\left(d_k^T y_k\right)^2} \end{aligned}$$

where

$$u = \frac{1}{2}\left(d_k^T y_k\right) g_{k+1}, \;\; v = 2\left(g_{k+1}^T d_k\right) y_k. \tag{2.137}$$

Taking into account the inequality

$$u^T v \leqslant \frac{1}{2}\left(\|u\|^2 + \|v\|^2\right),$$

from (2.137) we obtain

$$g_{k+1}^T d_{k+1} \leqslant -\frac{7}{2}\frac{\|u\|^2}{\left(d_k^T y_k\right)^2} = -\frac{7}{8}\|g_{k+1}\|^2.$$

The case when $\tau \neq \beta_{k+1}^{HZ}$ and $\beta_{k+1}^{HZ} \leqslant \tau < 0$ can be proved by noting that

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 + \tau g_{k+1}^T d_k \leqslant -\|g_{k+1}\|^2 + \beta_{k+1}^{HZ} g_{k+1}^T d_k \tag{2.138}$$

provided that $g_{k+1}^T d_k < 0$. If, on the other hand $g_{k+1}^T d_k > 0$, then (2.136) follows directly from the equality in (2.138). □

Obviously, for $\tau = \beta_{k+1}^{HZ}$, d_k is a direction of descent satisfying (2.136) as is the case for the scheme defined by (2.133), (2.130) and (2.134) since

$$\hat{\beta}_{k+1}^{HZ} = \max\left[\beta_{k+1}^{HZ}, \eta_k\right] \in \left[\hat{\beta}_{k+1}^{HZ}, \max\left[\beta_{k+1}^{HZ}, 0\right]\right]$$

which follows from the fact that η_k is negative.

Hager and Zhang prove in [91] global convergence of their method in the sense of condition (2.84). The quite elaborate proof borrows from the analysis by Gilbert and Nocedal given in [78].

Theorem 2.14. *Suppose that the assumptions (i)–(ii) of* Theorem 2.7 *are satisfied. Then,* Algorithm 2.7 *with β_k defined by (2.130), (2.133)–(2.134) and with the Wolfe line search rules generates the sequence $\{x_k\}$ such that*

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Instead of providing the proof of the theorem we show, following [91], the analysis of the algorithm applied to strongly convex functions.

Theorem 2.15. *Suppose that f is strongly convex and Lipschitz continuous on the level set defined in (i) of* Theorem 2.7. *That is, there exist constants $L < \infty$ and $\nu > 0$ such that (2.26) holds and*

$$(g(x) - g(y))^T (x - y) \geqslant \nu \|x - y\| \tag{2.139}$$

for all $x, y \in \mathscr{M}$. Then for the scheme (2.129)–(2.130) and with the Wolfe line search rules

$$\lim_{k\to\infty} \|g_k\| = 0.$$

Proof. Following the proof of Theorem 2.1 we can show that

$$\alpha_k \geqslant \frac{(1-\eta)}{L} \frac{\left\|g_k^T d_k\right\|}{\|d_k\|^2} \tag{2.140}$$

(cf. (2.29)).

Then, by the strong convexity assumption we have:

$$y_k^T d_k \leqslant \nu \alpha_k \|d_k\|^2. \tag{2.141}$$

Theorem 2.13 implies that if $g_k \neq 0$ then $d_k \neq 0$, and (2.141) shows that $y_k^T d_k > 0$. From the strong convexity assumption we know that f is bounded from below. Therefore, from the first Wolfe condition (2.46) we conclude that

$$\sum_{k=1}^{\infty} \alpha_k g_k^T d_k > -\infty. \tag{2.142}$$

Now, we take into account (2.142), (2.140) and (2.136) to obtain

$$\sum_{k=1}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty. \tag{2.143}$$

In order to conclude the proof we have to show that $\|d_k\| \leqslant c\|g_k\|$ for some positive constant c. To this end notice that

$$\|y_k\| \leqslant L\alpha_k \|d_k\|$$

and thus we can write

$$\begin{aligned}
\left|\beta_{k+1}^{HZ}\right| &= \left| \frac{y_k^T g_{k+1}}{d_k^T y_k} - 2\frac{\|y_k\|^2 d_k^T g_{k+1}}{\left(d_k^T y_k\right)^2} \right| \\
&\leqslant \frac{\|y_k\|\|g_{k+1}\|}{\nu\alpha_k\|d_k\|^2} + 2\frac{\|y_k\|^2\|d_k\|\|g_{k+1}\|}{\nu^2\alpha_k^2\|d_k\|^4} \\
&\leqslant \frac{L\alpha_k\|d_k\|\|g_{k+1}\|}{\nu\alpha_k\|d_k\|^2} + 2\frac{L^2\alpha_k^2\|d_k\|^3\|g_{k+1}\|}{\nu^2\alpha_k^2\|d_k\|^4} \\
&\leqslant \left(\frac{L}{\nu} + \frac{2L^2}{\nu^2}\right)\frac{\|g_{k+1}\|}{\|d_k\|}.
\end{aligned} \tag{2.144}$$

Eventually, we have

$$\|d_{k+1}\| \leqslant \|g_{k+1}\| + \left|\beta_{k+1}^{HZ}\right| \|d_k\| \leqslant c\|g_{k+1}\|$$

with $c = 1 + L/\nu + 2L^2/\nu^2$ and from (2.143) also

$$\sum_{k=1}^{\infty} \|g_k\|^2 < \infty$$

which concludes the proof. □

Notice that Theorem 2.15 concerns the scheme defined by (2.129)–(2.130). It does not hold when we use (2.130), (2.133)–(2.134), but in that case we can refer to Theorem 2.14. Furthermore, $\hat{\beta}_{k+1}^{HZ}$ is defined in such a way that when x_k is close to a solution then η_k tends to $-\infty$ and thus $\hat{\beta}_{k+1}^{HZ} = \beta_{k+1}^{HZ}$. From the analysis given in the proof it follows also that the scheme with (2.133), (2.130) and (2.135) would be convergent for strongly convex functions, if d_{k+1} were a direction of descent.

Now we are in the position to show that if the cosine of the angle between d_k and g_{k+1} is small then the second term in the brackets in formula (2.132) can be neglected. Hager and Zhang estimate the norm of the second term in the brackets and the norm of d_{k+1} with respect to the norm of g_{k+1} They evaluate

$$\frac{|d_k^T g_{k+1}|}{|d_k^T y_k|} \leqslant c_1 \varepsilon \|g_{k+1}\| \tag{2.145}$$

$$\|d_{k+1}\| \geqslant \sqrt{1-2c_2\varepsilon}\|g_{k+1}\| \tag{2.146}$$

where c_1, c_2 are some positive numbers and ε is the cosine of the angle between d_k and g_{k+1}. As a result we have

$$\frac{|d_k^T g_{k+1}|}{|d_k^T y_k|}\frac{\|y_k\|}{\|d_{k+1}\|} \leqslant \frac{c_1\varepsilon}{\sqrt{1-2c_2\varepsilon}} \tag{2.147}$$

and if ε is small then the left-hand side in (2.147) is also small. To show (2.145)–(2.146) we refer to condition (2.139) (and to (2.141)). We have

$$\frac{|d_k^T g_{k+1}|}{|d_k^T y_k|} \leqslant \frac{L}{\nu}\left|t_k^T g_{k+1}\right| \leqslant c_1\varepsilon\|g_{k+1}\|,$$

where $t_k = d_k/\|d_k\|$. Furthermore, from (2.129),

$$\begin{aligned}\|d_{k+1}\|^2 &= \|g_{k+1}\|^2 - 2\beta_{k+1}^{HZ} d_k^T g_{k+1} + \left(\beta_k^{HZ}\right)^2 \|d_k\|^2 \\ &\geqslant \|g_{k+1}\|^2 - 2\beta_k^{HZ} d_k^T g_{k+1},\end{aligned} \tag{2.148}$$

and from (2.144) we also have

$$\left|\beta_k^{HZ} d_k^T g_{k+1}\right| \leqslant c_2\left|t_k^T g_{k+1}\right|\|g_{k+1}\| = c_2\varepsilon\|g_{k+1}\|^2. \tag{2.149}$$

Equation (2.149) combined with (2.148) leads to (2.146).

On the basis of this analysis we can conclude that the Hager–Zhang algorithm achieves stronger convergence by impairing conjugacy.

2.11 Notes

The chapter is the introduction to nonlinear conjugate gradient algorithms. Our main interests is to establish the global convergence properties of several versions of the standard conjugate gradient algorithm such as Fletcher–Reeves, Polak–Ribière, or Hestenes–Stiefel.

Although it was quickly recognized that the Polak–Ribière is the most efficient conjugate gradient algorithm much efforts have been devoted to the analysis of the Fletcher–Reeves versions. This was mostly due to the very interesting result by Al-Baali [2] which opened the possibility of analyzing the conjugate gradient

algorithms with the common feature of having $\|g_{k+1}\|^2$ in the formula for β_{k+1}. We provide original prove of the Al-Baali convergence theorem. The proofs of the related convergence results in Sects. 8–9 follow the same pattern and their ingenuities lie in constructing the proper sequences which, under the assumption that a method is not convergent, lead to the contradiction with the thesis of Theorem 2.1. While reporting these results we refer to ingenuities of the original proofs.

Some other schemes for updating β_{k+1} along the Fletcher–Reeves scheme have been proposed, for example Fletcher considered the method, which he called *conjugate descent* which is based on the formula [71]

$$\beta_{k+1}^{CD} = \frac{\|g_{k+1}\|^2}{-d_k^T g_k}.$$

In fact, if we look for a conjugate gradient algorithm which is derived directly from its quadratic counterpart we have two choices for the numerator in β_{k+1}: $\|g_{k+1}\|^2$ and $g_{k+1}^T y_k$, and three choices for the denominator: $\|g_k\|^2$, $-d_k^T g_k$ and $d_k^T y_k$ [92].

Consider now the literature on the convergence of the Polak–Ribière versions of a conjugate gradient algorithm. In that case we have the negative result by Powell [169] who showed, using 3 dimensional example, that even with exact line search, the Polak–Ribière algorithm can cycle indefinitely without converging to a stationary point (see also [40]). As it was shown earlier in [168] the crucial condition guaranteeing the convergence (under the exact line search assumption) is

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0.$$

This condition (often erroneously stated that step-sizes go to zero), which is difficult to ensure if the Wolfe conditions are applied in the line search, hampered attempts to construct a globally convergent conjugate gradient algorithm with the Polak–Ribière scheme. In Sect. 9 we show in fact a hybrid method which uses the Fletcher–Reeves update scheme to ensure its convergence. Another way of improving the global convergence properties of the Polak–Ribière versions is to employ different line search rules. Such an approach is followed in [88] (see also [89]) where a new Armijo rule is proposed – α_k is defined as

$$\alpha_k = \max\left[\beta^j \frac{\tau\,|d_k^T g_k|}{\|d_k\|^2}\right]$$

with $j \geqslant 0$ being the smallest integer number which leads to

$$f(x_{k+1}) - f(x_k) \leqslant -\mu\,\alpha_k \|d_k\|^2$$
$$-\eta_1 \|g_{k+1}\|^2 \leqslant g_{k+1}^T d_k \leqslant \eta_2 \|g_{k+1}\|^2$$

where $\beta,\ \mu \in (0,1)$ and $0 < \eta_2 < 1 < \eta_1$. In another venue of the same spirit they analyzed using trust region approach combined with the Polak–Ribière version of a conjugate gradient method [81].

As far as the other possibilities are concerned among all 6 combinations of numerators and denominators in coefficients β_{k+1} we have to refer to the method analyzed by Liu and Storey in [124] and based on the updating rule

$$\beta_{k+1}^{LS} = \frac{g_{k+1}^T y_k}{-d_k^T g_k}.$$

Furthermore, we need to stress that the method discussed in Sect. 10 is in fact the Hestenes–Stiefel conjugate gradient algorithm with

$$\beta_{k+1}^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$$

provided that directional minimization is exact. The analysis of the method is provided closely following [91].

We end up the review of the standard nonlinear conjugate gradient algorithms by noting that the presented versions can be combined. Nazareth [143] introduces a two-parameter family of conjugate gradient methods:

$$\beta_{k+1} = \frac{\lambda_k \|g_{k+1}\|^2 + (1-\lambda_k) g_{k+1}^T y_k}{\tau_k \|g_k\|^2 + (1-\tau_k) d_k^T y_k}$$

where $\lambda_k, \tau_k \in [0,1]$. In [45] an even wider, a three parameter family is considered.

All versions of the conjugate gradient algorithm considered in this chapter we call standard ones. They are derived from Algorithm 1.4 presented in Chap. 1. It seems natural to look for extensions of the general conjugate gradient algorithm formulated as Algorithm 1.5. Surprisingly, that approach was first applied in nondifferentiable optimization so we postpone the presentation of the extension of the method of shortest residuals to Chap. 5 where the basic concepts of nondifferentiable optimization are also introduced.

Chapter 3
Memoryless Quasi-Newton Methods

3.1 Introduction

In Sect. 2.10 we have presented the conjugate gradient algorithm derived from the Hestenes–Stiefel method. The method is globally convergent under the Wolfe line search rules. Its strong convergent properties are achieved by modifying the coefficient β_k in such a way that it is no longer equivalent to the nonlinear Hestenes–Stiefel conjugate gradient algorithm. In Sect. 3.2 we show that the nonlinear Polak–Ribière method is equivalent to the nonlinear Hestenes–Stiefel algorithm provided that the directional minimization is exact. Having that in mind and the fact that Hager and Zhang do not stipulate condition (2.68) in Theorem 2.14 their main convergence result is remarkable.

Hager–Zhang formula for d_k is related to the formula used in the method proposed by Shanno in [191] and [192]. Shanno came to his method by looking for a scheme for β_k which would enable to state the equation for d_k as in quasi-Newton algorithms. We recall that the quasi-Newton formula is as follows

$$B_{k+1}d_{k+1} = -g_{k+1}. \tag{3.1}$$

Suppose that $m_{k+1}(\cdot)$ is a quadratic approximation of f around the point x_{k+1}:

$$m_{k+1}(z) = f(x_{k+1}) + g_{k+1}^T z + \frac{1}{2} z^T B_{k+1} z$$

with $z \in \mathscr{R}^n$. As far as B_{k+1} is concerned it should be the Hessian matrix of f evaluated at x_{k+1} if the approximation is of the second order. Assuming that Hessian matrix evaluation is expensive we require that m_{k+1} has gradients which match those of f at the points x_k and x_{k+1}. Since $\nabla m_{k+1}(0) = g_{k+1}$ and $x_{k+1} = x_k + \alpha_k d_k$ the second condition is stated as

$$\nabla m_{k+1}(-\alpha_k d_k) = g_{k+1} - \alpha_k B_{k+1} d_k = g_k$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

which implies that B_{k+1} must satisfy

$$B_{k+1}s_k = y_k \tag{3.2}$$

where

$$\begin{aligned} s_k &= x_{k+1} - x_k \\ y_k &= g_{k+1} - g_k. \end{aligned}$$

(We remind that (3.2) is called the secant equation)

In addition to (3.2) we require that B_{k+1} is positive definite. Equation (3.2) imposes n conditions on B_{k+1} and n additional conditions follow from the requirement that B_{k+1} is positive definite – all principal minors have this property. Eventually the secant equation (3.2) have infinitely many solutions since B_{k+1} is fully defined by specifying $(n-1)n/2$ elements. It is not our goal to discuss in detail different formulae for B_{k+1} which satisfy (3.2). We only state the most popular ones:

$$B_{k+1} = \left(I - \gamma_k y_k s_k^T\right) B_k \left(I - \gamma_k s_k y_k^T\right) + \gamma_k y_k y_k^T \tag{3.3}$$

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{3.4}$$

with

$$\gamma_k = \frac{1}{y_k^T s_k}. \tag{3.5}$$

Equation (3.3) is called Davidon–Fletcher–Powell (DFP) formula while (3.4) Broyden–Fletcher–Goldfarb–Shanno formula. Both (3.3) and (3.4) are members of the Broyden class described by the formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k \left(s_k^T B_k s_k\right) v_k v_k^T \tag{3.6}$$

where ϕ_k is a positive parameter and

$$v_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}. \tag{3.7}$$

Notice that we recover the DFP formula by setting $\phi_k = 0$ and the BFGS formula assuming $\phi_k = 1$ in (3.7).

The major drawback of a quasi-Newton method described by (3.1) is that we have to solve a set of linear equations in order to calculate d_k. Fortunately, there exist formulae similar to (3.6)–(3.7) that refer to the inverse of the matrix B_{k+1}. These formulae can be derived by applying the Sherman–Morrison–Woodbury rule (cf. (B.17) in Appendix B). If $k = 1$ in (B.17) then the Sherman–Morrison–Woodbury formula can be used to obtain the inverse of a rank-one update of A using A^{-1}.

Suppose that

$$\hat{A} = A + ab^T, \quad a, b \in \mathscr{R}^n,$$

then

$$\hat{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1 + b^TA^{-1}a}.$$

Using the lemma we can express formula for $H_{k+1} = B_{k+1}^{-1}$:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k} + \phi_k \left(y_k^T H_k y_k\right) v_k v_k^T \tag{3.8}$$

with

$$v_k = \frac{s_k}{y_k^T s_k} - \frac{H_k y_k}{y_k^T H_k y_k} \tag{3.9}$$

and $\phi_k \in [0,1]$. Putting $\phi_k = 0$ in (3.8)–(3.9) gives us the DFP update to the approximation of the inverse of the Hessian matrix, while $\phi_k = 1$ leads to the BFGS formula. Since H_{k+1} gives the approximation to the inverse of the Hessian matrix it satisfies the following secant equation

$$H_{k+1} y_k = s_k. \tag{3.10}$$

Suppose that α_k satisfies the Wolfe curvature condition

$$g_{k+1}^T d_k \geqslant \eta g_k^T d_k \geqslant g_k^T d_k,$$

since $\eta \in (0,1)$. Because d_k is a direction of descent, we also have

$$y_k^T s_k > 0 \tag{3.11}$$

and as we have shown in Sect. 1.7 H_{k+1} is positive definite provided that H_k has that property.

The Broyden update formula (3.6) contains three rank-one corrections to the matrix B_k. The third correcting term decreases the eigenvalues of the matrix – see [146] for details, If ϕ_k is negative, B_{k+1} becomes singular when

$$\phi_k^c = \frac{1}{1 - \nu_k}$$

with

$$\nu_k = \frac{\left(y_k^T B_k^{-1} y_k\right)\left(s_k^T B_k s_k\right)}{\left(y_k^T s_k\right)^2}.$$

Using the Hölder inequality one can show that $\nu_k \geqslant 1$ and that implies that $\phi_k^c \leqslant 0$. In conclusion, if B_1 is positive definite and $y_k^T s_k > 0$ then B_k remain positive definite provided that $\phi_k > \phi_k^c$.

Remarkable property of the Broyden update with $\phi_k > \phi_k^c$ is that it leads to the same points x_k under the assumption that the directional minimization is exact. This holds even when applied to nonlinear functions. Therefore, quasi-Newton methods with different B_k, but from the Broyden class, have directions d_k which differ only by their lengths and their step lengths α_k are different due to scaling factors in one-dimensional minimization along d_k.

If the function f is quadratic one can prove additional properties of quasi-Newton methods from the Broyden class. They are stated in Theorem 1.18 – directions $\{d_k\}$ are conjugate and H_n is the inverse of the Hessian matrix defining f.

In the subsequent sections we are mainly interested in quasi-Newton methods with limited memory, i.e. when B_{k+1} is the result of m quasi-Newton updates applied to a diagonal matrix using vectors $y_k, \ldots, y_{k-m}$ and $s_k, \ldots, s_{k-m}$. The special case $m = 0$ leads to the so-called memoryless quasi-Newton method.

The methods from the Broyden class have the following update formula for $m = 0$

$$B_{k+1} = I - \frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k \left(s_k^T s_k\right) v_k v_k^T$$

with

$$v_k = \frac{y_k}{y_k^T s_k} - \frac{s_k}{s_k^T s_k}.$$

3.2 Conjugate Gradient Algorithm as the Memoryless Quasi-Newton Method

The equivalence of the memoryless quasi-Newton algorithm to the conjugate gradient algorithm was established by Shanno. The starting point for Shanno is the Hestenes–Stiefel algorithm whose formula for β_{k+1} is as follows

$$\beta_{k+1} = \frac{g_{k+1}^T y_k}{y_k^T d_k}.$$

The Hestenes–Stiefel formula for β_{k+1} is, under the assumption of the exact directional minimization, equivalent to the Polak–Ribière one. In order to show that, notice that if

$$g_{k+1}^T d_k = 0, \tag{3.12}$$

then

$$\begin{aligned} y_k^T d_k &= \left(g_{k+1} - g_k\right)^T d_k \\ &= -g_k^T d_k = \|g_k\|^2 \end{aligned}$$

since, under (3.12),

$$\begin{aligned} g_k^T d_k &= -\|g_k\|^2 + \beta_k g_k^T d_{k-1} \\ &= -\|g_k\|^2. \end{aligned}$$

The direction in the Hestenes–Stiefel algorithm can be expressed in the form

$$\begin{aligned} d_{k+1} &= -g_{k+1} + \beta_{k+1} d_k = -g_{k+1} + \frac{g_{k+1}^T y_k}{y_k^T d_k} d_k \\ &= -\left(I - \frac{d_k y_k^T}{y_k^T d_k} \right) g_{k+1}. \end{aligned} \tag{3.13}$$

Equation (3.13) can be further developed, under assumption (3.12), into

$$\begin{aligned} d_{k+1} &= -\left(I - \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k} \right) g_{k+1} \\ &= -Q_{k+1} g_{k+1} \end{aligned} \tag{3.14}$$

with $s_k = \alpha_k d_k$.

Equation (3.14) has been proposed by Perry [158] who has also noticed that Q_{k+1} fulfills the equation similar (but not identical) to the secant equation (3.10):

$$y_k^T Q_{k+1} = p_k.$$

Besides the fact that Q_{k+1} does not satisfy the secant equation there is another drawback of the Perry's formula – the matrix Q_{k+1} is not symmetric. However, adding one extra term transforms it into symmetric matrix

$$Q_{k+1} = I - \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k} - \frac{y_k s_k^T}{y_k^T s_k}.$$

This step advocated by Shanno [191], together with another one, lead to the matrix $Q_{k+1}^\star$ fulfilling the secant equation (3.10):

$$Q_{k+1}^\star = I - \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k}.$$

We make two important observations related to $Q_{k+1}^\star$. First, if (3.12) holds then

$$d_{k+1} = -Q_{k+1}^\star g_{k+1}$$

is the same as that defined by the Hestenes–Stiefel rule.

Secondly, applying the BFGS update to the identity matrix also leads to $Q_{k+1}^\star$. We recall here that this formula is as follows

$$H_{k+1} = I - \frac{y_k y_k^T}{y_k^T y_k} + \frac{s_k s_k^T}{s_k^T y_k} + \left(y_k^T y_k\right) v_k v_k^T$$

with

$$v_k = \frac{s_k}{s_k^T y_k} - \frac{y_k}{y_k^T y_k}.$$

Since the BFGS update formula guarantees that H_{k+1} is positive definite, provided that H_k is positive definite and the curvature condition (3.11) holds, the matrix $Q_{k+1}^{\star}$ is also positive definite.

Eventually, we have arrived at the memoryless BFGS method and one could wonder what are benefits of using $Q_{k+1}^{\star}$ instead of the matrix H_{k+1} which is derived from matrix H_k not necessarily equal to the identity matrix. The reason is that storing of $Q_{k+1}^{\star}$ is not needed as d_{k+1} is also expressed by

$$d_{k+1} = -g_{k+1} - \left[\left(1 + \frac{y_k^T y_k}{s_k^T y_k}\right)\frac{g_{k+1}^T s_k}{s_k^T y_k} - \gamma_k \frac{g_{k+1}^T y_k}{s_k^T g_k}\right] s_k + \frac{g_{k+1}^T s_k}{s_k^T y_k} y_k. \tag{3.15}$$

Notice also that in order to evaluate d_{k+1} we need to store only four vectors: d_k, d_{k+1}, g_k and g_{k+1} – the same vectors as in the Polak–Ribière algorithm.

Since the memoryless BFGS method is equivalent to a conjugate gradient algorithm natural question arises whether the same applies to other memoryless versions of methods from the Broyden family. Setting H_k to I in (3.8)–(3.9) and multiplying the result by g_{k+1} gives

$$d_{k+1} = -g_{k+1} + \left[\frac{g_{k+1}^T y_k}{y_k^T y_k} + \phi_k\left(\frac{g_{k+1}^T s_k}{s_k^T y_k} - \frac{g_{k+1}^T y_k}{y_k^T y_k}\right)\right] y_k - \left[\phi_k \frac{y_k^T y_k}{s_k^T y_k}\left(\frac{g_{k+1}^T s_k}{s_k^T y_k} - \frac{g_{k+1}^T y_k}{y_k^T y_k}\right) - \frac{g_{k+1}^T s_k}{s_k^T y_k}\right] s_k.$$

Under the exact directional minimization assumption this formula reduces to

$$d_{k+1} = -g_{k+1} + \left[(1-\phi_k)\frac{g_{k+1}^T y_k}{y_k^T y_k}\right] y_k - \phi_k \frac{g_{k+1}^T y_k}{s_k^T y_k} s_k.$$

If $\phi_k = 1$ then we have the Hestenes–Stiefel conjugate gradient algorithm, otherwise y_k essentially modifies the direction and we cannot say that we are dealing with a conjugate gradient algorithm.

Unfortunately a memoryless quasi-Newton method does not have good scaling properties. In particular, we cannot guarantee that after n iterations the matrix H_n is close to the inverse of the Hessian matrix evaluated at the point x_n. As a result of

that the step length α_k is rarely close to one even when x_k is in the neighborhood of a solution, something which is characteristic for a standard quasi-Newton method.

Therefore, Shanno proposes modification of the direction rule (3.15). He refers to self-scaling quasi-Newton methods (also known as self–scaling variable metric – SSVM methods) investigated, among others by Oren, and Spedicato and Luenberger [148], [149], [150], [151], [152]. They considered an update of H_{k+1} according to the formula

$$H_{k+1} = \left(H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \phi_k v_k v_k^T \right) \gamma_k + \frac{s_k s_k}{s_k^T y_k}, \tag{3.16}$$

where the modification relies on adding the parameter γ_k which makes the sequence $\{x_k\}$ invariant under multiplication of the objective function by a constant.

Notice that the Newton sequence is invariant under this scaling. If

$$\hat{f}(x) = cf(x),$$

then

$$\hat{g}(x) = cg(x) \text{ and } \nabla^2 \hat{f}(x) = c\nabla^2 f(x)$$

thus

$$\begin{aligned} x_{k+1} &= x_k - \left[\nabla^2 \hat{f}(x_k)\right]^{-1} \hat{g}_k = x_k - c^{-1} \left[\nabla^2 f(x_k)\right]^{-1} c g_k \\ &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} g_k. \end{aligned}$$

Oren and Spedicato [152] consider the choice of parameter γ_k which minimizes the condition number of the matrix

$$H_k^{-1} H_{k+1}$$

and shows that this is achieved by γ_k which satisfies the relation

$$\phi_k = \frac{b_k (c_k - b_k \gamma_k)}{\gamma_k \left(a_k c_k - b_k^2\right)}$$

with

$$\begin{aligned} a_k &= y_k^T H_k y_k \\ b_k &= s_k^T y_k \\ c_k &= s_k^T H_k s_k. \end{aligned}$$

If we assume that $\phi_k = 1$ then the above relations gives the formula for γ_k

$$\gamma_k = \frac{s_k^T y_k}{y_k^T H_k y_k}. \tag{3.17}$$

The update defined by (3.16) and (3.17) was tried on several problems with rather disappointing results [193]. However, it was observed that using it in the update of the initial matrix H_1 provided marked improvement over unscaled quasi-Newton methods. Based on these observations Shanno proposes the scaled memoryless BFGS method

$$d_{k+1} = -\gamma_k g_{k+1} - \left[\left(1+\gamma_k\frac{y_k^T y_k}{s_k^T y_k}\right)\frac{g_{k+1}^T s_k}{s_k^T y_k} - \gamma_k\frac{g_{k+1}^T y_k}{s_k^T y_k}\right]s_k + \gamma_k\frac{g_{k+1}^T s_k}{s_k^T y_k}y_k, \tag{3.18}$$

$$\gamma_k = \frac{s_k^T y_k}{y_k^T y_k}. \tag{3.19}$$

Notice that if $g_{k+1}^T d_k = 0$ then (3.18) reduces to

$$d_{k+1} = \gamma_k\left(-g_{k+1} + \frac{g_{k+1}^T y_k}{s_k^T y_k}p_k\right).$$

Observe that the scaling given above is in fact changing of the direction length and can be easily accomplished in the step-length selection procedures discussed in Chap. 2.

3.3 Beale's Restart Rule

The Shanno algorithm pays special attention to the restarting procedure. Following Beale [6] he notices that when restarts are done, i.e. when

$$d_r = -g_r$$

for some r, then the direction is poorly scaled. These restarts are performed in order to improve the efficiency of a conjugate gradient algorithm, e.g. to avoid very small steps taken by a method in situations described by Powell and discussed in Sect. 2.6.

It is observed that using d_r produces very small reduction of the objective function and many function evaluations of the step-size selection procedure. On the other hand, the restarts, at least theoretically, guarantee n quadratic rate of convergence instead of a linear rate in their absent (cf. Sect. 4.3). Taking this into account Beale proposes the restarts with the last computed direction of descent d_r rather than restarting with $-g_r$. He shows that these directions not only guarantee faster rate of convergence but also potentially decent reduction of the objective function [6]. Beale modifies then the direction rule for iterates $r+1, r+2, \ldots, d_{r+n-1}$ to guarantee that directions $d_r, d_{r+1}, \ldots, d_{r+n-1}$ are conjugate in the case of a strongly quadratic function (see the analysis in Sect. 1.11)

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T d_k}{y_k^T d_k} d_k + \frac{g_{k+1}^T y_r}{d_r^T y_r} d_r. \tag{3.20}$$

The experience with the Beale's algorithm has been disappointing and this can be explained if we put the Beale's direction rule (3.20) into the framework considered by Perry and Shanno:

$$\begin{aligned} d_{k+1} &= -\left(I - \frac{d_k y_k^T}{d_k^T y_k} - \frac{d_r y_r^T}{d_r^T y_r}\right) g_{k+1} \\ &= -P_{k+1} g_{k+1}. \end{aligned}$$

The matrix P_{k+1} is not symmetric. Using the framework for the memoryless BFGS method we can transform matrix P_{k+1} to both symmetric and positive definite.

Define

$$\begin{aligned} \hat{H}_{r+1} &= I - \frac{s_r y_r^T + y_r s_r^T}{s_r^T y_r} + \left(1 + \frac{y_r^T y_r}{s_r^T y_r}\right) \frac{s_r s_r^T}{s_r^T y_r} \\ \hat{H}_{k+1} &= \hat{H}_{r+1} - \frac{s_k y_k^T \hat{H}_{r+1} + \hat{H}_{r+1} y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{y_k^T \hat{H}_{r+1} y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k} \end{aligned} \tag{3.21}$$

and

$$d_{k+1} = -\hat{H}_{k+1} g_{k+1}. \tag{3.22}$$

By evaluating (3.22) we come to the formula for d_{k+1}:

$$\begin{aligned} d_{k+1} = &-\hat{H}_{r+1} g_{k+1} + \frac{s_k^T y_{k+1}}{s_k^T y_k} \hat{H}_{r+1} y_k \\ &- \left[\left(1 + \frac{y_k \hat{H}_{r+1} y_k}{s_k^T y_k}\right) \frac{g_{k+1}^T s_k}{s_k^T y_k} - \frac{y_k^T \hat{H}_{r+1} g_{k+1}}{s_k^T y_k}\right] s_k \end{aligned} \tag{3.23}$$

with

$$\begin{aligned} \hat{H}_{r+1} g_{k+1} = &\, g_{k+1} - \frac{g_{k+1}^T s_r}{s_r^T y_r} y_r \\ &+ \left[\left(1 + \frac{y_r^T y_r}{s_r^T y_r}\right) \frac{g_{k+1}^T s_r}{s_r^T y_r} - \frac{g_{k+1}^T y_r}{s_r^T y_r}\right] s_r \end{aligned}$$

and

$$\hat{H}_{r+1} y_k = y_k - \frac{s_r^T y_k}{s_r^T y_r} y_r + \left[\left(1 + \frac{y_r^T y_r}{s_r^T y_r}\right) \frac{s_r^T y_k}{s_r^T y_r} - \frac{y_r^T y_k}{s_r^T y_r}\right] s_r.$$

The above formulae do not require matrix-vector products and we need to store only seven vectors: x_{k+1}, x_k, g_{k+1}, g_k, y_r, d_r and d_k.

Further improvement of the update is still possible by applying self-scaling mechanism based on the parameter γ_k discussed previously:

$$\hat{H}_{r+1} = \gamma_r \left(I - \frac{s_r y_r^T + y_r s_r^T}{s_r^T y_r} + \frac{y_r^T y_r}{s_r^T y_r} \frac{s_r s_r^T}{s_r^T y_r} \right) + \frac{s_r s_r^T}{s_r^T y_r}$$

with

$$\gamma_r = \frac{s_r^T y_r}{y_r^T y_r}.$$

Interesting interpretation of the Beale's restarting rule is obtained if we assume that the directional minimization is exact. Then, from formulae (3.21)–(3.22), we can interpret the direction d_{k+1} as calculated by the preconditioned conjugate gradient algorithm

$$d_{k+1} = -\hat{H}_{r+1} g_{k+1} + \frac{y_k^T \hat{H}_{r+1} g_{k+1}}{s_k^T y_k} d_k$$

with the matrix $\hat{H}_{r+1}$ solely defined by vectors s_r and y_r.

Having defined several versions of a scaled conjugate gradient algorithm the question is what combination of the introduced rules should be put together to get the best method. Shanno [191] performed several tests to discover that the Beale's restarts are helpful together with the Powell's criterion discussed in Sect. 2.6 (cf. (2.67))

$$|g_{k+1}^T g_k| \geqslant 0.2 \|g_{k+1}\|^2. \tag{3.24}$$

3.4 Convergence of the Shanno Algorithm

First, we analyze the properties of the matrix implicitly used in (3.18) which is the backbone of the Shanno algorithm. Consider the matrix

$$H_{k+1} = \gamma_k \left(I - \frac{s_k y_k^T + y_k s_k^T}{s_k^T y_k} + \frac{y_k^T y_k}{\left(s_k^T y_k\right)^2} s_k s_k^T \right) + \frac{s_k s_k^T}{s_k^T y_k}. \tag{3.25}$$

Davidon [51] shows that H_{k+1} has $n-2$ eigenvalues equal to γ_k:

$$\gamma_k = \frac{s_k^T y_k}{y_k^T y_k};$$

the other eigenvalues are

$$\lambda_{k+1} = \frac{s_k^T s_k}{s_k^T y_k} (1 - \sin \xi_k) \tag{3.26}$$

$$\Lambda_{k+1} = \frac{s_k^T s_k}{s_k^T y_k}(1+\sin\xi_k) \tag{3.27}$$

with ξ_k defined by

$$\cos^2\xi_k = \frac{\left(s_k^T y_k\right)^2}{\left(s_k^T s_k\right)\left(y_k^T y_k\right)}. \tag{3.28}$$

The corresponding eigenvectors are

$$v_{k+1} = s_k - \Lambda_{k+1} y_k$$
$$V_{k+1} = s_k - \lambda_{k+1} y_k.$$

In order to verify these claims notice that if λ_{k+1} and Λ_{k+1} satisfy

$$\lambda_{k+1}\Lambda_{k+1} = \gamma_k \frac{s_k^T s_k}{s_k^T y_k} \tag{3.29}$$

$$\lambda_{k+1} + \Lambda_{k+1} = \gamma_k \left(\frac{\left(s_k^T s_k\right)\left(y_k^T y_k\right)}{\left(s_k^T y_k\right)^2} + \frac{s_k^T s_k}{s_k^T y_k} \right) \tag{3.30}$$

then λ_{k+1}, Λ_{k+1} are indeed eigenvalues with the corresponding eigenvectors v_{k+1} and V_{k+1}. Equations (3.29)–(3.30) give the quadratic equation for calculating one of the eigenvalues

$$\left[\gamma_k \left(\frac{\left(s_k^T s_k\right)\left(y_k^T y_k\right)}{\left(s_k^T y_k\right)^2} + \frac{s_k^T s_k}{s_k^T y_k} \right) - \lambda_{k+1} \right] \lambda_{k+1} = \gamma_k \frac{s_k^T s_k}{s_k^T y_k}. \tag{3.31}$$

The solution to (3.31), after some elaborate transformations, is given by (3.26).

The basic Shanno algorithm is as follows.

Algorithm 3.1. (The basic Shanno algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

   $$d_1 = -g_1$$

   and $k = 1$.
2. Find α_k which satisfies the Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise calculate d_{k+1} according to

   $$d_{k+1} = -H_{k+1} g_{k+1}$$

   where H_{k+1} is defined by (3.25)
4. Increase k by one and go to Step 2.

Shanno proves two theorems concerning the global convergence of $\{x_k\}$ generated by the algorithm. The first one deals with the convex problem.

Theorem 3.1. *Suppose that $\{x_k\}$ is generated by the basic Shanno algorithm* – Algorithm 3.1. *Assume also that there exist positive numbers m and M such that*

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2 \tag{3.32}$$

for all x and z in $\mathscr{R}^n$. Then $\{x_k\}$ converges to the minimizer of f.

Proof. We have already shown that matrix H_{k+1} is positive definite under the Wolfe curvature condition. Thus there exists its square root which is positive definite – $H_{k+1}^{1/2}$. Therefore, if we set $z_{k+1} = H_{k+1}^{1/2} g_{k+1}$ we can provide the expression for $\cos^2 \theta_{k+1}$:

$$\begin{aligned} \cos^2 \theta_{k+1} &= \frac{\left(g_{k+1}^T d_{k+1}\right)^2}{\left(g_{k+1}^T g_{k+1}\right)\left(d_{k+1}^T d_{k+1}\right)} \\ &= \frac{\|z_{k+1}\|^4}{z_{k+1}^T H_{k+1} z_{k+1} z_{k+1}^T H_{k+1}^{-1} z_{k+1}}. \end{aligned} \tag{3.33}$$

Here, θ_{k+1} is the angle between vectors $-g_{k+1}$ and d_{k+1}. From the Kantorovich lemma (Lemma 1.2) we can write

$$\cos^2 \theta_{k+1} \geqslant \frac{4\lambda_{min}\lambda_{max}}{\left(\lambda_{min} + \lambda_{max}\right)^2}, \tag{3.34}$$

where λ_{min} is the smallest and λ_{max} the largest eigenvalue of H_{k+1}.

The next step in the proof is to show that λ_{k+1} is the smallest and Λ_{k+1} is the largest eigenvalue of H_{k+1}, i.e. that we have

$$\lambda_{k+1} \leqslant \gamma_k \leqslant \Lambda_{k+1}. \tag{3.35}$$

The right-hand side inequality is the result of the inequality

$$\cos^2 \xi_k = \frac{\left(s_k^T y_k\right)^2}{\left(s_k^T s_k\right)\left(y_k^T y_k\right)} \leqslant 1$$

which gives

$$\begin{aligned} \gamma_k = \frac{s_k^T y_k}{y_k^T y_k} &\leqslant \frac{s_k^T s_k}{s_k^T y_k} \\ &\leqslant \frac{s_k^T s_k}{s_k^T y_k}\left(1 + \sin \xi_k\right). \end{aligned}$$

Therefore, according to (3.27), we have $\gamma_k \leqslant \Lambda_{k+1}$.

In order to prove

$$\lambda_{k+1} \leqslant \gamma_k$$

notice that

$$\begin{aligned} 1 - \sin\xi_k &\leqslant 1 - \sin^2\xi_k = \cos^2\xi_k \\ &= \frac{\left(s_k^T y_k\right)^2}{\left(s_k^T s_k\right)\left(y_k^T y_k\right)} \end{aligned}$$

from the definition of ξ_k. This implies that

$$\lambda_{k+1} = \frac{s_k^T s_k}{s_k^T y_k}\left(1 - \sin\xi_k\right) \leqslant \gamma_k.$$

Having proved (3.35) we can refer to (3.34) and (3.29)–(3.30) to bound from below $\cos^2\theta_{k+1}$:

$$\begin{aligned} \cos^2\theta_{k+1} &\geqslant \frac{4\lambda_{k+1}\Lambda_{k+1}}{\left(\lambda_{k+1}+\Lambda_{k+1}\right)^2} \\ &= \frac{4 s_k^T s_k / y_k^T y_k}{4\left(s_k^T s_k\right)^2 / \left(s_k^T y_k\right)^2} \\ &= \frac{\left(s_k^T y_k\right)^2}{\left(s_k^T s_k\right)\left(y_k^T y_k\right)} \\ &= \cos^2\xi_k. \end{aligned}$$

In order to complete the proof we need to show that $\cos^2\xi_k$ is bounded below.

Based on assumption (3.32) and the Taylor's expansion theorem it is easy to verify the following estimates

$$\begin{aligned} m\|s_k\|^2 \leqslant s_k^T y_k &\leqslant M\|s_k\|^2 \\ y_k^T y_k &\leqslant M\|s_k\|^2 \end{aligned} \tag{3.36}$$

which together with (3.28) imply that

$$\begin{aligned} \cos^2\xi_k = \frac{\left(s_k^T y_k\right)^2}{\|s_k\|^2\|y_k\|^2} &= \frac{s_k^T y_k}{\|y_k\|^2}\frac{s_k^T y_k}{\|s_k\|^2} \\ &\geqslant \frac{m}{M}m = \frac{m^2}{M} \end{aligned} \tag{3.37}$$

since $s_k^T y_k$ is positive.

Remarks after the proof of Theorem 2.1 imply that

$$\lim_{k\to\infty} \|g_k\|^2 = 0.$$

Due to (3.32) f is strongly convex thus there is only one point $\bar{x}$ which is a minimizer of f and at this point we must have $g(\bar{x}) = 0$. □

The assumption that f is strictly convex is very restrictive. Shanno proves another global convergence result by imposing a weaker condition on f, requiring that the Hessian matrix is uniformly bounded above. However, in this case he is only able to show that $\liminf_{k\to\infty} \|g_k\| = 0$ demanding additionally that $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$. As we have seen in the previous chapter this condition is often required in order to guarantee global convergence of the Polak–Ribière version of the standard conjugate gradient algorithm. (Notice that the Hestenes–Stiefel method is equivalent to the Polak–Ribière algorithm under the exact directional minimization and that the discussed algorithm originates in the Hestenes–Stiefel approach).

Theorem 3.2. *Suppose that* $\{x_k\}$ *is generated by the basic Shanno algorithm* – Algorithm 3.1. *Assume also that there exists a positive number* M *such that*

$$z^T \nabla^2 f(x) z \leqslant M\|z\|^2 \tag{3.38}$$

for all x *and* z *in* $\mathscr{R}^n$. *If*

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0, \tag{3.39}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.40}$$

Proof. Since we no longer assume (3.32) we have to provide another estimate from below for $\cos^2 \xi_k$. Now we refer to the Wolfe curvature condition

$$y_k^T s_k \geqslant (\eta - 1) g_k^T s_k$$

which together with (3.36) give us

$$\begin{aligned}
\cos^2 \theta_{k+1} &\geqslant \cos^2 \xi_k \\
&\geqslant \frac{\left((\eta-1)^2 s_k^T g_k\right)^2}{\left(s_k^T s_k\right)\left(y_k^T y_k\right)} \\
&= \frac{(\eta-1)^2 \cos^2 \theta_k \|g_k\|^2 \|s_k\|^2}{M^2 \|s_k\|^4} \\
&\geqslant \frac{(\eta-1)^2}{M^2} \cos^2 \theta_k \frac{\|g_k\|^2}{\|s_k\|^2} \\
&\geqslant \left(\frac{(\eta-1)}{M}\right)^{2k} \cos^2 \theta_1 \prod_{j=1}^{k} \frac{\|g_j\|^2}{\|s_j\|^2}.
\end{aligned} \tag{3.41}$$

Assume now that (3.40) does not hold, i.e. there exists $\varepsilon > 0$ such that

$$\liminf_{k\to\infty} \|g_k\| = \varepsilon. \tag{3.42}$$

According to (3.39) and (3.42) there exists $\hat{k}$ such that for $k \geqslant \hat{k}$

$$\frac{\|g_k\|^2}{\|s_k\|^2} \geqslant \frac{M^2}{(\eta-1)^2}. \tag{3.43}$$

Equations (3.41) and (3.43) allow us to write

$$\begin{aligned} \cos^2\theta_{k+1} &\geqslant \cos^2\xi_k \\ &\geqslant \cos^2\theta_1 \prod_{j=1}^{k} \left[\left(\frac{(\eta-1)}{M} \right)^2 \frac{\|g_j\|^2}{\|s_j\|^2} \right] \\ &> \varepsilon > 0. \end{aligned} \tag{3.44}$$

However, from Theorem 2.1, since (3.44) holds, we also have $\lim \|g_k\| = 0$ which contradicts (3.42). This completes the proof. □

The memoryless BFGS method discussed so far is not the most efficient algorithm among methods investigated by Shanno [191]. He recommends the algorithm which takes steps defined by (3.25) every n iterations or when the Powell's condition (3.24) is not fulfilled. Otherwise the direction d_{k+1} is given by the memoryless BFGS rule with the Beale's modification, i.e.

$$d_{k+1} = -H_{k+1} g_{k+1} \tag{3.45}$$

$$\begin{aligned} H_{k+1} = H_{r+1} &- \left(\frac{H_{r+1} y_k s_k^T + s_k y_k^T H_{r+1}}{s_k^T y_k} \right) \\ &+ \left(1 + \frac{y_k^T H_{r+1} y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k}, \end{aligned} \tag{3.46}$$

where H_{r+1} is defined by (3.25) with k replaced by r.

The memoryless quasi-Newton algorithm proposed by Shanno is as follows.

Algorithm 3.2. (The Shanno algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -g_1$$

and $k = 1$, $r = 1$.

2. Find α_k which satisfies the Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.

3. If $\|g_{k+1}\| = 0$ then STOP. If $k = (r+1)n$, or

$$|g_{k+1}^T g_k| \geqslant 0.2\|g_{k+1}\|^2 \tag{3.47}$$

calculate d_{k+1} according to

$$d_{k+1} = -H_{k+1} g_{k+1} \tag{3.48}$$

where H_{k+1} is defined by (3.25). Otherwise calculate d_{+1} according to (3.48) with H_{k+1} expressed by (3.46). If $k = (r+1)n$ increase r by one.

4. Increase k by one and go to Step 2.

The proof of Theorem 3.1 is used to show global convergence of the above algorithm.

Theorem 3.3. *Suppose that $\{x_k\}$ is generated by the memoryless quasi-Newton algorithm by Shanno* – Algorithm 3.2. *Assume also that there exist positive numbers m and M such that (3.32) holds for all x and z in $\mathscr{R}^n$. Then $\{x_k\}$ converges to the minimizer of f.*

Proof. As in the proof of Theorem 3.1 the major part of the proof concerns the analysis of eigenvalues of matrices H_k applied in the algorithm.

We start by referring to the result given in [192] and concerning the matrix

$$Q_{k+1} = H_{r+1}^{-1} H_{k+1},$$

where H_{r+1} is given by (3.25) and H_{k+1} by (3.46). Shanno proves that the matrix Q_{k+1} has $n-2$ eigenvalues equal to unity and two nonzero eigenvalues δ_{k+1} and Δ_{k+1} such that

$$\delta_{k+1} \leqslant 1 \leqslant \Delta_{k+1}$$

and satisfying

$$\delta_{k+1}\Delta_{k+1} = \frac{p_k^T H_{r+1}^{-1} p_k}{s_k^T y_k} = a \tag{3.49}$$

$$\delta_{k+1} + \Delta_{k+1} = \frac{s_k^T H_{r+1}^{-1} s_k}{s_k^T y_k}\left(1 + \frac{y_k^T H_{r+1} y_k}{s_k^T y_k}\right) = ab. \tag{3.50}$$

Equations (3.49)–(3.6) can be solved with respect to δ_{k+1} and Δ_{k+1} giving

$$\delta_{k+1} = \frac{ab - \sqrt{a^2b^2 - 4a}}{2}$$

$$\Delta_{k+1} = \frac{ab + \sqrt{a^2b^2 - 4a}}{2}.$$

Having expressions for δ_{k+1} and Δ_{k+1} we obtain the condition number of Q_{k+1}:

$$\begin{aligned}\kappa(Q_{k+1}) &= \frac{\Delta_{k+1}}{\delta_{k+1}} \\ &= \frac{ab+\sqrt{a^2b^2-4a}}{ab-\sqrt{a^2b^2-4a}} \\ &\leqslant \frac{4a^2b^2}{4a} = ab^2.\end{aligned}$$

If we can bound from above values of a, b then we can show that $\kappa(Q_{k+1})$ is bounded and due to the relations

$$\begin{aligned}\kappa(H_{k+1}) &= \kappa(H_{r+1}Q_{k+1}) \\ &\leqslant \kappa(H_{r+1})\,\kappa(Q_{k+1}) \qquad (3.51)\end{aligned}$$

we are able to bound $\kappa(H_{k+1})$ provided that $\kappa(H_{r+1})$ are bounded.

From (3.49)

$$a \leqslant \frac{1}{\lambda_{r+1}}\frac{s_k^T s_k}{s_k^T y_k} \leqslant \frac{1}{\lambda_{r+1}}\frac{1}{m}$$

(recall that λ_{r+1} is the smallest eigenvalue of H_{r+1}) and by (3.50) we have (by taking into account that Λ_{r+1} is the largest eigenvalue of H_{r+1})

$$b \leqslant \left(1+\Lambda_{r+1}\frac{y_k^T y_k}{s_k^T y_k}\right) \leqslant \left(1+\Lambda_{r+1}\frac{m}{M}\right)$$

where the last inequality follows from (3.37).

According to the formula for Λ_{r+1} (cf. (3.27)):

$$\Lambda_{r+1} \leqslant \frac{2}{m}$$

thus both a and b are bounded.

Now, from (3.26)–(3.28), notice that

$$\begin{aligned}\kappa(H_{r+1}) &= \frac{\Lambda_{k+1}}{\lambda_{k+1}} = \frac{1+\sin\xi_k}{1-\sin\xi_k} \\ &= \frac{(1+\sin\xi_k)^2}{\cos^2\xi_k} = \frac{4\|s_k\|^4 y_k^T y_k}{\left(s_k^T y_k\right)^2} \\ &\leqslant \frac{4M}{m}\end{aligned}$$

thus, both $\kappa(Q_{k+1})$ and $\kappa(H_{r+1})$ are bounded. Eventually, from (3.51), we know that $\kappa(H_{k+1})$ are bounded.

We can write, using the same arguments which led to (3.34),

$$\cos^2\theta_{k+1} \geqslant \frac{4\lambda_{min}^k\lambda_{max}^k}{\left(\lambda_{min}^k+\lambda_{max}^k\right)^2} = \frac{4\kappa(H_{k+1})}{\left(1+\kappa(H_{k+1})\right)^2}$$

(where λ_{min}^k and λ_{max}^k are the smallest and the largest eigenvalue of H_{k+1}) which implies that

$$\cos^2\theta_{k+1} \geqslant \nu > 0$$

for some ν and all k. The rest of the proof mimics the appropriate lines of the proof of Theorem 3.1. □

The analysis employed in the proof of Theorem 3.2 can be used to show that the memoryless quasi-Newton algorithm is globally convergent provided that (3.39) holds.

Theorem 3.4. *Suppose that $\{x_k\}$ is generated by the memoryless quasi-Newton algorithm by Shanno* – Algorithm 3.2. *Assume also that there exists a positive number M such that (3.38) holds for all x and z in $\mathscr{R}^n$. If*

$$\lim_{k\to\infty}\|x_{k+1}-x_k\| = 0, \tag{3.52}$$

then

$$\liminf_{k\to\infty}\|g_k\| = 0. \tag{3.53}$$

Proof. If we can show that (3.47) holds for all sufficiently large k then d_{k+1} is calculated according to (3.25), (3.48) and the analysis of $\cos^2\theta_{k+1}$ given in the proof of Theorem 3.2 can also be applied in this case.

Following [192] assume that (3.53) does not hold thus there exists $\varepsilon > 0$ such that

$$\|g_k\| \geqslant \varepsilon \tag{3.54}$$

for all $k \geqslant k_1$. Furthermore, due to (3.52) there exists k_2 such that

$$\|s_k\| = \|x_{k+1}-x_k\| \leqslant 0.8\frac{\varepsilon}{2M} \tag{3.55}$$

for all $k \geqslant k_2$. Define $\hat{k} = \max[k_1,k_2]$, then

$$g_{k+1} = g_k + \int_0^1 \nabla^2 f(x_k+\zeta s_k)s_k d\zeta$$

from the Taylor's expansion theorem. Since, from (3.55),

$$\int_0^1 g_{k+1}^T \nabla^2 f(x_k + \zeta s_k) s_k d\zeta \leqslant M\|g_{k+1}\| \left(0.8\frac{\varepsilon}{2M}\right) \leqslant 0.8\|g_{k+1}\|^2$$

we can write

$$g_{k+1}^T g_{k+1} \leqslant g_{k+1}^T g_k + 0.8\|g_{k+1}\|^2$$

which implies that

$$|g_{k+1}^T g_k| \geqslant 0.2\|g_{k+1}\|^2$$

thus restarts are done at every kth iteration such that $k \geqslant \hat{k}$.

Next, using $\cos^2\theta_{\hat{k}}$ instead of $\cos^2\theta_1$, as in the proof of Theorem 3.2, we can show that

$$\cos^2\theta_k \geqslant \varepsilon \tag{3.56}$$

for all $\geqslant \hat{k}$. By referring to Theorem 2.1 we come to the contradiction with (3.54). This completes the proof. □

The Shanno's algorithm requires seven vectors to be stored – three more than the standard Polak–Ribière method. However, according to Shanno's numerical tests it also comfortably outperforms other versions of the standard conjugate gradient algorithm [191].

3.5 Notes

Our presentation of the Shanno algorithms is based on the papers [191] and [192]. Shanno analyzed eigenvalues of the memoryless quasi-Newton matrices to prove convergence of his algorithms. Because of this his approach gives deeper insight to considered algorithms in contrast to the other approach of analyzing convergence of quasi-Newton algorithms – it is presented in Chap. 5 together with the limited memory versions of quasi-Newton methods. We could use that approach to prove Theorem 3.1 – however it is not clear whether that proving technique could be applied in the proofs of the other theorems presented in the chapter.

The efficient implementation of the Shanno algorithm is available as CONMIN code [194]. In Figs. 3.1–3.6 we show efficiency of the CONMIN code. The Shanno–Phua code is numerically compared to that developed by Gilbert and Nocedal [78] and called CG+ (we applied the option METHOD=2 of the CG+ code in our tests which corresponds to the standard Polak–Ribière algorithm). The problems which we tested come from the CUTE collection [13] and are characterized in Table 3.1.

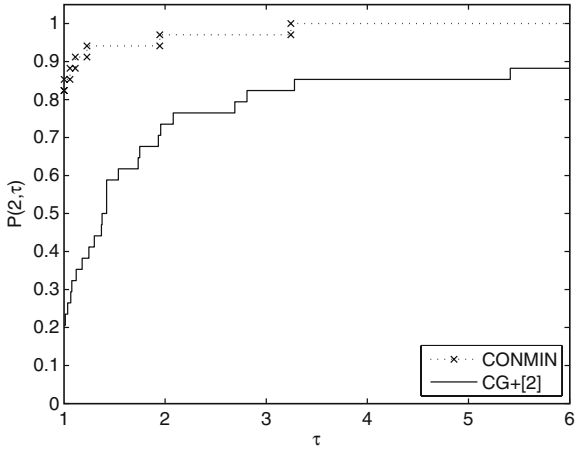

Fig. 3.1 Cumulative distribution ratio of the number of function evaluations: comparison of CONMIN to CG+

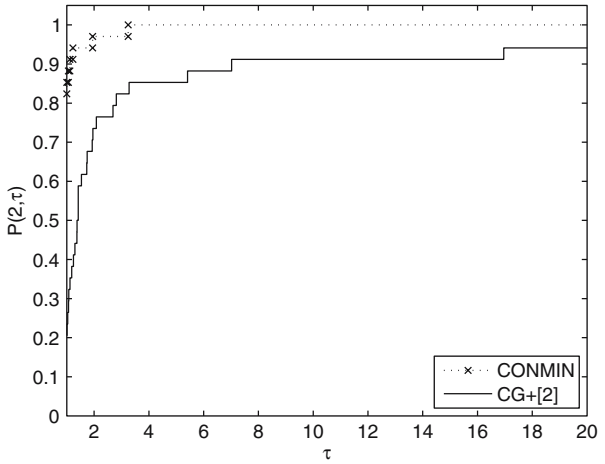

Fig. 3.2 Cumulative distribution ratio of the number of function evaluations: comparison of CONMIN to CG+

The results of the comparison are presented graphically with the help of performance profiles as developed by Dolan and Moré [57]. We show plots of the distribution functions for the performance ratios defined by three measures: the number of function evaluations, the CPU computing time and the number of iterations. Following [57] we define

$$m(i,p,s) = \text{value of the } i\text{th measure obtained by applying solver } s \text{ to solve problem } p$$

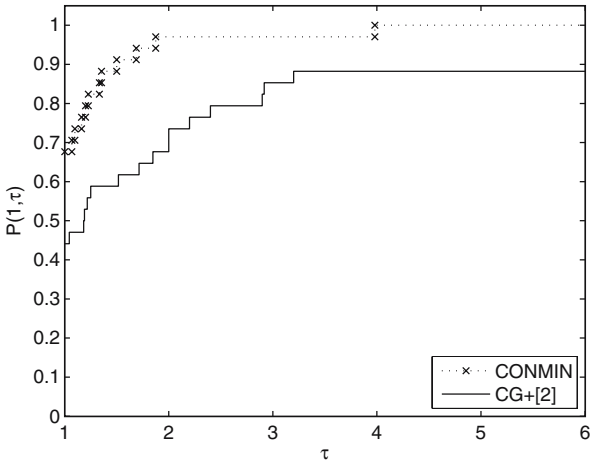

Fig. 3.3 Cumulative distribution ratio of CPU time: comparison of CONMIN to CG+

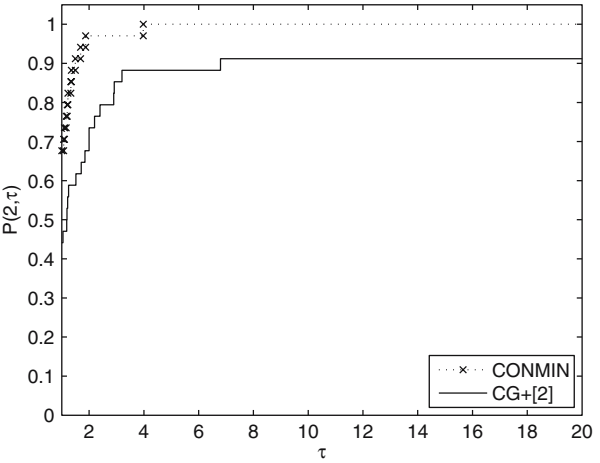

Fig. 3.4 Cumulative distribution ratio of CPU time: comparison of CONMIN to CG+

(here $i = 1$ corresponds to computing time, $i = 2$ to the number of function evaluations and $i = 3$ to the number of iterations). If we assume that we have n_p problems in the set $\mathscr{P}$ and n_s solvers in the set $\mathscr{S}$ then

$$r(i,p,s) = \frac{m(i,p,s)}{\min\{m(i,p,s) :\ s \in \mathscr{S}\}}$$

is the performance ratio of the sth solver with respect to the pth problem. Then the performance profile is the function which for each $\tau \in [1,\infty)$ assigns

$$P(i,\tau) = \frac{1}{n_p}\text{size}\,\{p \in \mathscr{P} :\ r(i,p,s) \leqslant \tau\}\,.$$

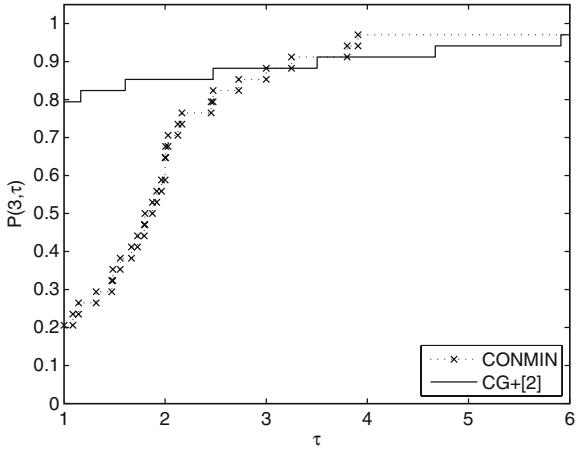

Fig. 3.5 Cumulative distribution ratio of the number of iterations: comparison of CONMIN to CG+

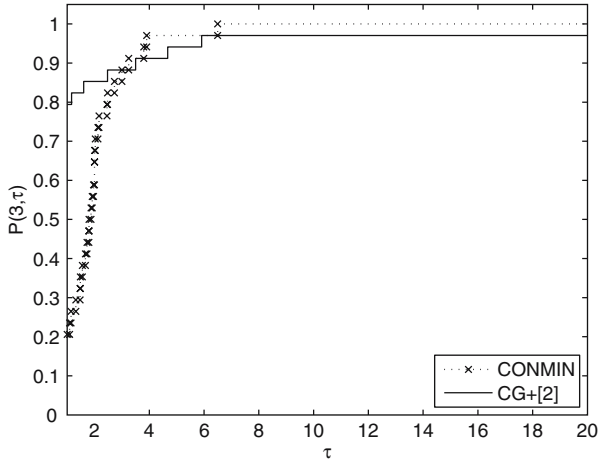

Fig. 3.6 Cumulative distribution ratio of the number of iterations: comparison of CONMIN to CG+

The results of our comparison show clear superiority (on our test problems) of the memoryless quasi-Newton algorithm, as developed by Shanno, over the standard conjugate gradient algorithm, as developed by Gilbert and Nocedal. The results are not surprising if we take into account that CONMIN requires seven vectors to bo stored in contrast to four needed by CG+.

The stopping criterion was

$$\|g_k\| / \max(1, \|x_k\|) \leqslant 10^{-6}.$$

Table 3.1 Unconstrained test problems

Problems	Dimension
BROWNAL, EXTROSNB, HILBERTA	10
FLETCHBV, FLETCHCR, MANCINO, POWER, SENSORS,	100
EIGENALS	110
DQRTIC, GENROSE	500
BROYDN7D, GENHUMPS, NONCVXU2, SPARSINE,	1000
BRYBND, DQDRTIC, MOREBV, SINQUAD, VAREIGVL	5000
CHAINWOO, DIXON3DQ, FMINSURF, LIARWD, NONCVXUN, NONDIA, NONDQUAR, POWELLSG, QUARTIC, SCHMVETT, SPMSRTLS, SROSENBR, TOINTGS, TQUARTIC, TRIDIA, WOODS	10000

Chapter 4
Preconditioned Conjugate Gradient Algorithms

4.1 Introduction

In Chap. 1 we introduced conjugate gradient algorithms with scaling. Such methods are called by Hestenes generalized conjugate gradient algorithms, or preconditioned conjugate gradient algorithms by others. The purpose of scaling in methods applied to quadratics is to transform eigenvalues of the Hessian matrix. Theorem 1.11 suggests that if eigenvalues are clustered then a conjugate gradient algorithm minimizes the quadratic in the number of iterations comparable to the number of clusters. Preconditioning in the quadratic case significantly improves the efficiency of a conjugate gradient algorithm. In fact it transforms a conjugate gradient algorithm to a viable optimization technique widely used in several numerical algebra problems especially when problem's dimension is large.

Scaling of a conjugate gradient algorithm applied to a nonconvex function is not comprehensively described in the literature. Although the general idea of scaling is the same as in the quadratic case there is a significant difference between two cases. In a quadratic case the Hessian matrix is constant and is known in advance. In a nonconvex case the Hessian matrices change during an iterative process and usually they are not available but are approximated. For these reasons preconditioning does not always improve the efficiency of a conjugate gradient algorithm and makes it competitive with quasi-Newton methods.

In the chapter we describe a generic algorithm which uses scaling and we do not investigate its convergence properties. Then, already discussed algorithms such as the Beale's algorithm is described as a preconditioned conjugate gradient algorithm. The relationship of a preconditioned conjugate gradient algorithm with quasi-Newton methods is also analyzed.

Preconditioning in nonconvex conjugate gradient algorithms is applied in order to increase their efficiency. In the case of quadratic functions the efficiency of these methods is measured by the number of iterations needed to find a solution. In the nonconvex case, since in general a solution is found by an infinite process, we have to introduce the measure of efficiency and that is done in the next section.

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

4.2 Rates of Convergence

A nonlinear conjugate gradient algorithm generates a sequence $\{x_k\}$ which is supposed to converge to a solution $\bar{x}$. The efficiency of the algorithm can be measured by the speed with which x_k approaches $\bar{x}$, or $\|x_k - \bar{x}\|$ tends to zero. For example, we can try to establish for the sequence the relation

$$\|x_{k+1} - \bar{x}\| \leqslant \gamma \|x_k - \bar{x}\|^p. \tag{4.1}$$

If (4.1) holds for all k and $p \geqslant 1$, $\gamma \leqslant 1$ then we also have

$$\|x_k - \bar{x}\| \leqslant \gamma^k \|x_1 - \bar{x}\|^{p^k}$$

and since x_k converges to $\bar{x}$ we can assume that $\|x_1 - \bar{x}\| < 1$. Therefore, the error vectors $x_k - \bar{x}$ are decreasing as rapidly as a geometric progression with the ratio γ. Thus, in analogy with the root test for the convergence of series, we are led to consider geometric averages of the $\|x_k - \bar{x}\|$.

For these reasons, we consider two measures of the speed of convergence for the sequence $\{x_k\}$.

Definition 4.1. Let $\{x_k\}$ be any convergent sequence with limit $\bar{x}$. We say that $\{x_k\}$ converges with Q-order at least p if there is a positive constant γ such that

$$\|x_{k+1} - \bar{x}\| \leqslant \gamma \|x_k - \bar{x}\|^p$$

for all k. We denote γ by $Q_p\{x_k\}$.

The second measure is also borrowed from [163] (cf. [153], [206]).

Definition 4.2. Let $\{x_k\}$ be any convergent sequence with limit $\bar{x}$. We say that $\{x_k\}$ converges with R-order at least p if there are constants $\gamma \in (0, \infty)$, $\theta \in (0,1)$ such that

$$\|x_{k+1} - \bar{x}\| \leqslant \gamma \theta^{p^k}$$

for all k. We denote γ by $R_p\{x_k\}$.

Definition 4.1 does not uniquely characterizes the speed of convergence. If $\{x_k\}$ converges with Q-order at least p then it also converges with Q-order at least $\tilde{p}$ for any $\tilde{p} \in (1, p]$. The following definition refers to the uniquely defined speed of convergence.

Definition 4.3. Let $\{x_k\}$ be any convergent sequence with limit $\bar{x}$. We say that $\{x_k\}$ converges with the exact Q-order p if there are positive constants γ_1, γ_2 such that

$$\gamma_1 \|x_k - \bar{x}\|^p \leqslant \|x_{k+1} - \bar{x}\| \leqslant \gamma_2 \|x_k - \bar{x}\|^p$$

for all k.

Following the same arguments applied to R-order of convergence we come to the next definition.

Definition 4.4. Let $\{x_k\}$ be any convergent sequence with the limit $\bar{x}$. We say that $\{x_k\}$ converges with the exact R-order p if there are constants $\gamma_1, \gamma_2 \in (0,\infty)$ and $\eta, \theta \in (0,1)$ such that

$$\gamma_1 \eta^{p^k} \leqslant \|x_{k+1} - \bar{x}\| \leqslant \gamma_2 \theta^{p^k}$$

for all k.

The exact Q and R-orders of convergence are uniquely defined. However, there are sequences which do not have an exact Q-order (R-order) of convergence although they converge with Q-order (R-order) at least p for some $p > 1$. Therefore, we can uniquely define the number

$$Q\{x_k\} = \sup\{p > 1 : \ \{x_k\} \text{ converges with } Q\text{-order at least } p\} \tag{4.2}$$

which leads to the definition.

Definition 4.5. The number $Q\{x_k\}$ defined in (4.2) is called the Q-order of the sequence $\{x_k\}$.

One can show that if $\{x_k\}$ has an exact Q-order of convergence then (cf. [163])

$$Q\{x_k\} = \sup\left\{p > 1 : \ \lim_{k\to\infty} \frac{\|x_{k+1} - \bar{x}\|}{\|x_k - \bar{x}\|^p} = 0\right\} \tag{4.3}$$

$$Q\{x_k\} = \inf\left\{p > 1 : \ \limsup_{k\to\infty} \frac{\|x_{k+1} - \bar{x}\|}{\|x_k - \bar{x}\|^p} = +\infty\right\}. \tag{4.4}$$

Analogously we have the number $R\{x_k\}$ for the R-order convergence.

Definition 4.6. The number

$$R\{x_k\} = \sup\{p > 1 : \ \{x_k\} \text{ converges with } R\text{-order at least } p\}$$

is called the R-order of convergence of $\{x_k\}$.

Similarly to (4.3)–(4.4) we have

$$R\{x_k\} = \sup\left\{p > 1 : \ \lim_{k\to\infty} \|x_k - \bar{x}\|^{1/p^k} = 0\right\}$$

$$R\{x_k\} = \inf\left\{p > 1 : \ \limsup_{k\to\infty} \|x_k - \bar{x}\|^{1/p^k} = 1\right\}$$

if $\{x_k\}$ has the exact R-order of convergence.

R-order of convergence presents less precise measure of the speed of convergence as shows the theorem stated in [163].

Theorem 4.1. *(i) If $\{x_k\}$ converges with Q-order at least p, then $\{x_k\}$ converges with R-order at least p.*

(ii) If $\{x_k\}$ has the exact Q-order of convergence p, then $\{x_k\}$ has the exact R-order of convergence p.

(iii) $R\{x_k\} \geqslant Q\{x_k\}$.

Sequences $\{x_k\}$ with Q-order at least 1, 2 play an important role in theory (and in practice), and we introduce some additional terminology that will be useful on occasion. When $Q_1\{x_k\} = 0$ we say that $\{x_k\}$ has Q-superlinear convergence at $\bar{x}$, while, if $0 < Q_1\{x_k\} < 1$ the convergence is called Q-linear. Analogously $Q_2\{x_k\} = 0$ means Q-subquadratic convergence and $0 < Q_2\{x_k\} < \infty$ Q-quadratic convergence at $\bar{x}$.

Subsequently we will work with sequences of orders of convergence greater than one. Analysis of these sequences is greatly helped by the sufficient conditions for R-orders convergence stated in [163] (cf. [153], [206]).

To this end assume that α_0,...,α_m are real numbers such that

$$\alpha_i \geqslant 0, \quad i = 0, \ldots, m \tag{4.5}$$

$$\alpha = \alpha_0 + \ldots + \alpha_m > 0 \tag{4.6}$$

and define the polynomial equation with these numbers

$$p_m(t) = t^{m+1} - \alpha_0 t^m - \alpha_1 t^{m-1} - \cdots - \alpha_m = 0. \tag{4.7}$$

Since our sufficient conditions for the R-order convergence will be expressed in terms of roots of equation (4.7) we refer to the following result proved in [154].

Proposition 4.1. *If (4.5)–(4.6) hold, then (4.7) has a unique positive root τ and all other roots have moduli not greater than τ.*

We are interested in the case when

$$\tau > 1$$

which, as can be easily shown, is equivalent to

$$\alpha > 1. \tag{4.8}$$

The next theorem, which we present after Potra [163], has also its versions stated in [153] and [206].

Theorem 4.2. *Suppose that conditions (4.5)–(4.6) and (4.8) are satisfied, $\tau > 1$ is the unique positive root of (4.7) and that $\{x_k\}$ converges to $\bar{x}$.*

(i) If there is a constant $\gamma > 0$ such that

$$\|x_{k+1} - \bar{x}\| \leqslant \gamma \prod_{i=0}^{m} \|x_{k-i} - \bar{x}\|^{\alpha_i}, \quad k = m, m+1, \ldots,$$

then $\{x_k\}$ converges with R-order at least τ.

(ii) If there are two constants $\gamma_1 > 0$, $\gamma_2 > 0$ *such that*

$$\gamma_1 \prod_{i=0}^{m} \|x_{k-i} - \bar{x}\|^{\alpha_i} \leqslant \|x_{k+1} - \bar{x}\| \leqslant \gamma_2 \prod_{i=0}^{m} \|x_{k-i} - \bar{x}\|^{\alpha_i},$$

$k = m, m+1, \ldots$, *then* $\{x_k\}$ *has the exact R-order of convergence* τ.

The theorem will be applied to prove the convergence rates of sequences generated by some version of a preconditioned conjugate gradient algorithm. Before doing that, we analyze the speed of convergence of sequences of standard conjugate gradient algorithms.

So far we have considered convergence rates of the sequence $\{\|x_k - \bar{x}\|\}$. The definitions and convergence results can be extended to a more general setting in which we work not with sequences but with iterative processes generating sequences. Furthermore, we can apply these concepts to arbitrary convergent sequences of positive numbers – details are given, for example, in [153] and [206]. Therefore, for the simplicity of presentation, if we refer to the rate of convergence of an iterative procedure, we mean rates of convergence of sequences constructed by the procedure.

When a conjugate gradient algorithm is applied to a general nonlinear function one cannot expect that convergence would occur in a finite number of iterations. However, when the function is strongly convex then close to a solution the convergence should be quite fast. If the current point x_k is near a solution then a quadratic approximation to the minimized function is adequate and we can refer to the finite convergence of a conjugate gradient algorithm.

Crowder and Wolfe [38] (see also [167]) prove that a conjugate gradient algorithm has at worst linear rate of convergence but they also show on a quadratic function that Q-order of superlinear convergence cannot be achieved. Furthermore, they claim that a *restarted* version of a conjugate gradient algorithm can exhibit a better rate of convergence.

The *restarted* version of a conjugate gradient algorithm assumes that every n iterations $d_k = -g_k$, or in other words after a fixed number of iterations we discard old information that may not be beneficial and we start building new conjugate directions. That strategy can be justified if the function approximation by a quadratic function is adequate. If, after n iterations, a solution is not achieved we have to switch to another quadratic approximation and build conjugate directions from the scratch.

The *restarted* conjugate gradient algorithm is stated as follows – we present its Polak–Ribière version, the others can be defined analogously.

Algorithm 4.1. (The restarted conjugate gradient algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -g_1, \tag{4.9}$$

$r = 1$ and $k = 1$.

2. Find α_k which minimizes f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : \ x = x_k + \alpha d_k, \ \alpha \in (0,\infty)\}$. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP. Otherwise, if $k = rn$ increase r by one and set

$$d_{k+1} = -g_{k+1}.$$

If $k \neq rn$ calculate d_{k+1} according to

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k,$$

where

$$\beta_{k+1} = \frac{(g_{k+1} - g_k)^T g_k}{\|g_k\|^2}.$$

4. Increase k by one and go to Step 2.

The rate of convergence of the restarted conjugate gradient algorithm was analyzed in [46], [47], [49] and in [33] – the main convergence result is presented below.

Theorem 4.3. *Suppose that*

(i) the level set $\mathscr{M} = \{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\}$ *is compact and convex,*
(ii) $\nabla^2 f(x)$ *is Lipschitz continuous on* $\mathscr{M}$*,*
(iii) there exist positive constants m and M such that

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2 \tag{4.10}$$

for all $x \in \mathscr{M}$, $z \in \mathscr{R}^n$.

Then for every sequence $\{x_k\}$ *generated by* Algorithm 4.1 *there exists a positive constant* γ *such that*

$$\limsup_{k\to\infty} \frac{\|x_{k+n} - \bar{x}\|}{\|x_k - \bar{x}\|^2} \leqslant \gamma < \infty.$$

where $\bar{x}$ *is the minimum point of* f.

The thesis of the theorem also applies to Algorithm 4.1 with β_k determined by the Fletcher–Reeves formula. The convergence rate is determined by comparing the algorithm to the Newton method which, as we show below, exhibits the Q-quadratic order of convergence.

Using restarts in conjugate gradient algorithms is fully justified from theoretical point of view and the stated convergence result gives another argument for using these versions of conjugate gradient algorithms. However, the result doesn't say much in favor of conjugate gradient algorithms. First, quasi-Newton methods can

have n-step Q-superquadratic rate of convergence, i.e.

$$\lim_{k\to\infty} \frac{\|x_{k+n} - \bar{x}\|}{\|x_k - \bar{x}\|^2} = 0,$$

for example if $\{x_k\}$ is generated by a quasi-Newton method discussed in [185]. Secondly, using restarts every n iterations is impractical when n is large, and for this class of problems conjugate gradient algorithms are intended.

Since rates of convergence of conjugate gradient algorithms are not satisfactory several improvements were proposed to speed up the convergence. Following earlier work on linear conjugate gradient algorithms attention was focused on using preconditioning. Two approaches to preconditioned conjugate gradient algorithms emerged. In the first approach, discussed in the next section, at every iteration a new preconditioning matrix is used, while in the second one a preconditioner is updated every fixed number of iterations and after the update conjugate gradient iterations are performed with the constant preconditioner. It appears that the second approach has led to significant improvement of conjugate gradient algorithms.

4.3 Conjugate Gradient Algorithm and the Newton Method

The Newton method has the Q-quadratic rate of convergence and for that reason it is the scheme which is often used in other algorithms to improve their efficiency. The Newton method for minimizing function f is stated below.

Algorithm 4.2. (The Newton algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$ and set $k = 1$.
2. Calculate

$$d_k = -\left[\nabla^2 f(x_k)\right]^{-1} g_k.$$

3. Substitute $x_k + d_k$ for x_{k+1}. If $\|g_{k+1}\| = 0$ then STOP, otherwise increase k by one and go to Step 2.

Theorem 4.4. *Suppose that the assumptions of* Theorem 4.3 *hold with the set $\mathscr{M}$ replaced by some neighborhood of $\bar{x}$ where $\bar{x}$ is the minimum point of f. Then*

(i) if the point x_1 is sufficiently close to $\bar{x}$, the sequence generated by Algorithm 4.2 *converges to $\bar{x}$,*

(ii) the rate of convergence of $\{x_k\}$ is Q-quadratic.

Proof. We have from the Taylor's expansion theorem

$$\begin{aligned} x_{k+1} - \bar{x} &= x_k - \bar{x} - \left[\nabla^2 f(x_k)\right]^{-1} g_k \\ &= \left[\nabla^2 f(x_k)\right]^{-1} \left(\nabla^2 f(x_k)(x_k - \bar{x}) - g_k\right) \\ &= \left[\nabla^2 f(x_k)\right]^{-1} (\nabla^2 f(x_k) - \int_0^1 \nabla^2 f\left(\bar{x} + t(x_k - \bar{x})dt\right) \times \\ &\quad \times (x_k - \bar{x}), \end{aligned} \tag{4.11}$$

since $g(\bar{x}) = 0$.

Equation (4.11), the Lipschitz continuity of $\nabla^2 f$ and the boundedness of $[\nabla^2 f]^{-1}$ imply that

$$\|x_{k+1} - \bar{x}\| \leqslant \|x_k - \bar{x}\|^2 M \int_0^1 L t dt = \frac{ML}{2} \|x_k - \bar{x}\|^2 \tag{4.12}$$

where M is such that $\| \left[\nabla^2 f(x_k)\right]^{-1} \| \leqslant M$. If x_k is sufficiently close to $\bar{x}$ we can assume that $\|x_k - \bar{x}\| \leqslant \frac{1}{2}$ and (4.12) ensures convergence of x_k to $\bar{x}$. Equation (4.12) also guarantees Q-quadratic rate of convergence. □

The Newton algorithm follows from the analogous Newton scheme applied to a set of nonlinear equations. Notice that minimizing a strongly convex function is equivalent to finding a point x which solves the equations

$$g(x) = 0. \tag{4.13}$$

If we use the Newton method to solve equations (4.13) its iteration will be based on the relation

$$g_k + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

since we assume that the next point should satisfy $g_{k+1} = 0$. This is exactly the main step of Algorithm 4.2. The Newton scheme is applied with great success in many optimization algorithms, most of all in several SQP (Sequential Quadratic Programming) algorithms for problems with equality constraints. However, its use in nonlinear conjugate gradient algorithms is not straightforward. In this section we consider the case when exact Hessian matrices are used at every iteration of a conjugate gradient algorithm. Following Chap. 1 the preconditioned nonlinear conjugate gradient algorithm could be stated as follows.

Algorithm 4.3. (The preconditioned conjugate gradient algorithm with exact Hessian)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, calculate

$$d_k = -\left[\nabla^2 f(x_k)\right]^{-1} g_k$$

and set $k = 1$.

2. Find α_k by minimizing f on the line $\mathscr{L}_k = \{x \in \mathscr{R}^n : x = x_k + \alpha d_k,\ \alpha \in (0,\infty)\}$. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1} = \| = 0$ then STOP. Otherwise calculate

$$d_{k+1} = -\left[\nabla^2 f(x_{k+1})\right]^{-1} g_{k+1} + \hat{\beta}_{k+1} d_k \tag{4.14}$$

$$\hat{\beta}_{k+1} = \frac{y_k^T \left[\nabla^2 f(x_{k+1})\right]^{-1} g_{k+1}}{y_k^T d_k} \tag{4.15}$$

Increase k by one and go to Step 2.

The justification for formulae (4.14)–(4.15) will be given in the section. Here, we are interested in the rate of convergence of Algorithm 4.3. As the formulae suggest we consider the scheme which combines both the Newton and a conjugate gradient methods. The question we ask is wether the scheme provides improvement over the Newton method. To present the main results concerning the scheme we need a new notation – $h(x) = O(x^p)$ if there exists constant $C < \infty$ such that $\|h(x)\| \leqslant C\|x\|^p$ for small values of $\|x\|$.

The theorem which establishes the rate of convergence of Algorithm 4.3 refers to the technical lemma presented in [4].

Lemma 4.1. *Suppose that assumptions of* Theorem 4.3 *hold. Then, if* $h_k = x_k - \bar{x}$, *we have*

$$s_k = O(\|h_k\|) = O(\|g_k\|) \tag{4.16}$$

$$d_k = O(\|h_k\|) \tag{4.17}$$

$$\alpha_k = 1 + O(\|h_{k-1}\|). \tag{4.18}$$

Proof. We follow the proof given in [4]. Notice that

$$\begin{aligned} s_k &= x_{k+1} - x_k = x_{k+1} - \bar{x} + \bar{x} - x_k \\ &= h_{k+1} - h_k. \end{aligned} \tag{4.19}$$

Furthermore, using the mean value theorem, since $g(\bar{x}) = 0$, we can write

$$f(x_k) = f(\bar{x}) + \frac{1}{2} h_k^T \nabla^2 f(\xi) h_k, \tag{4.20}$$

where $\xi = \bar{x} + t h_k$, $t \in [0,1]$. Next, using assumption (4.10) and (4.20) we come to the relations

$$\frac{2}{c_1}\left(f(x_k) - f(\bar{x})\right) \leqslant \|h_k\|^2 \leqslant \frac{2}{c_2}\left(f(x_k) - f(\bar{x})\right) \tag{4.21}$$

for some positive constants c_1, c_2. Noting that $f(x_{k+1}) < f(x_k)$, from (4.21), we also obtain

$$\begin{aligned}\|h_{k+1}\|^2 &\leqslant \frac{2}{c_2}\left(f(x_{k+1}) - f(\bar{x})\right) \leqslant \frac{2}{c_1}\left(f(x_k) - f(\bar{x})\right)\\ &\leqslant \|h_k\|^2\end{aligned}$$

and

$$\|h_{k+1}\| \leqslant c_3\|h_k\| \tag{4.22}$$

for some positive c_3, which from (4.19), implies that

$$\|s_k\| \leqslant (1+c_3)\|h_k\| \tag{4.23}$$

which proves the first part of (4.16). The second part of (4.16) follows from (4.10) and

$$g_k = \int_0^1 \nabla^2 f\left(\bar{x} + t(x_k - \bar{x})\right) dt (x_k - \bar{x}) \tag{4.24}$$

which is valid since $g(\bar{x}) = 0$.

In order to prove (4.17) notice first that, due to our assumption $g_k^T d_{k-1} = 0$, we have

$$\begin{aligned}y_{k-1}^T d_{k-1} &= \left(g_k - g_{k-1}\right)^T d_{k-1} = -g_{k-1}^T d_{k-1}\\ &= g_{k-1}^T \left[\nabla^2 f(x_{k-1})\right]^{-1} g_{k-1}.\end{aligned} \tag{4.25}$$

But (4.10) holds thus we can write

$$g_{k-1}^T \left[\nabla^2 f(x_{k-1})\right]^{-1} g_{k-1} \geqslant \frac{1}{c_2}\|g_{k-1}\|^2 \tag{4.26}$$

and by P_k we denote $\hat{\beta}_k d_{k-1}$:

$$P_k = \frac{y_{k-1}^T \left[\nabla^2 f(x_k)\right]^{-1} g_k}{y_{k-1}^T d_{k-1}} d_{k-1}.$$

The crucial step in the proof is the claim by Al-Baali and Fletcher (cf. (33) in [4]) that

$$P_k = O(\|h_k\|^2),$$

then, if we assume that (4.17) is true for $k-1$, according to (4.16) and (4.26) the following holds

$$d_k = -\left[\nabla^2 f(x_k)\right]^{-1} g_k + O(\|h_k\|^2) \tag{4.27}$$

which according to (4.16) and (4.24) gives

$$d_k = O(\|h_k\|)$$

and since $d_1 = O(\|g_1\|) = O(\|h_1\|)$ (4.17) is proved by induction.

From (4.27) we have

$$d_k^T \nabla^2 f(x_k) d_k = -g_k^T d_k + O(\|h_k\|^3), \tag{4.28}$$

the Lipschitz continuity of $\nabla^2 f$ and the Taylor's expansion theorem justify the relation

$$\begin{aligned} g_{k+1} &= g_k + \int_0^1 \nabla^2 f(x_k + ts_k) dt s_k \\ &= g_k + \int_0^1 \left[\nabla^2 f(x_k + ts_k) - \nabla^2 f(x_k)\right] dt s_k + \int_0^1 \nabla^2 f(x_k) dt s_k \\ &= g_k + \nabla^2 f(x_k) s_k + O(\|s_k\|^2). \end{aligned} \tag{4.29}$$

Equations (4.28)–(4.29), (4.16)–(4.17) and $g_{k+1}^T d_k = 0$ allow us to write

$$d_k^T \nabla^2 f(x_k) d_k = -d_k^T g_k + O(\|h_k\|^3) \tag{4.30}$$

$$\alpha_k d_k^T \nabla^2 f(x_k) d_k = -d_k^T g_k + O(\|h_k\|^3) \tag{4.31}$$

which lead to

$$\alpha_k = 1 + \frac{O(\|h_k\|^3)}{d_k^T \nabla^2 f(x_k) d_k}$$

which is equivalent to (4.18) if we take into account (4.10) and (4.22). □

Lemma 4.1 is needed to establish the *R*-order of convergence of Algorithm 4.3.

Theorem 4.5. *Suppose that assumptions of* Lemma 4.1 *are satisfied. Then* Algorithm 4.3 *converges to the minimum point* $\bar{x}$ *and has R-order at least* $p = \frac{1}{2}\left(1 + \sqrt{5}\right)$.

Proof. Al-Baali and Fletcher show that using arguments similar to those which led to (4.27) and (4.29) one can come to the relations

$$g_k = -\nabla^2 f(\bar{x}) h_k + O(\|h_k\|^2)$$

which together with

$$d_k = -\left[\nabla^2 f(\bar{x})\right]^{-1} g_k + O(\|h_k\|^2).$$

imply that

$$d_k = -h_k + O(\|h_k\|^2).$$

Since

$$d_k = \frac{s_k}{\alpha_k} = \frac{h_{k+1} - h_k}{\alpha_k},$$

from (4.22) and (4.18), we come to the relation

$$\|h_{k+1}\| \leqslant c\|h_k\|\|h_{k-1}\|, \tag{4.32}$$

where c is some positive constant.

Setting $\eta_k = c\|h_k\|$ we can write (4.32) as

$$\eta_{k+1} \leqslant \eta_k \eta_{k-1}. \tag{4.33}$$

The rest is a standard R-order rate of convergence analysis. Assuming that η_k is sufficiently close to zero, or in other words that x_k is sufficiently close to $\bar{x}$, we have η_k, $\eta_{k-1} < 1$. Then (4.33) forces $\{\eta_k\}$ to converge to zero, or $\{x_k\}$ to converge to $\bar{x}$. Applying part (i) of the thesis of Theorem 4.2 we come to the conclusion that $\{\eta_k\}$ has R-order of convergence at least $p = \frac{1}{2}\left(1 + \sqrt{5}\right)$ which is the only positive root of the equation

$$t^2 - t - 1 = 0.$$

□

The theorem does not state that the sequence $\{x_k\}$ converges with the exact R-order p (not to mention the exact Q-order p). However, it is also shown in [4] that there exists a sequence $\{b_k\}$ of positive numbers such that $b_k \to 0$ and

$$\|x_k - \bar{x}\| \leqslant cb_k \tag{4.34}$$

for some constant c. Furthermore, there are constants $0 < \gamma_1 \leqslant \gamma_2$ such that

$$\gamma_1 b_k^p \leqslant b_{k+1} \leqslant \gamma_2 b_k^p. \tag{4.35}$$

From Theorem 4.1 we know that $\{b_k\}$ converges with the exact Q and R-order p. However, this does not imply any Q-order rate of convergence of the sequence $\{x_k\}$. The major drawback of the concept of Q-order rate of convergence is that it is not consistent with natural ordering of sequences. Suppose that $\{\alpha_k\}$, $\{\beta_k\}$ are two sequences of positive numbers converging to zero and such that

$$\alpha_k \leqslant \beta_k, \;\; k = 0, 1, \ldots.$$

If $\{\beta_k\}$ converges with Q-order at least p (the exact Q-order p), then $\{\alpha_k\}$ does not have to converge with Q-order at least p (the exact Q-order p). However, the implication holds for R-order of convergence. In fact this appears to be one of the motivations for introducing the R-order [163].

Summing up (4.34)–(4.35) do not imply that $\{x_k\}$ converges to $\bar{x}$ with Q-order greater then one. Al-Baali and Fletcher construct an example on which numerical performance of Algorithm 4.3 suggest the Q-order $p \approx 1.618$. Furthermore, they show on the same example that Algorithm 4.3 with

$$\hat{\beta}_{k+1} = \frac{g_{k+1}^T \left[\nabla^2 f(x_{k+1})\right]^{-1} g_{k+1}}{g_k^T \left[\nabla^2 f(x_k)\right]^{-1} g_k}$$

(which corresponds to the Fletcher–Reeves version of Algorithm 4.3) may converge Q-linearly.

The convergence analysis of Algorithm 4.3 presented in [4] suggests that we must be very careful with the modification of the Newton scheme. We have a similar, negative, outcome when we try to restrict the step-size α_k so near a solution it is not equal to one (e.g. [146]).

So does the analysis imply that a preconditioned nonlinear conjugate gradient algorithm is doomed for failure? It does not have to be the case. We can look at a preconditioned nonlinear conjugate gradient algorithm from the perspective of the quadratic case. We assume that f, in the neighborhood of a solution, can be approximated by a strongly convex function and a preconditioned nonlinear conjugate gradient algorithm mimics a linear conjugate gradient algorithm applied to a strongly convex quadratic function. This means that the preconditioner is chosen only once and then the preconditioned conjugate gradient iterations are performed with the fixed preconditioning matrix. Since we do not know in advance whether the quadratic approximation is adequate we change a preconditioner every fixed number of iterations. Such constructed preconditioned nonlinear conjugate gradient algorithm is considered in the subsequent sections.

4.4 Generic Preconditioned Conjugate Gradient Algorithm

Using scaling matrix in a conjugate gradient algorithm is equivalent to solving a problem in the transformed space. Consider a linear transformation D which maps the space of $\hat{x} \in \mathscr{R}^n$ into the original space of decision variables $x \in \mathscr{R}^n$:

$$D: \ \hat{x} \in \mathscr{R}^n \longrightarrow x \in \mathscr{R}^n,$$

where D is nonsingular. After transformation our optimization problem becomes

$$\min_{\hat{x} \in \mathscr{R}^n} \left[\hat{f}(\hat{x}) = f(D\hat{x})\right].$$

Function $\hat{f}$ has gradients and the Hessian matrices which can be calculated from corresponding gradients and the Hessian matrices of function f:

$$\nabla \hat{f}(\hat{x}) = D^T \nabla f(x) \tag{4.36}$$

$$\nabla^2 \hat{f}(\hat{x}) = DD^T \nabla^2 f(x). \tag{4.37}$$

Conjugate gradient algorithm in the space of $\hat{x}$ variables has the following direction rule

$$\hat{d}_{k+1} = -\hat{g}_{k+1} + \hat{\beta}_{k+1}\hat{d}_k \tag{4.38}$$

with

$$\hat{\beta}_{k+1} = \frac{(\hat{g}_{k+1} - \hat{g}_k)^T \hat{g}_k}{\|\hat{g}_k\|^2} \tag{4.39}$$

$$\hat{\beta}_{k+1} = \frac{\|\hat{g}_{k+1}\|^2}{\|\hat{g}_k\|^2} \tag{4.40}$$

$$\hat{\beta}_{k+1} = \frac{(\hat{g}_{k+1} - \hat{g}_k)^T \hat{g}_k}{\hat{d}_k^T (\hat{g}_{k+1} - \hat{g}_k)} \tag{4.41}$$

for the Polak–Ribière, the Fletcher–Reeves and the Hestenes–Stiefel versions, respectively.

Using (4.36) and taking into account that $d_k = D_k\hat{d}_k$ (assuming that D changes from one iteration to another) we can write (4.38) as

$$d_{k+1} = -D_{k+1}D_{k+1}^T g_{k+1} + \hat{\beta}_{k+1}d_k$$

(provided that $D_{k+1}D_k^{-1} = I$!).

Introducing matrix H_k defined by

$$H_{k+1}^{-1} = D_{k+1}D_{k+1}^T$$

we arrive at the most popular form of a direction rule for a preconditioned conjugate gradient algorithm

$$d_{k+1} = -H_{k+1}g_{k+1} + \hat{\beta}_{k+1}d_k. \tag{4.42}$$

Two important observations can be made about formula (4.42). First, if we take

$$H_{k+1} = \left[\nabla^2 f(x_{k+1})\right]^{-1}$$

and neglect the term with d_k we have the Newton direction which is the best scaled direction. Secondly, the coefficient $\hat{\beta}_{k+1}$ in (4.39)–(4.41) is taken directly from a conjugate gradient direction rule in the space of $\hat{x}$ variables. This means that it should be calculated by

$$\hat{\beta}_{k+1} = \frac{(g_{k+1} - g_k)^T H g_k}{g_k^T H g_k} \tag{4.43}$$

$$\hat{\beta}_{k+1} = \frac{g_{k+1}^T H g_{k+1}}{g_k^T H g_k} \tag{4.44}$$

$$\hat{\beta}_{k+1} = \frac{(g_{k+1} - g_k)^T H g_k}{d_k^T (g_{k+1} - g_k)} \tag{4.45}$$

if we assume that $H_{k+1} = H_k = H$. Notice that validity of (4.45) follows from nonsingularity of H which is implied by that of D.

Our generic preconditioned conjugate gradient algorithm is stated as follows.

Algorithm 4.4. (The generic preconditioned conjugate gradient algorithm)

Parameters: sequence of symmetric positive definite matrices $\{H_k\}$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -H_1 g_1$$

   and $k = 1$.
2. Find α_k using some step-length selection procedure which guarantees that the next direction is a direction of descent. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP. Otherwise calculate d_{k+1} according to

$$d_{k+1} = -H_{k+1} g_{k+1} + \hat{\beta}_{k+1} d_k.$$

   Increase k by one and go to Step 2.

We do not specify in this chapter what conditions should be imposed on α_k in order d_{k+1} to be a direction of descent (they are presented while the method of shortest residuals is discussed in Chap. 8). However, notice that if α_k is the result of the exact directional minimization then, since H_{k+1} is positive definite, d_{k+1} is a direction of descent.

The other crucial point concerning our generic algorithm is the choice of the sequence $\{H_k\}$. In the next section we show that the memoryless quasi-Newton algorithm by Shanno is in fact a representative of the generic algorithm in which the matrices H_k are specified. Here, following suggestions by Powell [170] we introduce scaling matrices derived from a quasi-Newton approximations to the Hessian matrix.

We start from the BFGS updating formula for the approximation of the Hessian matrix

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}.$$

Powell assumes that B_k^{-1} can be factorized in the following way

$$B_k^{-1} = D_k D_k^T \tag{4.46}$$

where D_k is nonsingular. Then, he refers to the multiplicative form of the BFGS updating formula

$$B_{k+1}^{-1} = \left(I - s_k z_k^T\right) B_k^{-1} \left(I - z_k s_k^T\right) \tag{4.47}$$

with

$$z_k = \frac{B_k s_k}{\sqrt{(s_k^T y_k)(s_k^T B_k s_k)}} + \frac{y_k}{s_k^T y_k}. \tag{4.48}$$

This formula was introduced by Brodlie, Gourlay and Greenstadt [14]. Taking into account (4.46)–(4.47) Powell provides the update for D_k

$$D_{k+1} = \left(I - s_k z_k^T\right) D_k. \tag{4.49}$$

The major drawback of the update is that we need the product $B_k s_k$ which requires storing also the matrix B_k. Powell claims that in the usual case s_k is a multiple of $B_k^{-1} g_k$ so the product $B_k s_k$ can be evaluated without the knowledge of B_k. Furthermore, Powell suggests using, instead of the matrix D_{k+1} defined by (4.49), the matrix $\hat{D}_{k+1}$ which is the modification of D_{k+1} obtained by postmultiplying it by the orthogonal matrix:

$$\hat{D}_{k+1} = D_{k+1} \Omega_{k+1}.$$

Since Ω_{k+1} is orthogonal the matrix $\hat{D}_{k+1}$ is also a factor of the matrix B_{k+1}^{-1}:

$$\begin{aligned} \hat{D}_{k+1} \hat{D}_{k+1}^T &= D_{k+1} \Omega_{k+1} \Omega_{k+1}^T D_{k+1}^T \\ &= D_{k+1} D_{k+1}^T = B_{k+1}^{-1}. \end{aligned} \tag{4.50}$$

Powell shows that within this approach the product $B_k s_k$ does not have to be evaluated. Moreover, it is characterized by better numerical properties.

From (4.50) it also follows that the matrix $\hat{D}_{k+1}$ preserves another important property of the matrix D_{k+1} which is that their columns are conjugate directions with respect to the matrix B_{k+1}

$$\hat{D}_{k+1}^T B_{k+1} \hat{D}_{k+1} = I.$$

Powell's quasi-Newton method is further analyzed in Chap. 12 where it is compared to other methods which use column scaling to increase the efficiency of variable metric methods.

4.5 Quasi-Newton-like Variable Storage Conjugate Gradient Algorithm

The first systematically developed preconditioned conjugate gradient algorithm was proposed by Buckley and LeNir [20] (see also [18] and [21]). The inspiration for them was the memoryless quasi-Newton algorithm by Shanno. We remind that the

Shanno algorithm has the direction formula

$$d_{k+1} = -H_{k+1} g_{k+1} \tag{4.51}$$

$$H_{k+1} = H_{r+1} - \left(\frac{H_{r+1} y_k s_k^T + s_k y_k^T H_{r+1}}{s_k^T y_k} \right) + \\ + \left(1 + \frac{y_k^T H_{r+1} y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k}, \tag{4.52}$$

where H_{r+1} is defined by

$$H_{r+1} = \gamma_r \left(I - \frac{s_r y_r^T + y_r s_r^T}{s_r^T y_r} + \frac{y_r^T y_r}{(s_r^T y_r)^2} s_r s_r^T \right) + \frac{s_r s_r^T}{s_r^T y_r}. \tag{4.53}$$

Under the exactness of line searches the direction rule is that of a preconditioned conjugate gradient algorithm with the scaling matrix H_r:

$$d_{k+1} = -H_r g_{k+1} + \frac{y_k^T H_r y_k}{s_k^T y_k} d_k.$$

Buckley and LeNir extends the Shanno algorithm by using the 'padding' procedure applied by Shanno which resulted in formulae (4.52)–(4.53). Before we go again through the 'padding' procedure to obtain the Buckley–LeNir method we remark that its effect on the Shanno algorithm would be the rules (4.51)–(4.53) in which (4.53) is substituted by

$$H_{r+1} = H - \frac{s_r y_r^T H + H y_r s_r^T}{s_r^T y_r} + \frac{y_r^T H y_r}{(s_r^T y_r)^2} s_r p_r^T + \frac{s_r s_r^T}{s_r^T y_r} \tag{4.54}$$

thus the diagonal matrix $\gamma_r I$ is replaced by a positive definite matrix H.

We start with the preconditioned Hestenes–Stiefel algorithm

$$d_{k+1} = -H g_{k+1} + \beta_{k+1} d_k$$

$$\beta_{k+1} = \frac{y_k^T H g_{k+1}}{d_k^T y_k}$$

which can be expressed as

$$d_{k+1} = -\left(H - \frac{s_k y_k^T H}{s_k^T y_k} \right) g_{k+1} \\ = -Q_{k+1} g_{k+1}.$$

Matrix Q_{k+1} is not positive definite, however using Shanno's 'padding' we can make this matrix symmetric and positive definite

$$H_{k+1} = H - \frac{s_k y_k^T H + H y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{y_k^T H y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k} \tag{4.55}$$

by adding terms with s_k which vanish if exact line searches are used. It can be shown that H_{k+1} satisfies the secant equation, i.e. $H_{k+1}y_k = s_k$. Notice that the matrix H_{k+1} is the BFGS update of the matrix H. We denote by U the function of getting H_{k+1} from H, s_k and y_k by the update procedure (4.55):

$$H_{k+1} = U(H, y_k, s_k). \tag{4.56}$$

Matrix H in (4.56) can be any symmetric positive definite matrix. It seems natural to use as H the matrix which is calculated by m BFGS updates of the identity matrix based on m successive pairs (s_{r-m}, y_{r-m}), (s_{r-m+1}, y_{r-ma+1}), ...,(s_r, y_r). This means that before we take a preconditioned conjugate gradient step we perform m steps of the BFGS method

$$H_{r+i} = U(H_{r+i-1}, y_{r+i-1}, s_{r+i-1}),\ i = 1, \ldots, m$$

with H_r being some initial matrix assumed at the rth restarting iteration. From the $(r+m+1)$th iteration the scaling matrix is fixed as H_{r+m} and the Buckley–LeNir algorithm performs preconditioned conjugate gradient iterations, i.e.

$$H_{r+i} = U(H_{r+m}, y_{r+i-1}, s_{r+i-1}),\ i = m+1, m+2, \ldots, m+n.$$

For every iteration we calculate d_{k+1} according to

$$d_{k+1} = -H_{k+1}g_{k+1}$$

so we need an efficient procedure for calculating the matrix-vector product $H_{k+1}v$. Buckley and LeNir observe that

$$\begin{aligned} H_{k+1}v &= H_q v - \left[\frac{u_k^T v}{\zeta_k} - \left(1 + \frac{\nu_k}{\zeta_k}\right)\frac{s_k^T v}{\zeta_k}\right] s_k - \frac{s_k^T v}{\zeta_k} u_k \\ &= H_q v - \sigma_k p_k - \varsigma_k u_k \end{aligned} \tag{4.57}$$

with $\nu_k = y_k^T u$, $\zeta_k = s_k^T y_k$ and $u_k = H_q y_k$. Here, q is either equal to k $(r < k < r+m)$, or $r+m$. When we compute $H_{k+1}g_{k+1}$ we need to evaluate $H_q g_{k+1}$ and $H_q y_k$ which can be accomplished efficiently using (4.57) recursively:

$$\begin{aligned} H_q v &= H_1^0 v - \sum_{i=r}^{q} \left\{ \left[\frac{u_i^T v}{\zeta_i} - \left(1 + \frac{\nu_i}{\zeta_i}\right)\frac{s_i^T v}{\zeta_i}\right] s_i - \frac{s_i^T v}{\zeta_i} u_i \right\} \\ &= H_1^0 v - \sum_{i=r}^{q} (\sigma_i s_i + \varsigma_i u_i). \end{aligned}$$

This implies that the evaluation requires $4qn$ operations if we ignore the cost of calculating $H_1^0 v$. Therefore, the total number of operations to calculate $H_{k+1}g_{k+1}$ is equal to $(8q+7)n$ when $q = r+m$, or to $(8(k-r)+7)n$ when $q = k$ $(r < k < r+m)$, if H_1^0 is a diagonal matrix. In both cases the cost is not excessive for $m \ll n$.

In order to avoid excessive computational cost of evaluating $H_1^0 v$ we have to take a diagonal matrix as H_1^0. Buckley and LeNir recommend the identity matrix scaled by the coefficient γ_r defined as in SSVM methods by Oren and Spedicato [152] and used by Shanno:

$$\gamma_r = \frac{s_r^T y_r}{y_r^T y_r}. \tag{4.58}$$

More general diagonal matrices taken as initial matrices at restarting iterations will be discussed later. Notice that if

$$H_r = \gamma_r I, \tag{4.59}$$

then the updating formula for H_{k+1}, $r \leqslant k \leqslant r+m-1$ is

$$H_{k+1} = \left(H_k - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k} + \frac{y_k^T H_k y_k}{s_k^T y_k} \frac{s_k s_k^T}{s_k^T y_k} \right) \gamma_r + \frac{s_k s_k^T}{s_k^T y_k}$$

which is in fact the Oren–Spedicato scaled version of the BFGS update.

It is arguable whether the initial matrix at a restarting iteration should be defined by (4.59). In [71] Fletcher proposes that the identity matrix is scaled by the coefficient κ_r, i.e.

$$H_r = \kappa_r I \tag{4.60}$$

where

$$\kappa_r = \frac{2(f(x_k) - f(x_{k-1}))}{s_k^T g_k}. \tag{4.61}$$

The initialization of quasi-Newton iterations by (4.60)–(4.61) was tested by Shanno and by Buckley and LeNir with results comparable to those obtained by using (4.58)–(4.59). Notice that both γ_r and κ_r are positive if the Wolfe curvature condition is satisfied in a step-length selection procedure. The condition $s_{r+i-1}^T y_{r+i-1} > 0$ is also needed to guarantee that every H_{r+i} is positive definite – this is a consequence of the fact that H_{r+i} is the BFGS update of H_{r+i-1}.

The Buckley–LeNir algorithm consists of two phases. In the first phase we build a scaling matrix H_{r+m} which is then used in a preconditioned conjugate gradient algorithm. Of course we could use any other positive definite matrix in place of H_{r+m} eliminating quasi-Newton iterations from the algorithm. For example, we could define matrices H_{k+1} by

$$H_{k+1} = U(H_{r+1}, y_k, s_k)$$

for $r+1 < k \leqslant r+n$, with H_{r+1} determined according to (4.54). This would be the extension of the memoryless quasi-Newton algorithm by Shanno where the

modification of the Shanno algorithm relies on taking a general positive definite matrix H at the rth restarting iteration.

The Buckley–LeNir algorithm can be stated as follows.

Algorithm 4.5. (The Buckley–LeNir algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -g_1$$

$r = 1$ and $k = 1$.

2. Find α_k that satisfies the Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP. Otherwise calculate d_{k+1} according to

$$\begin{aligned} H_{k+1} &= U(\gamma_k I, y_k, s_k) \\ d_{k+1} &= -H_{k+1} g_{k+1}, \end{aligned}$$

if $k = r + n$, or

$$g_{k+1}^T g_k \geqslant \nu \|g_k\|^2 \tag{4.62}$$

where $\nu \in (0,1)$. Set $r = k+1$ and go to Step 4.
If $r+1 < k \leqslant r+m$ then calculate d_{k+1} according to

$$H_{k+1} = U(H_k, y_k, s_k) \tag{4.63}$$

$$d_{k+1} = -H_{k+1} g_{k+1} \tag{4.64}$$

and go to Step (4).
If $r+m < k \leqslant k+n-1$ then determine d_{k+1} according to

$$H_{k+1} = U(H_{r+m}, y_k, s_k) \tag{4.65}$$

$$d_{k+1} = -H_{k+1} g_{k+1}. \tag{4.66}$$

4. Increase k by one and go to Step 2.

Buckley and LeNir do not prove global convergence of their method. However, the method possesses all ingredients to be a viable procedure. The direction d_k is always a direction of descent since the matrix H_k is positive definite. Furthermore, the matrices H_k are generated by the BFGS updates which appear in the most successful quasi-Newton methods and although there is no global convergence result when applied to nonconvex problems they are robust algorithms. In addition, when $m = 1$ the Buckley–LenNr algorithm is equivalent to the memoryless quasi-Newton algorithm by Shanno which is globally convergent under some mild assumptions.

One of the most important features of the Buckley–LeNir algorithm is that restarts are taking every n iterations. They are needed in order to have constant

scaling matrices H_{r+m} for $(n-m)$ iterations. If they used different scaling matrix at each iteration then they would, in fact, resort to a variable storage quasi-Newton method which will be discussed in the next chapter. Restarts every n iterations ensure global convergence of the Buckley–LeNir algorithm. The proof of the global convergence relies on the analysis of the basic Shanno algorithm.

Theorem 4.6. *Suppose that $\{x_k\}$ is generated by the Buckley–LeNir method (*Algorithm 4.5*) applied without checking the condition (4.62). Assume also that there exist positive numbers m and M such that*

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2$$

for all x and z. Then $\{x_k\}$ converges to the minimizer of f.

Proof. Denote by k_r, $r = 1,2,\ldots$ indices of these iterations at which restarts take place. According to the description of the method there is at least one restart every n iterations. At these iterations we have

$$\begin{aligned} H_{k_r+1} &= U(\gamma_{k_r} I, y_{k_r}, s_{k_r}) \\ d_{k_r+1} &= -H_{k_r+1} g_{k_r+1}. \end{aligned}$$

While proving Theorem 3.1 we showed that

$$\begin{aligned} \cos^2\theta_{k_r+1} &= \frac{4\lambda_{k_r+1}\Lambda_{k_r+1}}{\lambda_{k_r+1}+\Lambda_{k_r+1}} \\ &\geqslant \frac{m^2}{M} \end{aligned} \tag{4.67}$$

where λ_{k_r+1} and Λ_{k_r+1} are the smallest and the largest, respectively, eigenvalues of H_{k_r+1}.

Furthermore, we have

$$\begin{aligned} f(x_{k_r+2}) - f(x_{k_r+1}) &\leqslant \mu\alpha_{k_r+1} g_{k_r+1}^T d_{k_r+1} \\ &= -\mu\alpha_{k_r+1}\cos\theta_{k_r+1}\|d_{k_r+1}\|\|g_{k_r+1}\|. \end{aligned} \tag{4.68}$$

On the other hand, from (4.66) and the Wolfe curvature condition, we have

$$\begin{aligned} \alpha_{k_r+1}^2 M\|d_{k_r+1}\|^2 &= M\|s_{k_r+1}\|^2 \geqslant s_{k_r+1}^T y_{k_r+1} \\ &\geqslant (\eta-1) s_{k_r+1}^T y_{kr+1} \\ &= (\eta-1)\alpha_{k_r+1} g_{k_r+1}^T d_{k_r+1} \end{aligned}$$

which gives

$$\alpha_{k_r+1} \geqslant \frac{(1-\eta)\cos\theta_{k_r+1}\|g_{k_r+1}\|}{M\|d_{k_r+1}\|}.$$

This together with (4.68) imply that

$$f(x_{k_r+2}) - f(x_{k_r+1}) \leqslant -\frac{\mu(1-\eta)}{M}\cos^2\theta_{k_r+1}\|g_{k_r+1}\|^2. \tag{4.69}$$

Suppose now that

$$\liminf_{r\to\infty}\|g_{k_r+1}\| = \varepsilon > 0, \tag{4.70}$$

then, from (4.67) and (4.69) we come to the conclusion that f is unbounded which is impossible due to our assumption (4.66). Therefore, (4.70) is not true and since f has unique solution $\{x_k\}$ converges to its minimizer. □

Restarting every n iterations is crucial for proving global convergence of the Buckley–LeNir algorithm applied to convex functions. It appears that if we relax the convexity assumption then the Powell's restarts guarantee convergence to a stationary point. (Notice that the situation is parallel to that we have in the case of the memoryless quasi-Newton algorithm by Shanno.)

Theorem 4.7. *Suppose that $\{x_k\}$ is generated by the Bckley–LeNir algorithm* – Algorithm 4.5. *Assume also that there exists a positive number M such that*

$$z^T\nabla^2 f(x)z \leqslant M\|z\|^2$$

for all x and z. If

$$\lim_{k\to\infty}\|x_{k+1} - x_k\| = 0, \tag{4.71}$$

then

$$\liminf_{k\to\infty}\|g_k\| = 0.$$

Proof. The proof mimics the proof of Theorem 3.4. First we assume that there exists $\varepsilon > 0$ such that

$$\liminf_{k\to\infty}\|g_k\| = \varepsilon. \tag{4.72}$$

Under (4.71) and (4.72) we can show that there exists an index $\hat{k}$ such that for $k \geqslant \hat{k}$

$$\left|g_{k+1}^T g_k\right| \geqslant \nu\|g_k\|^2.$$

This means that for $k \geqslant \hat{k}$ d_{k+1} is determined by

$$\begin{aligned} H_{k+1} &= U(\gamma_k I, y_k, s_k) \\ d_{k+1} &= -H_{k+1}g_{k+1}, \end{aligned}$$

thus for k sufficiently large directions are calculated as in the basic Shanno algorithm. Therefore we can refer to the proof of Theorem 3.4 to come to the contradiction with (4.72). □

Buckley and LeNir presents in [20] results of several numerical tests obtained with their algorithms. As expected while m increases then the number of function evaluations decreases. However, it is not always the case that the BFGS method (the algorithm which does not perform conjugate gradient iterations and restarts, in which m in unbounded) is the best method. Their numerical experiments do not enable us to claim that the performance of the variable storage quasi-Newton algorithm improves with larger values of m. That observation is a little surprising since Buckley and LeNir use more accurate directional minimization at conjugate gradient iterations thus while m increases then also the number of quasi-Newton iterations increases and as the result the number of function evaluations decreases. The explanation probably lies in their another observation – the number of iterations has tendency to increase with m.

Although the number of function evaluations can decrease with the increase of m the computing time usually increases due to more elaborate numerical algebra involved. Buckley and LeNir claim that their algorithm is a family of methods parameterized by m which can be tuned to a specific problem guaranteeing the smallest overall computing time.

4.6 Scaling the Identity

Numerical tests show that the choice of $H_{r+1} = I$ may be unsuitable in variable storage quasi-Newton methods. Notice that in fact we use

$$H_{r+1} = U(\gamma_r I, y_r, s_r)$$

with the choice of γ_r supported by Oren–Spedicato analysis of self-scaling variable metric methods [152]. In this section we give some hints concerning other choices than $\gamma_r I$ as the initial matrix in quasi-Newton approximations.

First, notice that it is not possible to satisfy the secant equation $Hy = s$ with H of the form γI. However, we can try to do that in a given direction v. Projecting the secant equation in a direction v such that $v^T y \neq 0$ gives for γ the value

$$\gamma_v = \frac{s^T v}{y^T v}.$$

Gilbert and Lemaréchal consider vectors v belonging to $\mathcal{V} = \{v : v = \alpha y + \beta s, \alpha, \beta \in (0, \infty)\}$. If $v \in \mathcal{V}$ then $v^T y$ and $s^T v$ are positive and so γ_v is also positive. (This follows from the fact that $s^T y$ is positive under the Wolfe conditions.) Applying the Hölder inequality we can show that

$$\gamma' = \frac{s^T y}{y^T y} \leqslant \gamma_v \leqslant \frac{s^T s}{s^T y} = \gamma''. \tag{4.73}$$

There is strong theoretical justification for using either γ', or γ'' to scale the initial identity matrix. Following Gilbert and Lemaréchal we cite the most important properties of γ' and γ''

Lemma 4.2. *If γ' is defined by (4.73) then the following holds*

(i) γ' is the Rayleigh quotient of $\hat{H}$ in the direction y, i.e.

$$\gamma' = \frac{y^T \hat{H} y}{y^T y} \tag{4.74}$$

with

$$\hat{H} = \hat{B}^{-1} = \left(\int_0^1 \nabla^2 f(x + \xi s) d\xi \right)^{-1}. \tag{4.75}$$

(ii) γ' minimizes in $\gamma \in \mathscr{R}$ the norm

$$\|(\gamma I) y - s\|. \tag{4.76}$$

(iii) γ' minimizes for $\gamma > 0$ the condition number of $U(\gamma I, y, s)$.
(iv) γ' is the unique solution of the problem

$$\min_{\gamma \in \mathscr{R}} \min_{H \in L(y,s)} \|\gamma I - H\|_F, \tag{4.77}$$

where $\|\cdot\|_F$ is the Frobenius norm and

$$L(y,s) = \left\{ H \in \mathscr{R}^{n \times n} : \det(H) \neq 0 \text{ and } Hy = s \right\}.$$

Proof. *(i)*. Using the Taylor's expansion theorem we can write

$$y = \hat{B} s$$

where $\hat{B}$ is defined in (4.75). Substituting $s = \hat{B}^{-1} y = \hat{H} y$ in (4.73) we obtain (4.74).

(ii). The solution to the least squares problem (4.76) is given by the solution to the equation

$$y^T y \gamma = s^T y$$

which proves *(ii)*.

The proof of *(iii)* is given in [152] while the proof of *(iv)* is provided by Gilbert and Lemaréchal [77]. □

From the properties of γ' listed above the property *(ii)* seems to indicate that γ' is a proper scaling factor since the quasi-Newton iterations are based on the BFGS update.

The γ'' has also properties which suggest that it would be a good candidate for the scaling factor. They are listed in the next lemma which we cite without the proof following [77].

Lemma 4.3. *If γ'' is defined by (4.73) then the following holds*

(i) $1/\gamma''$ *is the Rayleigh quotient of* $\hat{B}$ *in the direction y, i.e.*

$$\frac{1}{\gamma''} = \frac{y^T \hat{B} y}{y^T y}$$

with $\hat{B}$ *defined by (4.75).*

(ii) γ'' *minimizes in* $\gamma \in \mathscr{R}$ *the norm*

$$\left\| y - s\left(\frac{1}{\gamma} I\right) \right\|.$$

(iii) γ' *minimizes for* $\gamma > 0$ *the condition number of* $\hat{U}(\gamma I, y, s)$*, where* $\hat{U}(\gamma I, y, s)$ *is the inverse DFP update:*

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}.$$

(iv) γ'' *is the unique solution of the problem*

$$\min_{\gamma \in \mathscr{R}, \gamma \neq 0} \min_{H \in L(y,s)} \left\| \frac{1}{\gamma} I - H \right\|_F.$$

Modest numerical tests performed by Gilbert and Lemaréchal point to γ' as a better scaling parameter. This seems due to the fact that γ'', which is larger according to (4.73) than γ', is too large and many function evaluations are to be performed to meet the Wolfe criteria.

Poor choice of the initial diagonal matrix can propagate in the next iterations and when $m \ll n$ it has strong impact on the overall performance of a method. We could remedy that by increasing m but that requires additionally two n – dimensional vectors for every increase of m. It seems that better cure is to start with a diagonal matrix which does not have all elements equal as it is in the case of the matrix γI.

Following Gilbert and Lemaréchal we consider diagonal matrices with respect to the scalar product $\langle \cdot, \cdot \rangle$ and the orthonormal basis $\{e_1, e_2, \ldots, e_n\}$. We say that D is diagonal if $e_i^T D e_j = 0$ for $i \neq j$. Diagonal elements $D^{(i)} = e_i^T D e_i$, $i = 1, \ldots, n$ define matrix D by

$$D = \sum_{i=1}^{n} D^{(i)} e_i e_i^T. \tag{4.78}$$

Observe that if we take as the basis the eigenvectors of $\hat{H}$ (given by (4.75)) then due to (4.78) for

$$D^{(i)} = \frac{s^T e_i}{y^T e_i}$$

we have $D = \hat{H}$.

In general the eigenvectors of $\hat{H}$ are not available and we approximate D by a sequence of updates which preserve the diagonal property. For example, we diagonalize the inverse BFGS update getting the formula

$$D_{k+1}^{(i)} = D_k^{(i)} + \left(\frac{1}{s_k^T y_k} + \frac{y_k^T D_k y_k}{\left(s_k^T y_k\right)^2} \right) \left(s_k^T e_i\right)^2 - \frac{2 D_k^{(i)} y_k^T e_i s_k^T e_i}{s_k^T y_k}$$

(cf. [77]).

Having D_k as the initial matrix we can modify the update formula in the first quasi-Newton iteration:

$$H_{r+n+1} = H_{k+1} = U\left(D_k, y_k, s_k\right).$$

Other diagonalized version of the quasi-Newton updates are also possible [77]. So far there is no enough numerical evidence to favor any of the updates against the others but they can potentially improve efficiency of variable storage quasi-Newton methods.

4.7 Notes

The concepts of Q-order and R-order of convergence are introduced to present the result of Al-Baali and Fletcher on the convergence of the preconditioned conjugate gradient algorithm with the exact Hessian matrices. These orders of convergence are discussed to much extent in [153] – we follow the paper by Potra [163].

Buckley–LeNir algorithm was implemented and is available as the 630 algorithm in ACM web site.

Powell's quasi-Newton method with column scaling described in Sect. 4.4 has many successors which are presented in Chap. 12. Among them is the limited memory reduced-Hessian quasi-Newton method which can be regarded as the main competitor to variable storage algorithms outlined in this chapter.

Chapter 5
Limited Memory Quasi-Newton Algorithms

5.1 Introduction

The memoryless quasi-Newton method stops short in creating an efficient compromise between a robust conjugate gradient algorithm and more efficient quasi-Newton method which uses limited storage.

The memoryless quasi-Newton method is based on the direction

$$d_{k+1} = -H_{k+1} g_{k+1}$$

with

$$H_{k+1} = I - \frac{y_k y_k^T}{y_k^T y_k} + \frac{s_k s_k^T}{s_k^T y_k} + \left(y_k^T y_k\right) v_k v_k^T \tag{5.1}$$

and

$$v_k = \frac{s_k}{s_k^T y_k} - \frac{y_k}{y_k^T y_k}. \tag{5.2}$$

The BFGS update formula guarantees that H_{k+1} is positive definite provided that H_k is positive definite and the curvature condition holds:

$$s_k^T y_k > 0.$$

Equivalently (5.1) can be stated as

$$H_{k+1} = \left(I - \frac{s_k y_k}{y_k^T s_k}\right)\left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

Introducing

$$V_k = I - \rho_k y_k s_k^T \tag{5.3}$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

$$\rho_k = \frac{1}{y_k^T s_k} \tag{5.4}$$

it can be further transformed to

$$H_{k+1} = V_k^T V_k + \rho_k s_k s_k^T. \tag{5.5}$$

Formula (5.5) can be regarded as a special case of the BFGS update in which the matrix H_k is substituted by the identity matrix. Therefore, the BFGS update rule can be written as

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T. \tag{5.6}$$

As we will see (5.6) leads to an efficient calculation of the product $H_k g_k$ if the matrices H_k are not stored but are built recursively according to formula (5.6).

The inverse Hessian approximation H_k is generally a dense matrix thus neither its storing or using in formulating the product $H_k g_k$ is prohibitive with respect to required storage and with respect to computing time. Instead, a certain number (say m) of the pairs (s_l, y_l) is kept in order to compute the matrix H_k at the kth iteration using (5.6) with some initial matrix H_k^0. In the next iteration the oldest pair (s_l, y_l) is replaced by the new pair (s_k, y_k).

The calculation of H_k can be expressed by the formula

$$\begin{aligned} H_k = {} & \left(V_{k-1}^T \cdots V_{k-m}^T\right) H_k^0 \left(V_{k-m} \cdots V_{k-1}\right) + \\ & \rho_{k-m} \left(V_{k-1}^T \cdots V_{k-m+1}^T\right) s_{k-m} s_{k-m}^T \left(V_{k-m+1} \cdots V_{k-1}\right) + \ldots + \\ & \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{5.7}$$

Taking into account that representation of H_k Nocedal [144] (following Matthias and Strang recursion for the standard BFGS update – cf. [129]) developed the recursive formula for calculating $H_k g_k$.

Algorithm 5.1. (The two-loop recursion)

Data: $(s_{k-m}, y_{k-m}), \ldots, (s_{k-1}, y_{k-1})$, H_k^0, g_k.

1. Substitute g_k for q. Set $i = k-1$.
2. Calculate ρ_i according to (5.4) and

$$\begin{aligned} \alpha_i &= \rho_i s_i^T q \\ \theta_i &= q - \alpha_i y_i. \end{aligned}$$

   Substitute θ_i for q. Decrease i by one.
3. If i is greater or equal to $k-m$ go to Step 2.
4. Substitute $H_k^0 q$ for r. Set $i = k-m$.

5. Calculate

$$\beta = \rho_i y_i^T r$$
$$\vartheta = r + s_i(\alpha_i - \beta).$$

Substitute ϑ for r. Increase i by one.

6. If i is less or equal to $k-1$ go to Step 5.

The cost of calculating $H_k g_k$ results from $4mn$ multiplications needed to perform two loops and from creating vector r in Step 4. If H_k^0 is a diagonal matrix then we need another n multiplications. Due to the second ingredient of the cost usually the diagonal matrix H_k^0 is chosen and the most widely used choice is

$$H_k^0 = \sigma_{k-1} I.$$

However, non-diagonal choices of H_k^0 are also possible since that would only increase the cost of the operation in Step 4. For example, we could use as H_k^0 some sparse approximation B_k^0 to the Hessian matrix and if the Cholesky factor L_k of the matrix is also sparse then the cost of solving the equations

$$L_k L_k^T r = q.$$

would not be prohibitive in large-scale optimization.

The two-loop recursion algorithm is superior to a straightforward approach of calculating $H_k g_k$. The inverse BFGS formula can be written as

$$H_{k+1} = H_k + a_k a_k^T - b_k b_k^T + c_k c_k^T$$

where

$$a_k = \frac{s_k}{\sqrt{s_k^T y_k}}$$
$$b_k = \frac{H_k y_k}{\sqrt{y_k H_k y_k}}$$
$$c_k = s_k \sqrt{\tau_k} - \frac{H_k y_k}{\sqrt{\tau_k}}$$

and

$$\tau_k = \frac{y_k^T H_k y_k}{p_k^T y_k}.$$

In a typical iteration, the matrix H_k is obtained by updating the initial matrix H_k^0 m times using the most recent pairs $(s_{k-m}, y_{k-m}), (s_{k-m+1}, y_{k-m+1}), \ldots, (s_{k-1}, y_{k-1})$.

Therefore, H_k can be written as

$$H_k = H_k^0 + \sum_{i=k-m}^{k-1} \left(a_i a_i^T - b_i b_i^T + c_i c_i^T \right). \tag{5.8}$$

The vectors a_i, b_i and c_i can be computed by the following procedure.

Algorithm 5.2. (Unrolling the inverse BFGS formula)

Data: $(s_{k-m}, y_{k-m}), \ldots, (s_{k-1}, y_{k-1})$, H_k^0.

1. Set $i = k - m$.
2. Calculate

$$a_i = \frac{s_i}{\sqrt{s_i^T y_i}}$$

$$b_i = H_k^0 y_i + \sum_{j=k-m}^{i-1} \left[\left(a_j^T s_i \right) a_j - \left(b_j^T s_i \right) b_j \right]$$

$$\xi_i = s_i^T b_i, \ \tau_i = \frac{\xi_i}{s_i^T y_i}.$$

   Substitute

$$\frac{b_i}{\xi_i} \ \text{and} \ \sqrt{\tau_i} s_i - \frac{b_i}{\sqrt{\tau_i}} \tag{5.9}$$

   for b_i and c_i, respectively. Increase i by one.
3. If i is less or equal to $k-1$ go to Step 2.

Provided that H_k^0 is a diagonal matrix the procedure requires $\frac{3}{2}m^2 n$ operations since the vectors b_i have to be recomputed when the pair (s_{k-m}, y_{k-m}) is replaced by (s_k, y_k). Another $4mn$ multiplications are needed to compute $H_k g_k$ if (5.8) is used. This implies that the straightforward approach is considerably less efficient compared with the two-loop recursion algorithm.

The limited memory BFGS algorithm is stated as follows.

Algorithm 5.3. (The limited memory BFGS algorithm)

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$d_1 = -g_1$$

   and $k = 1$.
2. Find α_k which satisfies the Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.

3. If $\|g_{k+1}\| = 0$ then STOP, otherwise calculate

$$d_{k+1} = -H_{k+1}g_{k+1}$$

where $H_{k+1}g_{k+1}$ is evaluated according to the two-loop recursion procedure.
4. Increase k by one and go to Step 2.

5.2 Global Convergence of the Limited Memory BFGS Algorithm

The limited memory BFGS algorithm is globally convergent if the function f is twice continuously differentiable and its Hessian is uniformly bounded. The proof of this fact is essentially the same as the proof of the global convergence of the BFGS version of a quasi-Newton method. Since the matrix H_k is the result of a finite number of the BFGS updates of a diagonal matrix the global convergence is easier to establish.

Lemma 5.1. *Assume that the level set*

$$\mathscr{M} = \{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\}$$

is convex and there exist positive constants m and M such that

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2 \tag{5.10}$$

for all $x \in \mathscr{M}$ and $z \in \mathscr{R}^n$. Let α_k be chosen according to the Wolfe conditions. Then the following inequalities hold:

$$\frac{\|y_k\|^2}{y_k^T s_k} \leqslant M \tag{5.11}$$

$$\frac{s_k^T B_k s_k}{y_k^T s_k} \leqslant \frac{\alpha_k}{1-\eta} \tag{5.12}$$

where B_k is determined according to the BFGS formula for the Hessian matrix approximation

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k} \tag{5.13}$$

with B_1 a symmetric positive definite matrix.

Proof. The proof heavily borrows from the convergence analysis given in [29] (cf. [27] and [146]).

In order to prove (5.11) define $z_k = \bar{G}_k^{\frac{1}{2}} s_k$, where $\bar{G}_k^{\frac{1}{2}} \bar{G}_k^{\frac{1}{2}} = \bar{G}_k$ and

$$\bar{G}_k = \int_0^1 \nabla^2 f(x_k + \tau \alpha_k d_k) d\tau.$$

Then, in order to derive (5.11) we notice that

$$\begin{aligned} \frac{y_k^T y_k}{s_k^T y_k} &= \frac{s_k^T \bar{G}_k^2 s_k}{s_k^T \bar{G}_k s_k} \\ &= \frac{z_k^T \bar{G}_k z_k}{\|z_k\|^2} \\ &\leqslant M. \end{aligned}$$

We have

$$\alpha_k g_k = -B_k s_k$$

thus

$$\alpha_k s_k^T g_k = -s_k^T B_k s_k. \tag{5.14}$$

From the curvature condition we also have

$$\begin{aligned} y_k^T s_k &= g_{k+1}^T s_k - g_k^T s_k \\ &\geqslant (1-\eta)\left(-g_k^T s_k\right). \end{aligned} \tag{5.15}$$

Taking into account (5.14) and (5.15) we prove (5.12):

$$\begin{aligned} \frac{s_k^T B_k s_k}{y_k^T s_k} &\leqslant \frac{s_k^T B_k s_k}{(1-\eta)(-g_k^T s_k)} \\ &= \frac{\alpha_k}{1-\eta}. \end{aligned}$$

□

The next lemma plays crucial role in the convergence analysis of the BFGS quasi-Newton method.

Lemma 5.2. *Suppose that the sequence $\{B_k\}$ is constructed by the BFGS formula with $s_k^T y_k > 0$ and B_1 a positive definite matrix. Let α_k be chosen according to the Wolfe conditions. Then*

$$\operatorname{trace}(B_{k+1}) = \operatorname{trace}(B_k) - \frac{\|B_k s_k\|^2}{s_k^T B_k s_k} + \frac{\|y_k\|^2}{s_k^T y_k} \tag{5.16}$$

$$\det(B_{k+1}) = \det(B_k) \frac{s_k^T y_k}{s_k^T B_k s_k} \tag{5.17}$$

where $\mathrm{trace}(A)$ *and* $\det(A)$ *denote the trace and the determinant of the matrix A (cf. Appendix B). Furthermore, if* B_k *is updated according to the limited memory BFGS formula with*

$$B_k^0 = \frac{y_k^T y_k}{s_k^T y_k} I = \gamma_k I$$

and the assumptions of Lemma 5.1 *are satisfied then there exist positive constants* c_1 *and* c_2 *such that*

$$\mathrm{trace}(B_k) \leqslant c_1 \tag{5.18}$$

$$\det(B_k) \geqslant c_2. \tag{5.19}$$

Proof. The trace of a square matrix is the sum of its elements on the diagonal. Therefore, we have

$$\begin{aligned}\mathrm{trace}(B_{k+1}) &= \mathrm{trace}(B_k) - \mathrm{trace}\left(\frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}\right) + \mathrm{trace}\left(\frac{y_k y_k^T}{s_k^T y_k}\right) \\ &= \mathrm{trace}(B_k) - \frac{\|B_k s_k\|^2}{s_k^T B_k s_k} + \frac{\|y_k\|^2}{s_k^T y_k}\end{aligned}$$

thus (5.16) is shown.

The proof of (5.17) is more elaborate and since it is technical in nature we do not present it here. The details can be found in [157] (see also the sketch of the proof in [146] – the proof of Theorem 8.5 and Exercise 8.9).

Since

$$B_k^0 = \frac{y_k^T y_k}{s_k^T y_k} I$$

and

$$\frac{m}{M} \leqslant \frac{y_k^T y_k}{s_k^T s_k} \cdot \frac{s_k^T s_k}{s_k^T y_k} = \frac{y_k^T y_k}{s_k^T y_k} \leqslant M$$

(note (5.10) and that $s_k^T y_k = s_k^T \bar{G}_k p_k$) we have

$$\mathrm{trace}(B_k^0) \leqslant M \tag{5.20}$$

$$\det(B_k^0) \geqslant \frac{m}{M}. \tag{5.21}$$

In order to show (5.18) we take into account (5.16) and (5.20) which give the estimate

$$\mathrm{trace}(B_k) \leqslant M + rM. \tag{5.22}$$

Here, r denotes the number of pairs (y_i, s_i) which build the matrix B_k. Thus we take $c_1 = (1+r)M$.

Due to assumption (5.10) the step-sizes α_k are uniformly bounded by $2(1-\mu)/m$ – see the proof of Theorem 7.4. Thus, (5.19) follows from (5.12), (5.17) and (5.21) because

$$\det(B_k) \geqslant \left(\frac{1-\eta}{\alpha_k}\right)^r \det(B_k^0) \geqslant \left(\frac{m(1-\eta)}{2(1-\mu)}\right)^r \frac{m}{M} = c_2.$$

□

We will need estimates (5.18)–(5.19) to prove the global convergence of the limited memory BFGS algorithm. They have been obtained under the assumption of the convexity of the function f. However, these estimate can also be derived under more general conditions.

Lemma 5.3. *Let $\{B_k\}$ be generated by the limited memory BFGS formula with $B_k^0 = \gamma_k I$ and y_k, s_k, α_k satisfying the conditions*

$$m_1 \leqslant \frac{y_k^T y_k}{s_k^T y_k} \leqslant m_2$$
$$\alpha_k \leqslant m_3$$

for all k and some constants m_1, m_2 and m_3. Then, there exist positive constants c_1 and c_2 such that (5.18) and (5.19) hold.

Proof. The proof follows the lines of the proofs of Lemma 5.1 and Lemma 5.2. □

The global convergence of the limited memory BFGS algorithm is a straightforward conclusion of Theorem 2.1 and Lemma 5.3.

Theorem 5.1. *Suppose that $\{B_k\}$ is generated by the limited memory BFGS formula with $B_k^0 = \gamma_k I$ and the assumptions of* Lemma 5.1 *are fulfilled Then*

$$\lim_{k\to\infty} \|g_k\| = 0.$$

Proof. In order to use the results of Theorem 2.1 first of all we have to show that d_k are directions of descent. B_k is a positive definite matrix and so is B_k^{-1}, thus

$$s_k^T g_k = -g_k B_k^{-1} g_k < 0.$$

From Theorem 2.1 it follows that

$$\lim_{k\to\infty} \cos^2\theta_k \|g_k\|^2 = 0, \tag{5.23}$$

where θ_k is the angle between vectors s_k and $-g_k$, i.e.

$$\cos\theta_k = \frac{-g_k^T s_k}{\|g_k\|\|s_k\|}. \tag{5.24}$$

We need to prove that there exists $c > 0$ such that $\cos\theta_k \geqslant c$. Then, from (5.23), the thesis will follow.

Since B_k is a positive definite matrix, from the SVD decomposition (cf. Appendix B), its spectral norm $\|B_k\|_2$ is given by

$$\|B_k\|_2 = \lambda_1(B_k)$$

where $\lambda_1(B_k)$ is the largest eigenvalue of B_k. Analogously, the spectral norm of B_k^{-1} is stated by

$$\|B_k^{-1}\|_2 = \frac{1}{\lambda_n(B_k)}$$

where $\lambda_n(B_k)$ is the smallest eigenvalue of B_k. Furthermore, we have

$$\lambda_n(B_k)\|x\|^2 = \frac{\|x\|^2}{\|B_k^{-1}\|_2} \leqslant x^T B_k x \leqslant \|B_k\|_2\|x\|^2 = \lambda_1(B_k)\|x\|^2.$$

Consider now the sequence $\{B_k\}$. From Lemma 5.3 there exist positive constants d_1 and d_2 such that

$$\lambda_1(B_k) \leqslant d_1 \tag{5.25}$$
$$\lambda_n(B_k) \geqslant d_2 \tag{5.26}$$

for all k. It follows from the fact that

$$\text{trace}(B_k) = \sum_{i=1}^{n} \lambda_i(B_k)$$
$$\det(B_k) = \prod_{i=1}^{n} \lambda_i(B_k)$$

where $\lambda_1(B_k), \ldots, \lambda_n(B_k)$ are eigenvalues of B_k. Equations (5.25)–(5.26) imply that

$$\|B_k\|_2\|B_k^{-1}\|_2 \leqslant c$$

for some c and all k.

On the other hand,

$$\begin{aligned}\cos\theta_k &= \frac{g_k^T B_k^{-1} g_k}{\|g_k\|\|B_k^{-1}g_k\|} \\ &\geqslant \frac{\|g_k\|^2}{\|B_k\|_2\|g_k\|\|B_k^{-1}\|_2\|g_k\|} \\ &\geqslant \frac{1}{c}\end{aligned}$$

which together with (5.23) prove the thesis. □

The limited memory BFGS algorithm has only proven linear rate of convergence. The following theorem is proved in [125].

Theorem 5.2. *Suppose that assumptions of* Lemma 5.1 *are satisfied. Then there exists* $r \in [0,1)$ *such that*

$$f(x_{k+1}) - f(\bar{x}) \leqslant r^k [f(x_1) - f(\bar{x})] \tag{5.27}$$

and $\{x_k\}$ *converges R-linearly to the solution* $\bar{x}$.

Proof. The following relation is crucial in the proof:

$$f(x_{k+1}) - f(\bar{x}) \leqslant \left[1 - c\cos^2\theta_k\right](f(x_k) - f(\bar{x})). \tag{5.28}$$

Here, c is some positive scalar and $\cos\theta_k$ is defined by (5.24). Equation (5.28) is proved, for example, in [166], [29] – we follow the proof given in [29].

As in (4.69) we can prove that

$$f(x_{k+1}) - f(x_k) \leqslant -\tilde{c}\|g_k\|^2\cos^2\theta_k \tag{5.29}$$

with $\tilde{c} = \mu(1-\eta)/M$. On the other hand, since f is convex (cf. Appendix A), we have

$$\begin{aligned} f(x_k) - f(\bar{x}) &\leqslant g_k^T(x_k - \bar{x}) \\ &\leqslant \|g_k\|\|x_k - \bar{x}\| \end{aligned} \tag{5.30}$$

and if we introduce the average Hessian value on the segment determined by x_k and $\bar{x}$:

$$\tilde{G}_k = \int_0^1 \nabla^2 f(x_k + \tau(\bar{x} - x_k))d\tau,$$

then

$$g_k = \tilde{G}_k(x_k - \bar{x})$$

and

$$m\|x_k - \bar{x}\|^2 \leqslant g_k^T(x_k - \bar{x}),$$

from which we have

$$\|x_k - \bar{x}\| \leqslant \frac{1}{m}\|g_k\|$$

which together with (5.30) imply

$$m(f(x_k) - f(\bar{x})) \leqslant \|g_k\|^2$$

which substituted into (5.29) gives (5.28) with $c = m\tilde{c}$.

If we introduce r satisfying: $0 \leqslant r \leqslant 1-c < 1$, then (5.28) leads to (5.27). Furthermore, from our assumptions, since $g(\bar{x}) = 0$,

$$\frac{1}{2}m\|x_k - \bar{x}\|^2 \leqslant f(x_k) - f(\bar{x})$$

which with (5.27) give

$$\|x_{k+1} - \bar{x}\| \leqslant r^{k/2}\left[2(f(x_1) - f(\bar{x}))/m\right]^{1/2}$$

which completes the proof. □

Theorem 5.1 establishes global convergence of the limited memory BFGS algorithm since always $H_k = B_k^{-1}$. These results can be extended to algorithms in which matrices B_k are updated according to the formula by Broyden

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k} + \phi\left(s_k^T B_k s_k\right) v_k v_k^T$$

where $\phi \in [0,1)$ and

$$v_k = \frac{y_k}{s_k^T y_k} - \frac{B_k s_k}{s_k^T B_k s_k}.$$

(We recall that $\phi = 0$ gives the BFGS formula while $\phi = 1$ leads to the DFP method.) This follows from the fact that the formula (5.17) is still valid and although (5.16) is not true (and thus (5.22) is not valid) we can show that there exists the positive c such that

$$\text{trace}(B_k) \leqslant c^k.$$

provided that (5.10) holds – the details are shown in [29]. Notice that Theorem 5.1 does not cover the case of the DFP updating formula.

5.3 Compact Representation of the BFGS Approximation of the Inverse Hessian Matrix

The two-loop recursion algorithm is an effective tool for computing the product $H_k g_k$ when H_k is built by the limited memory BFGS formula. The cost of the operation is $(4m+1)n$ multiplications when the matrix H_k^0 is diagonal. If we construct only the method for unconstrained problems then the two-loop recursion algorithm is the right choice. However, the recursive formula is less economical when we perform some calculations which are required in the constrained case. More specifically in many algorithms for bound constrained problems we need the operations $H_k e_i$ where e_i is the unit vector. It appears that the cost of this operation is comparable

to the cost of the $H_k g_k$ evaluation if the two-loop recursion formula is used. Moreover, some approaches in constrained optimization are based on the direct Hessian approximation $B_k = H_k^{-1}$. Apparently we do not have the representation of B_k in the form (5.7).

These two reasons persuaded Byrd, Nocedal and Schnabel to develop another form of H_k and B_k representation [28]. The new representation is often more convenient than (5.7).

Let us define the $n \times k$ matrices S_k and Y_k by

$$S_k = [s_1, s_2, \ldots, s_k], \quad Y_k = [y_1, y_2, \ldots, y_k].$$

As in Sect. 1 we introduce the matrix V_k defined by (5.3)–(5.4). (One can check that $V_k V_k = V_k$ thus it is a projection matrix.) The following lemma, given in [28], is needed to derive the compact representation of the matrix H_k.

Lemma 5.4. *The product of k projection matrices of the form (5.3) satisfies*

$$V_1 \cdots V_k = I - Y_k R_k^{-1} S_k \tag{5.31}$$

where R_k is the $k \times k$ matrix

$$(R_k)_{i,j} = \begin{cases} s_i^T y_j & \text{if } i \leqslant j \\ 0 & \text{otherwise} \end{cases}. \tag{5.32}$$

Proof. According to (5.32) the matrix R_k is as follows

$$R_k = \begin{bmatrix} s_1^T y_1 & s_1^T y_2 & \cdots & s_1^T y_k \\ 0 & s_2^T y_2 & \cdots & s_2^T y_k \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \vdots & s_k^T y_k \end{bmatrix}.$$

As in [28] the proof is by induction. For $k = 1$ we have

$$R_1^{-1} = \frac{1}{\rho_k},$$

thus, due to the definition of V_1,

$$V_1 = I - y_1 \frac{1}{\rho_k} s_1^T = I - y_1 R_1^{-1} s_1^T$$

and the thesis holds. Assume now that (5.31) is valid for k, we have to prove that (5.31) is also true for $k+1$.

First, consider the matrix R_{k+1}^{-1}. According to (5.32) R_{k+1} can be written as

$$R_{k+1} = \begin{bmatrix} R_k & S_k^T y_{k+1} \\ 0 & 1/\rho_{k+1} \end{bmatrix}.$$

Applying the formula for the inverse of the block quadratic matrix:

$$\begin{bmatrix} A & B \\ 0 & C \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{bmatrix} \tag{5.33}$$

to the matrix R_{k+1} we come to the relation

$$R_{k+1}^{-1} = \begin{bmatrix} R_k^{-1} & -\rho_{k+1}R_k^{-1}S_k^T y_{k+1} \\ 0 & \rho_{k+1} \end{bmatrix}. \tag{5.34}$$

Notice that (cf. (5.34))

$$\begin{aligned} Y_{k+1}R_{k+1}^{-1}S_{k+1}^T &= [Y_k, y_{k+1}] \begin{bmatrix} R_k & -\rho_{k+1}R_k^{-1}S_k^T y_{k+1} \\ 0 & \rho_{k+1} \end{bmatrix} \begin{bmatrix} S_k^T \\ s_{k+1}^T \end{bmatrix} \\ &= Y_k R_k^{-1} S_k^T + \rho_{k+1} Y_k R_k^{-1} y_{k+1} s_{k+1}^T - \rho_{k+1} y_{k+1} s_{k+1}^T \end{aligned}$$

thus we can write

$$I - Y_{k+1}R_{k+1}^{-1}S_{k+1}^T = \left(I - Y_k R_k^{-1} S_k^T\right)\left(I - \rho_{k+1} y_{k+1} s_{k+1}^T\right) \tag{5.35}$$

and since

$$V_1 V_1 \cdots V_k = \left(I - Y_k R_k^{-1} P_k^T\right)\left(I - \rho_{k+1} y_{k+1} s_{k+1}^T\right)$$

from (5.35) we have that (5.31) is true with k replaced by $k+1$. This completes the proof. □

Lemma 5.4 enables us to represent H_k in a compact form suitable for efficient $H_k g_k$ evaluation.

Theorem 5.3. *Let H_1 be symmetric and positive definite and assume that the k pairs $\{(s_i, y_i)\}_{i=1}^k$ satisfy $s_i^T y_i > 0$, $i = 1, \ldots, k$. Assume that H_k is obtained by updating H_1 k times using the pairs $\{(s_i, y_i)\}_{i=1}^k$ and the inverse BFGS formula. Then*

$$\begin{aligned} H_{k+1} = H_1 + [S_k \; H_1 Y_k] &\begin{bmatrix} R_k^{-1}\left(D_k + Y_k^T H_1 Y_k\right) R_k^{-1} & -R_k^{-T} \\ -R_k^{-1} & 0 \end{bmatrix} \times \\ &\times \begin{bmatrix} S_k^T \\ Y_k^T H_1 \end{bmatrix} \end{aligned} \tag{5.36}$$

where R_k is given by (5.32) and $D_k \in \mathscr{R}^{n\times n}$ is the diagonal matrix given by

$$D_k = \operatorname{diag}\left[s_1^T y_1, \ldots, s_k^T y_k\right].$$

Proof. We follow the proof given in [28]. H_{k+1} is expressed as the sum of two matrices M_{k+1} and N_{k+1}

$$H_{k+1} = M_{k+1} + N_{k+1}, \;\; k \geqslant 1$$

with

$$\begin{aligned} M_1 &= H_1 \\ M_{k+1} &= V_k^T M_k V_k, \;\; k > 1 \end{aligned}$$

and

$$N_1 = \rho_1 s_1 s_1^T \tag{5.37}$$

$$N_{k+1} = V_k^T N_k V_k + \rho_k s_k s_k^T, \;\; k > 1. \tag{5.38}$$

From the definition of M_k and Lemma 5.4 we also have

$$\begin{aligned} M_{k+1} &= \left(V_k^T \ldots V_1^T\right) H_1 \left(V_1 \ldots V_k\right) \\ &= \left(I - S_k R_k^{-T} Y_k^T\right) H_1 \left(I - Y_k R_k^{-1} S_k^T\right). \end{aligned}$$

Notice that if

$$N_{k+1} = S_k R_k^T D_k R_k^{-T} S_k^T \tag{5.39}$$

then

$$\begin{aligned} H_{k+1} &= M_{k+1} + N_{k+1} \\ &= H_1 + \begin{bmatrix} S_k & H_1 Y_k \end{bmatrix} \begin{bmatrix} R_k^{-T}\left(D_k + Y_k^T H_1 Y_k\right) R_k^{-1} & -R_k^{-T} \\ -R_k^{-T} & 0 \end{bmatrix} \times \\ &\quad \times \begin{bmatrix} S_k^T \\ Y_k^T H_1 \end{bmatrix}. \end{aligned}$$

It remains to show that (5.39) holds. The proof is by induction. For $k = 1$ we have

$$N_1 = \rho_1 s_1 s_1^T = \rho_1 \left(s_1^T y_1\right) \rho_1 s_1 s_1^T = S_1 R_1^{-T} D_1 R_1^{-1} S_1^T$$

thus (5.39) is true in this case. Assume that (5.39) is valid for k we have to show that it is also the case for $k+1$.

Due to the definition of N and our assumption (5.39) (with (5.38)) we have

$$N_{k+2} = V_{k+1}^T S_k R_k^{-T} D_k R_k^{-1} S_k^T V_{k+1} + \rho_{k+1} s_{k+1} s_{k+1}^T. \tag{5.40}$$

Furthermore, due to the definition of V_k, we can write

$$\begin{aligned} R_k^{-1} S_k^T V_{k+1} &= R_k^{-1} S_k^T \left(I - \rho_{k+1} y_{k+1} s_{k+1}^T\right) \\ &= \begin{bmatrix} R_k^{-1} & -\rho_{k+1} R_k^{-1} S_k^T y_{k+1} \end{bmatrix} \begin{bmatrix} S_k^T \\ s_{k+1}^T \end{bmatrix} \\ &= \begin{bmatrix} R_k^{-1} & -\rho_{k+1} R_k^{-1} S_k^T y_{k+1} \end{bmatrix} S_{k+1}^T. \end{aligned}$$

Observe, from (5.34), that

$$[I \;\; 0]R_{k+1}^{-1} = \left[R_k^{-1} \;\; -\rho_{k+1}R_k^{-1}S_k^T y_{k+1}\right] \tag{5.41}$$

thus

$$R_k^{-1}S_k^T V_{k+1} = [I \;\; 0]R_{k+1}^{-1}S_{k+1}^T.$$

Formula (5.34) implies that

$$s_{k+1} = S_{k+1}R_{k+1}^{-T}e_{k+1}\frac{1}{\rho_{k+1}} \tag{5.42}$$

where e_{k+1} is the unit vector. Equation (5.42) together with (5.40) and (5.41) imply that

$$\begin{aligned} N_{k+2} &= S_{k+1}R_{k+1}^{-T}\begin{bmatrix} I \\ 0 \end{bmatrix} D_k [I \;\; 0]R_{k+1}^{-1}S_{k+1}^T + \\ &\quad S_{k+1}R_{k+1}^{-T}\begin{bmatrix} 0 & & \\ & \ddots & \\ & & 0 \\ & & & 1/\rho_{k+1} \end{bmatrix} R_{k+1}^{-1}S_{k+1}^T \\ &= S_{k+1}R_{k+1}^{-T}\begin{bmatrix} D_k & 0 \\ 0 & 1/\rho_{k+1} \end{bmatrix} R_{k+1}^{-1}S_{k+1}^T \\ &= S_{k+1}R_{k+1}^{-T}D_{k+1}R_{k+1}^{-1}S_{k+1}^T \end{aligned}$$

which completes the induction arguments and the proof of the theorem. □

The compact representation is useful in unconstrained optimization if the cost of the evaluation of $H_k g_k$ is comparable to that of the two-loop recursion algorithm. Suppose that at the kth iteration of the limited memory BFGS algorithm we express H_k according to Theorem 5.3, i.e.

$$\begin{aligned} H_k = \sigma_k I + [S_k \;\; \sigma_k Y_k] &\begin{bmatrix} R_k^{-1}\left(D_k + \sigma_k Y_k^T Y_k\right)R_k^{-1} & -R_k^{-T} \\ -R_k^{-1} & 0 \end{bmatrix} \times \\ &\times \begin{bmatrix} S_k^T \\ \sigma_k Y_k^T \end{bmatrix} \end{aligned} \tag{5.43}$$

where we have assumed that $H_k^0 (= H_1) = \sigma_k I$,

$$S_k = [s_{k-m}, \ldots, s_{k-1}], \quad Y_k = [y_{k-m}, \ldots, y_{k-1}]$$

and

$$(R_k)_{i,j} = \begin{cases} \left(s_{k-m-1+i}\right)^T \left(y_{k-m-1+j}\right) & \text{if } i \leqslant j \\ 0 & \text{otherwise} \end{cases}.$$

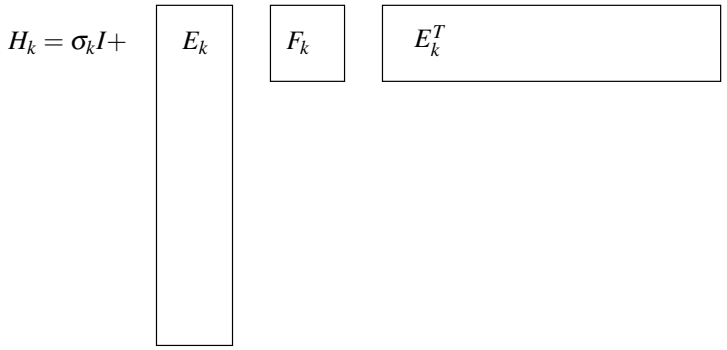

Fig. 5.1 The compact representation of the matrix H_k

$$D_k = \text{diag}\left[s_{k-m}^T y_{k-m}, \ldots, s_{k-1}^T y_{k-1}\right].$$

The block structure of the compact representation of the matrix H_k is given in Fig. 5.1.

Having (5.43) we can estimate the cost of $H_k g_k$ evaluation. First of all, we have to compute $S_k^T g_k$ and $Y_k^T g_k$. This requires $2mn$ multiplications. Secondly, we need to update the matrices $Y_k^T Y_k$, R_k and D_k. We can do that by using $Y_{k-1}^T Y_{k-1}$, R_{k-1} and D_{k-1} from the previous iteration. Then, in order to evaluate R_k from R_{k-1}, we have to delete the first row and column from R_{k-1} and then add a new column on the right given by

$$S_k^T y_{k-1} = S_k^T (g_k - g_{k-1})$$

and a new row at the bottom, which is zero in its first $m-1$ components. Since $S_k^T g_k$ has to be calculated anyway and $S_k^T g_{k-1}$ has its first $m-1$ components equal to the last $m-1$ components of $S_{k-1}^T g_{k-1}$, which we can store from the previous iteration, the cost of evaluating $S_k^T y_{k-1}$ is determined by the cost of calculating $s_{k-1}^T g_{k-1}$. However, this can be accomplished in $O(m^2)$ operations – see [28] for details.

The new components of R_k, $Y_k^T Y_k$ and D_k can be computed in $O(m)$ operations using the formulae

$$s_i^T y_{k-1} = s_i^T g_k - s_i^T g_{k-1}, \quad i = k-m, \ldots, k-1 \tag{5.44}$$

$$y_i^T y_{k-1} = y_i^T g_k - y_i^T g_{k-1}, \quad i = k-m, \ldots, k-2 \tag{5.45}$$

$$y_{k-1}^T y_{k-1} = -g_k^T g_k + 2(g_k - g_{k-1})^T g_k + g_{k-1}^T g_{k-1} \tag{5.46}$$

since the quantities on the right are already available once $g_k^T g_k$, $S_k^T g_k$ and $Y_k^T g_k$ have been calculated.

Finally, $H_k g_k$ is determined by

$$H_k g_k = \sigma_k g_k + [S_k \;\; \sigma_k Y_k]\, w \tag{5.47}$$

where $w \in \mathscr{R}^{2m}$ is given by

$$w = \begin{bmatrix} R_k^{-T}\left(D_k + \sigma_k Y_k^T Y_k\right) R_k^{-1}\left(S_k^T g_k\right) - \sigma_k R_k^{-T}\left(Y_k^T g_k\right) \\ -R_k^{-1}\left(S_k^T g_k\right) \end{bmatrix}.$$

We can summarize our analysis in the procedure which lists the data carried out from the previous iterations and puts the calculations in the proper order.

Algorithm 5.4. (The compact update of the matrix H_k)

Data: matrices S_{k-1}, Y_{k-1}, R_{k-1}, $Y_{k-1}^T Y_{k-1}$, D_{k-1}; vectors $S_{k-1}^T g_{k-1}$, $Y_{k-1}^T g_{k-1}$ and scalar $g_{k-1}^T g_{k-1}$ (from the previous iteration). Vectors s_{k-1}, y_{k-1} and g_k evaluated at the current iteration.

1. Remove the first vectors from S_{k-1} and Y_{k-1}. Add vectors s_{k-1} and y_{k-1} at the last positions of the matrices S_{k-1} and Y_{k-1}, respectively, to create the matrices S_k and Y_k.
2. Compute $g_k^T g_k$, $S_k^T g_k$, $Y_k^T g_k$ and $s_{k-1}^T g_{k-1}$.
3. Update R_k, $Y_k^T Y_k$ and D_k with the help of (5.44)–(5.46).
4. Assume σ_k and evaluate $H_k g_k$ according to (5.47)–(5.48).

The Steps 2 and 4 require $(2m+1)n$ floating point operations and the rest operations amount to $O(m^2)$. Thus, in total the compact representation of the limited memory BFGS enables to determine $H_k g_k$ in $(4m+2)n + O(m^2)$ operations which is comparable to the cost of the two-loop recursion algorithm.

5.4 The Compact Representation of the BFGS Approximation of the Hessian Matrix

The limited memory BFGS method applied to bound constrained problems requires the compact representation of the matrix B_k [26]. Since $B_k = H_k^{-1}$ the representation of the matrix B_k can be derived from the formula for H_k using the Sherman–Morrison–Woodbury rule. According to (5.36) H_{k+1} can be written as

$$H_{k+1} = H_1 + E_k F_k E_k^T$$

with

$$E_k = [S_k \;\; H_1 Y_k] \tag{5.48}$$

$$F_k = \begin{bmatrix} R_k^{-T}\left(D_k + Y_k^T H_1 Y_k\right) R_k^{-1} & -R_k^{-T} \\ -R_k^{-1} & 0 \end{bmatrix}, \tag{5.49}$$

thus, due to Sherman–Morrison–Woodbury formula (cf. (B.17) with S replaced by E_k and T by $F_k E_k^T$) we have

$$\begin{aligned} B_{k+1} &= B_1 - B_1 E_k \left(I + F_k E_k^T B_1 F_k\right)^{-1} F_k E_k^T B_1 \\ &= B_1 - B_1 E_k \left(F_k^{-1} + E_k^T B_1 E_k\right)^{-1} E_k^T B_1 \end{aligned} \tag{5.50}$$

where $B_1 = H_1^{-1}$. Applying (5.33) to F_k^{-1} results in

$$F_k^{-1} = \begin{bmatrix} 0 & -R_k \\ -R^{-T} & -\left(D_k + Y_k^T H_1 Y_k\right) \end{bmatrix}$$

and since

$$\begin{aligned} E_k^T B_1 E_k &= \begin{bmatrix} S_k^T \\ Y_k^T H_1 \end{bmatrix} B_1 \begin{bmatrix} S_k & H_1 Y_k \end{bmatrix} \\ &= \begin{bmatrix} S_k^T B_1 S_k & S_k^T Y_k \\ Y_k^T S_k & Y_k^T H_1 Y_k \end{bmatrix} \end{aligned} \tag{5.51}$$

we have

$$F_k^{-1} + E_k^T B_1 E_k = \begin{bmatrix} S_k^T B_1 S_k & S_k^T Y_k - R_k \\ Y_k^T S_k - R_k^T & -D_k \end{bmatrix}. \tag{5.52}$$

Combining (5.50) and (5.52) leads to the theorem which is the counterpart of Theorem 5.3.

Theorem 5.4. *Suppose that B_1 is symmetric and positive definite and that the pairs $\{(s_i, y_i)\}_{i=1}^k$ satisfy $s_i^T y_i > 0$. If B_k is obtained by applying the BFGS formula to the matrix B_1 k times using (5.13) and the pairs $\{(s_i, y_i)\}_{i=1}^k$. Then*

$$B_{k+1} = B_1 - \begin{bmatrix} B_1 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_1 S_k & L_k \\ L_k & -D_k \end{bmatrix} \begin{bmatrix} S_k^T B_1 \\ Y_k \end{bmatrix}, \tag{5.53}$$

with

$$L_k = S_k^T Y_k - R_k. \tag{5.54}$$

Furthermore, the matrix (5.52) can be decomposed in the following way

$$\begin{bmatrix} S_k^T B_1 S_k & L_k \\ L_k^T & -D_k \end{bmatrix} = \begin{bmatrix} J_k & -L_k D_k^{-\frac{1}{2}} \\ 0 & D_k^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} J_k^T & 0 \\ D_k^{-\frac{1}{2}} L_k^T & -D_k^{\frac{1}{2}} \end{bmatrix}, \tag{5.55}$$

where J_k is the lower triangular matrix satisfying

$$J_k J_k^T = S_k^T B_1 S_k + L_k D_k^{-1} L_k^T. \tag{5.56}$$

Proof. Equations (5.53)–(5.54) have already been proved. In order to justify (5.55) it suffices to show that the matrix on the right-hand side of (5.56) is positive definite. The proof is straightforward. First, notice that since B_1 and D_k are positive definite, the matrices $S_k^T B_1 S_k$ and $L_k D_k^{-1} L_k^T$ are positive semidefinite. The relation

$$z^T \left(S_k^T B_1 S_k + L_k D_k^{-1} L_k^T \right) z = 0.$$

happens if $L_k^T z = 0$ and $S_k z = 0$ which implies that $Y_k^T S_k z = 0$. Taking into account (5.54) we also have that $R_k^T z = 0$ thus $z = 0$ since R_k is tridiagonal with positive elements on its diagonal. □

The compact representation of B_k is used in algorithms for constrained optimization such as the limited memory BFGS method for problems with bounds [26]. Suppose that at the kth iteration of the limited memory BFGS algorithm we express B_k according to Theorem 5.4, i.e.

$$B_k = \gamma_k I - [\gamma_k S_k \;\; Y_k] \begin{bmatrix} \gamma_k S_k^T S_k & L_k \\ L_k & = -D_k \end{bmatrix} \begin{bmatrix} \gamma_k S_k^T \\ Y_k^T \end{bmatrix}$$

where $\gamma_k = 1/\sigma_k$ and

$$S_k = [s_{k-m}, \ldots, s_{k-1}], \quad Y_k = [y_{k-m}, \ldots, y_{k-1}].$$

The implementation of these algorithms requires the efficient BFGS update of B_k and the efficient computation of $B_k v$ and $Z^T B_k Z$ where $v \in \mathscr{R}^n$ and Z is an $n \times t$ matrix whose columns are unit vectors.

The following algorithm updates B_k and computes the product $B_k v$.

Algorithm 5.5. (The compact update of the matrix B_k)

Data: matrices S_{k-1}, Y_{k-1}, L_{k-1}, $S_{k-1}^T S_{k-1}$, D_{k-1}; vectors s_{k-1}, y_{k-1} evaluated at the current iteration and a given v.

1. Remove the first vectors from S_{k-1} and Y_{k-1}. Add vectors s_{k-1} and y_{k-1} at the last positions of the matrices S_{k-1} and Y_{k-1}, respectively, to create the matrices S_k and Y_k.
2. Compute L_k, $S_k^T S_k$, D_k and γ_k, e.g. by

$$\gamma_k = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}.$$

3. Compute the Cholesky factorization of the matrix $\gamma_k S_k^T S_k + L_k D_k^{-1} L_k^T$ to obtain $J_k J_k^T$.
4. Solve the system of linear equations for p

$$\begin{bmatrix} J_k & -L_k D_k^{-\frac{1}{2}} \\ 0 & D_k^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} J_k^T & 0 \\ D_k^{-\frac{1}{2}} L_k^T & -D_k^{\frac{1}{2}} \end{bmatrix} p = w,$$

where

$$w = \begin{bmatrix} Y_k^T v \\ \gamma_k S_k^T v \end{bmatrix}.$$

5. Evaluate

$$B_k v = \gamma_k v - [Y_k \;\; \gamma_k S_k]\, p.$$

In the above algorithm we can use the representation of L_k

$$(L_k)_{i,j} = \begin{cases} s_{k-m-1+i}^T y_{k-m-1+j} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases} \tag{5.57}$$

which follows directly from the definition of L_k.

The Step 2 costs $(2m+1)n$ floating point operations, as the Step 5. The Cholesky factorization can be accomplished in $O(m^3)$ operations the same as the solution to the equations in Step 4. Adding $2mn$ operations in Step 4 needed to calculate the vector w we come up with the total count of operations equal to $(6m+1)n+O(m^2)$.

The evaluation of $Z^T B_k Z$, with $Z \in \mathscr{R}^{n\times t}$, can be efficiently done using the representation

$$Z^T B_k Z = \gamma_k I_t - \begin{bmatrix} \hat{Y}_k & \gamma_k \hat{S}_k \end{bmatrix} \begin{bmatrix} J_k^T & 0 \\ D_k^{-\frac{1}{2}} L_k^T & -D_k^{\frac{1}{2}} \end{bmatrix}^{-1} \begin{bmatrix} J_k & -L_k D_k^{-\frac{1}{2}} \\ 0 & D_k^{\frac{1}{2}} \end{bmatrix} \times$$
$$\times \begin{bmatrix} \hat{Y}_k^T \\ \gamma_k \hat{S}_k^T \end{bmatrix},$$

where I_t is the identity matrix of dimension t and $\hat{Y}_k = ZY_k$, $\hat{S}_k = ZS_k$ are $t \times m$ matrices.

5.5 Numerical Experiments

In Tables 5.1–5.2 we show how the number m influences the performance of the L-BFGS-B code which is the implementation of the algorithm presented in [26] intended for bound constrained optimization. The stopping criterion assumed in numerical experiments was

$$\|g_k\| \leqslant \varepsilon \max[1, \|x\|] \leqslant 10^{-5}. \tag{5.58}$$

L-BFGS-B code was tested on problems from the CUTE collection characterized in Table 7.4. These are the same as problems described in Table 3.1 with the exception that dimensions of these problems were set to the their maximal values (for few

Table 5.1 Numerical results: performance of L-BFGS-B code with different values of m on problems given in Table 7.4

Problem	L-BFGS-B (m = 3)			L-BFGS-B (m = 5)			L-BFGS-B (m = 17)		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	2	18	0.28	2	18	0.28	2	18	0.27
BROYDN7D	14385	15474	308.9	14097	14878	335.44	14067	14730	563.09
BRYBND	27	39	0.38	26	38	0.43	33	43	0.91
CHAINWOO	32807	35318	368.81	31701	34027	437.78	31984	34397	966.98
DIXON3DQ	3841	403	24.72	7291	7546	64.16	12378	12665	290.16
DQDRTIC	14	23	0.08	10	18	0.07	8	15	0.06
DQRTIC	43	56	0.17	43	56	0.22	63	77	0.76
EIGENALS	40386	42702	794.17	14552	15057	288.47	15711	16161	362.17
EXTROSNB	809	1024	0.02	733	974	0.03	678	897	0.12
FLETCHBV	FAIL	FAIL	573.79	FAIL	FAIL	698.9	FAIL	FAIL	925.99
FLETCHCR	5114	5519	4.72	5002	5684	5.88	4931	5778	13.8
FMINSURF	870	908	16.71	660	674	15.37	495	503	24.73
GENHUMPS	7685	9904	55.4	6217	8602	55.36	6287	8479	107.53
GENROSE	1179	1304	0.56	1093	1241	0.66	1079	1319	1.57
HILBERTA	26	35	0.01	13	17	0.01	12	15	0.01
LIARWD	26	40	0.3	24	35	0.32	25	35	0.58
MANCINO	9	13	0.16	8	12	0.15	7	11	0.14
MOREBV	30	35	0.14	26	30	0.15	25	29	0.27
NONCVXU2	14919	15690	169.02	12228	12585	169.71	8132	8332	246.47
NONCVXUN	FAIL	FAIL	540.46	FAIL	FAIL	668.92	FAIL	FAIL	1476.89
NONDIA	16	31	0.24	17	32	0.28	20	34	0.48
NONDQUAR	1242	1387	9.91	938	1032	9.9	997	1120	26.69
POWELLSG	71	95	0.61	45	55	0.49	33	40	0.73
POWER	470	504	3.57	402	433	4.08	370	394	9.62
QUARTC	45	59	0,38	45	59	0.48	45	59	1.05
SCHMVETT	41	48	0.7	39	45	0.75	35	44	1.11
SENSORS	23	32	0.26	25	34	0.28	26	35	0.29
SPARSINE	15201	16109	18.07	11430	11988	16.36	6533	6807	19.8
SPMSRTLS	208	222	2.46	190	204	2.73	183	193	5.5
SROSENBR	14	23	0.13	15	23	0.17	15	23	0.24
TOINTGSS	2	7	0.04	2	7	0.04	2	7	0.04
TQUARTIC	18	30	0.18	16	28	0.2	19	30	0.37
TRIDIA	4281	4502	33.61	2607	2705	27.13	1615	1665	43.05
VAREIGVL	87	98	0.7	74	82	0.68	67	76	1.08
WOODS	68	95	0.68	48	74	0.6	45	67	1.13
Total	193547	251747	2930.34	160104	218293	2806.48	156864	664102	5093.68

problems we applied smaller dimensions since their maximal values led to excessive computing times).

The results of numerical comparisons are presented in Figs. 5.2–5.7. These results show that the optimal choice of m does not necessarily mean the large value of m, also with respect to the criterion of the number of function evaluations. 'FAIL' in the tables refers to the situations when the prescribed maximum

Table 5.2 Numerical results: performance of L-BFGS-B code with different values of m on problems given in Table 7.4

Problem	L-BFGS-B (m = 5)			L-BFGS-B (m = 25)			L-BFGS-B (m = 35)		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	2	18	028	2	18	0.27	2	18	0.28
BROYDN7D	14097	14878	335.44	14063	14732	714.14	14047	14719	903.22
BRYBND	26	38	0.43	33	45	1.04	33	45	1.08
CHAINWOO	31701	34027	437.78	31957	34243	1300.73	32790	35007	1790.75
DIXON3DQ	7291	7546	64.16	11991	12258	390.81	11945	12225	528.21
DQDRTIC	10	18	0.07	8	15	0.06	8	15	0.06
DQRTIC	43	56	0.22	71	85	1.17	81	94	1.73
EIGENALS	14552	15057	288.47	15268	15655	379.9	12707	13057	352.83
EXTROSNB	733	974	0.03	670	874	0.24	662	894	0.49
FLETCHBV	FAIL	FAIL	698.9	FAIL	FAIL	1129.95	FAIL	FAIL	1276.74
FLETCHCR	5002	5684	5.88	4946	5791	19.51	4924	5787	27.61
FMINSURF	660	674	15.37	467	479	31.14	449	461	39.46
GENHUMPS	6217	8602	55.36	6503	8862	147.3	6537	8746	193.52
GENROSE	1093	1241	0.66	1091	1331	2.3	1097	1331	3.38
HILBERTA	13	17	0.01	12	15	0.01	12	15	0.01
LIARWD	24	35	0.32	25	35	0.62	25	35	0.62
MANCINO	8	12	0.15	7	11	0.14	7	11	0.14
MOREBV	26	30	0.15	25	29	0.3	25	29	0.29
NONCVXU2	12228	12585	169.71	8981	9197	366.42	8616	8819	466.64
NONCVXUN	FAIL	FAIL	668.92	FAIL	FAIL	2003.27	FAIL	FAIL	2651.44
NONDIA	17	32	0.28	20	34	0.49	20	34	0.49
NONDQUAR	938	1032	9.9	863	963	31.91	725	818	36.45
POWELLSG	45	55	0.49	33	40	0.86	31	38	0.81
POWER	402	433	4.08	371	393	13.35	368	389	17.79
QUARTC	45	59	0.48	45	59	1.31	47	61	1.62
SCHMVETT	39	45	0.75	35	44	1.34	35	44	1.34
SENSORS	25	34	0.28	29	98	0.81	35	188	1.55
SPARSINE	11430	11988	16.36	5653	5863	23.82	5353	5530	31.45
SPMSRTLS	190	204	2.73	186	197	7.25	184	198	9.4
SROSENBR	15	23	0.17	15	23	0.24	15	23	0.24
TOINTGSS	2	7	0.04	2	7	0.04	2	7	0.04
TQUARTIC	16	28	0.2	19	30	0.38	19	30	0.38
TRIDIA	2607	2705	27.13	1780	1832	65.93	1853	1897	93.97
VAREIGVL	74	82	0.68	71	81	1.44	67	77	1.61
WOODS	48	74	0.6	46	71	1.43	48	70	1.76
Total	160104	218293	2806.48	182210	213417	6639.92	153743	210716	8437.40

number of function evaluations (50000) has been exceeded – recorded performance parameters, before abnormal exits from L-BFGS-B code, are taken into account in the rows stating the total CPU time and the total number of iterations and functions evaluations.

In Tables 5.3–5.4 we also show the performance of L-BFGS code on problems from the CUTE collection specified in Table 7.4. The efficiency profiles are

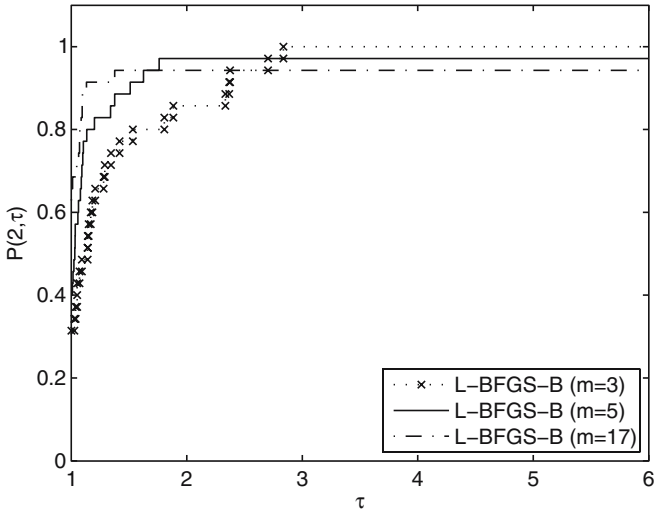

Fig. 5.2 Cumulative distribution ratio of the number of function evaluations: performance of the L-BFSG-B code for different values of m (cf. Table 5.1)

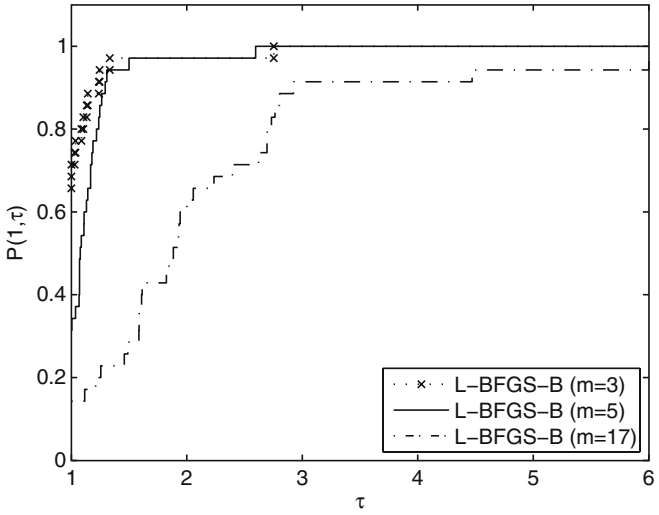

Fig. 5.3 Cumulative distribution ratio of CPU time: performance of the L-BFSG-B code for different values of m (cf. Table 5.1)

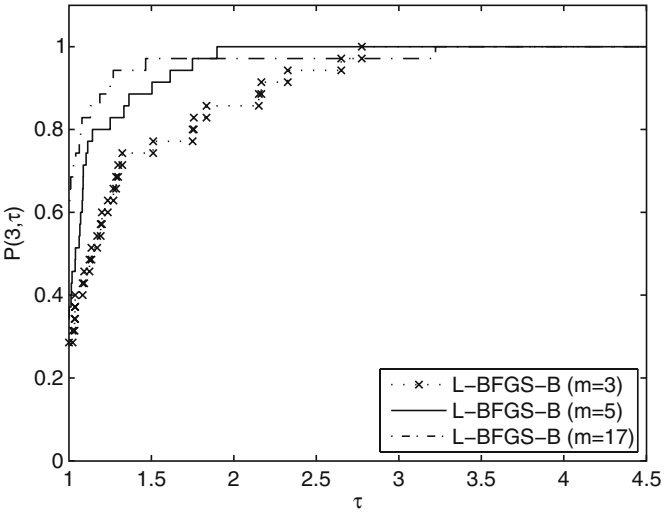

Fig. 5.4 Cumulative distribution ratio of the number of iterations: performance of the L-BFSG-B code for different values of m (cf. Table 5.1)

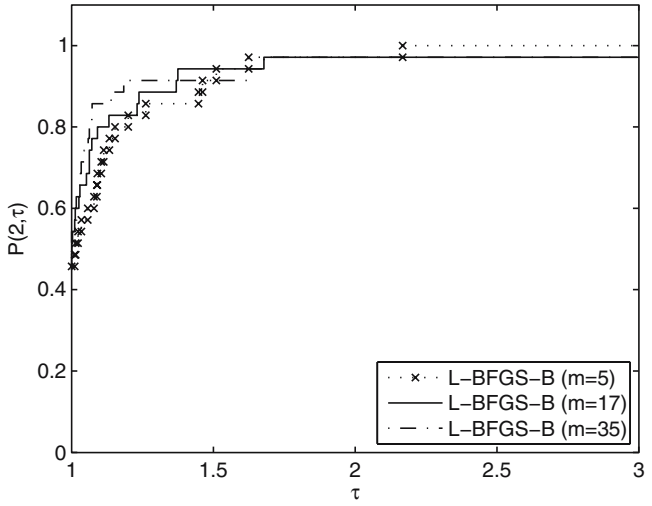

Fig. 5.5 Cumulative distribution ratio of the number of function evaluations: performance of the L-BFSG-B code for different values of m (cf. Table 5.2)

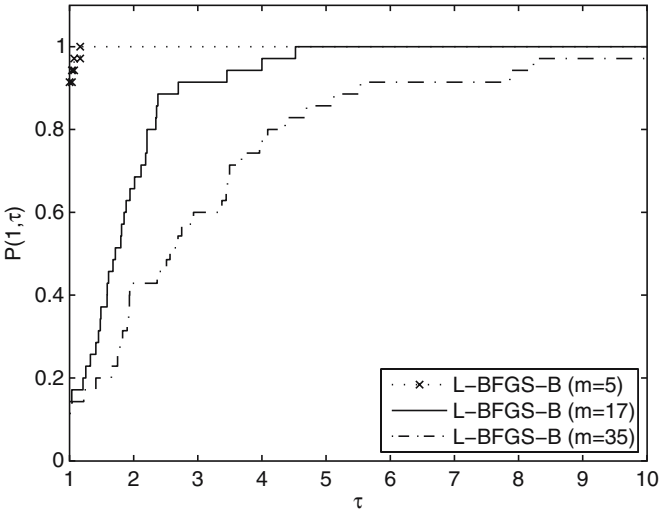

Fig. 5.6 Cumulative distribution ratio of CPU time: performance of the L-BFSG-B code for different values of m (cf. Table 5.2)

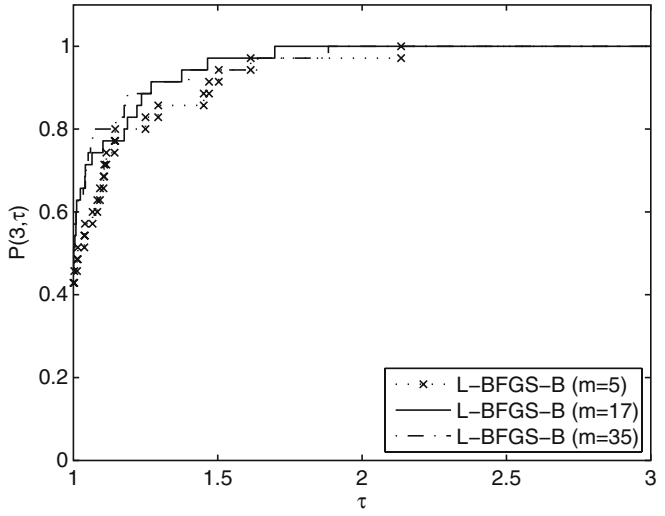

Fig. 5.7 Cumulative distribution ratio of the number of iterations: performance of the L-BFSG-B code for different values of m (cf. Table 5.2)

Table 5.3 Numerical results: performance of L-BFGS code with different values of m on problems given in Table 7.4

Problem	L-BFGS (m = 3)			L-BFGS (m = 5)			L-BFGS (m = 17)		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	3	5	0.1	3	5	0.09	3	5	0.09
BROYDN7D	3198	3242	49.3	3124	3152	48.88	3220	3300	56.08
BRYBND	82	96	0.64	41	50	0.34	45	51	0.4
CHAINWOO	31435	33847	187.28	323	357	2.07	304	336	2.4
DIXON3DQ	820	857	2.2	1961	2022	5.77	2558	2607	11.47
DQDRTIC	13	23	0.04	13	20	0.04	12	19	0.04
DQRTIC	36	44	0.05	36	44	0.06	36	44	0.08
EIGENALS	9490	10019	177.84	6163	6378	113.7	8033	8247	149.98
EXTROSNB	883	1118	0.01	812	1070	0.01	752	998	0.01
FLETCHBV	1883	1922	11.78	982	1022	6.55	900	935	7.34
FLETCHCR	5049	5442	2.12	5007	5682	2.24	4924	5777	2.75
FMINSURF	1061	1107	11.91	985	1009	11.31	722	744	10.1
GENHUMPS	8715	10660	38.38	8489	10999	40.15	8570	10768	47.34
GENROSE	1168	1307	0.25	1095	1229	0.24	1080	1318	0.3
HILBERTA	78	91	0.01	26	29	0.01	21	23	0.01
LIARWD	26	33	0.15	24	28	0.14	23	26	0.14
MANCINO	10	14	0.18	10	14	0.18	10	14	0.18
MOREBV	10	12	0.02	9	11	0.02	9	11	0.02
NONCVXU2	1818	1889	10.76	1800	1849	11.03	1966	2013	15.03
NONCVXUN	2476	2559	14.41	1884	1938	11.44	2545	2596	19.43
NONDIA	17	22	0.12	18	23	0.12	20	25	0.16
NONDQUAR	182	201	0.54	192	220	0.63	106	121	0.5
POWELLSG	156	181	0.49	60	69	0.21	39	44	0.18
POWER	556	584	1.32	407	426	1.09	404	410	1.69
QUARTC	38	46	0.12	38	46	0.13	38	46	0.16
SCHMVETT	35	38	0.4	34	41	0.43	30	34	0.39
SENSORS	20	22	0.18	20	23	0.19	21	23	0.19
SPARSINE	8715	9223	5.72	5918	6227	3.98	3638	3768	2.78
SPMSRTLS	154	164	0.99	150	162	1	147	160	1.21
SROSENBR	17	20	0.06	17	20	0.06	18	20	0.08
TOINTGSS	20	25	0.14	16	23	0.13	16	23	0.14
TQUARTIC	22	29	0.1	22	27	0.1	21	27	0.12
TRIDIA	3999	4234	10.75	2650	2739	7.85	1883	1926	8.46
VAREIGVL	16	20	0.09	16	20	0.09	16	20	0.1
WOODS	99	132	0.44	85	119	0.42	84	109	0.51
Total	82300	89228	528.89	42430	47093	270.70	42214	46588	339.86

presented in Figs. 5.8–5.13. These results support our claim that the best performance does not have to be achieved for large values of m – it seems that the choice $m = 5$ is reasonable if we take into account both the CPU time and the number of functions evaluations. Another conclusion which we can draw from Tables 5.1–5.4 is that L-BFGS code performs better than L-BFGS-B as far as the unconstrained problems are concerned. The code L-BFGS-B uses the compact representation of

Table 5.4 Numerical results: performance of L-BFGS code with different values of m on problems given in Table 7.4

Problem	L-BFGS (m = 5)			L-BFGS (m = 17)			L-BFGS (m = 35)		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	3	5	0.09	3	5	0.09	3	5	0.09
BROYDN7D	3124	3152	48.88	3220	3300	56.08	3205	3264	62.78
BRYBND	41	50	0.34	45	51	0.4	49	54	0.48
CHAINWOO	323	357	2.07	304	336	2.4	466	503	4.65
DIXON3DQ	1961	2022	5.77	2558	2607	11.47	2883	2937	19.37
DQDRTIC	13	20	0.04	12	19	0.04	12	19	0.04
DQRTIC	36	44	0.06	36	44	0.08	36	44	0.09
EIGENALS	6163	6378	113.7	8033	8247	149.98	5727	5888	110.53
EXTROSNB	812	1070	0.01	752	998	0.01	736	973	0.01
FLETCHBV	982	1022	6.55	900	935	7.34	952	985	9.88
FLETCHCR	5007	5682	2.24	4924	5777	2.75	4920	5729	3.6
FMINSURF	985	1009	11.31	722	744	10.1	711	729	12.41
GENHUMPS	8489	10999	40.15	8570	10768	47.34	8968	10786	58.66
GENROSE	1095	1229	0.24	1080	1318	0.3	1087	1328	0.37
HILBERTA	26	29	0.01	21	23	0.01	21	23	0.01
LIARWD	24	28	0.14	23	26	0.14	23	26	0.14
MANCINO	10	14	0.18	10	14	0.18	10	14	0.18
MOREBV	9	11	0.02	9	11	0.02	9	11	0.02
NONCVXU2	1800	1849	11.03	1966	2013	15.03	2005	2045	19.76
NONCVXUN	1884	1938	11.44	2545	2596	19.43	2686	2743	26.39
NONDIA	18	23	0.12	20	25	0.16	20	25	0.16
NONDQUAR	192	220	0.63	106	121	0.5	95	105	0.6
POWELLSG	60	69	0.21	39	44	0.18	40	44	0.21
POWER	407	426	1.09	404	410	1.69	397	404	2.49
QUARTC	38	46	0.13	38	46	0.16	38	46	0.2
SCHMVETT	34	41	0.43	30	34	0.39	30	34	0.4
SENSORS	20	23	0.19	21	23	0.19	23	25	0.2
SPARSINE	5918	6227	3.98	3638	3768	2.78	3654	3752	3.5
SPMSRTLS	150	162	1	147	160	1.21	145	154	1.44
SROSENBR	17	20	0.06	18	20	0.08	18	20	0.08
TOINTGSS	16	23	0.13	16	23	0.14	16	23	0.13
TQUARTIC	22	27	0.1	21	27	0.12	21	27	0.11
TRIDIA	2650	2739	7.85	1883	1926	8.46	1898	1933	12.71
VAREIGVL	16	20	0.09	16	20	0.1	16	20	0.1
WOODS	85	119	0.42	84	109	0.51	84	106	0.64
Total	42430	47093	270.70	42214	46588	339.86	41004	44824	352.43

the BFGS matrices while the code L-BFGS is based on the two-loop recursion algorithm so as far as the cost of linear algebra involved these two codes are comparable. However, these codes use different step-size selection procedures and that probably explains such a big difference in their performance (Figs. 5.14, 5.15.)

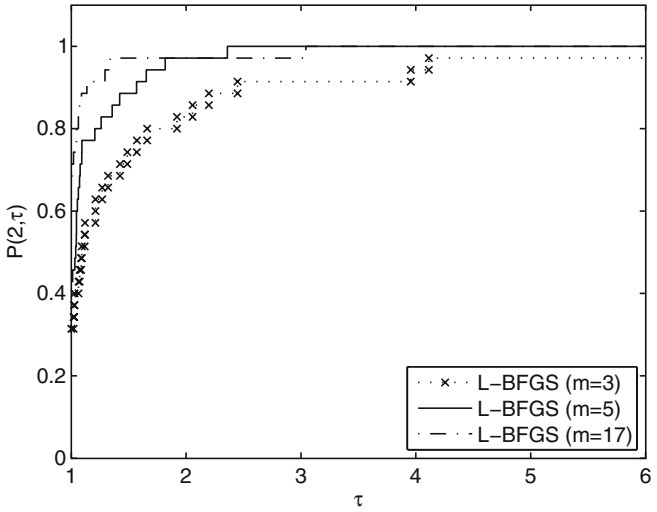

Fig. 5.8 Cumulative distribution ratio of the number of function evaluations: performance of the L-BFGS code for different values of m (cf. Table 5.3)

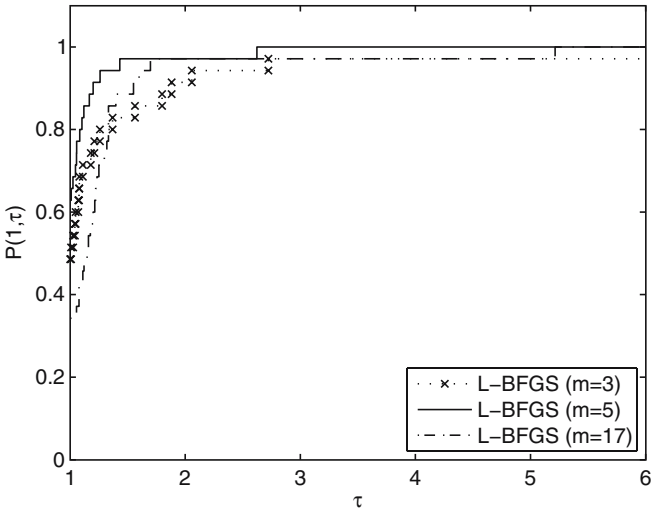

Fig. 5.9 Cumulative distribution ratio of CPU time: performance of the L-BFGS code for different values of m (cf. Table 5.3)

5.6 Notes

The strategy to build matrices H_k from the most recent m pairs (s_i, y_i) works well in practice. Small values of m result in low computing time, however the method is less robust then. On the other hand when m increases the cost of numerical algebra

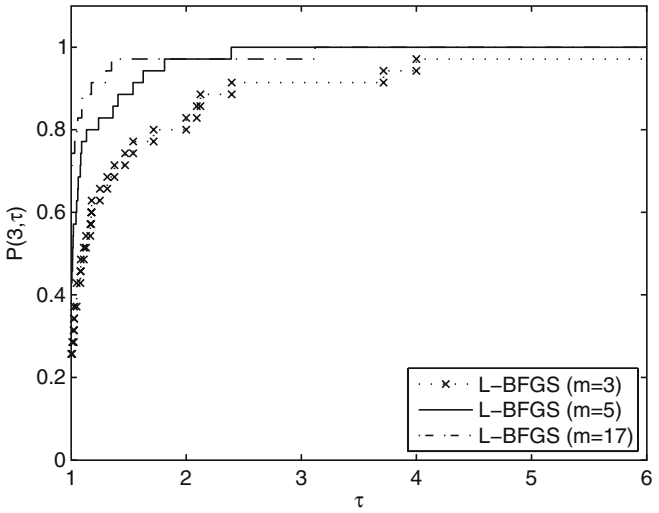

Fig. 5.10 Cumulative distribution ratio of the number of iterations: performance of the L-BFGS code for different values of m (cf. Table 5.3)

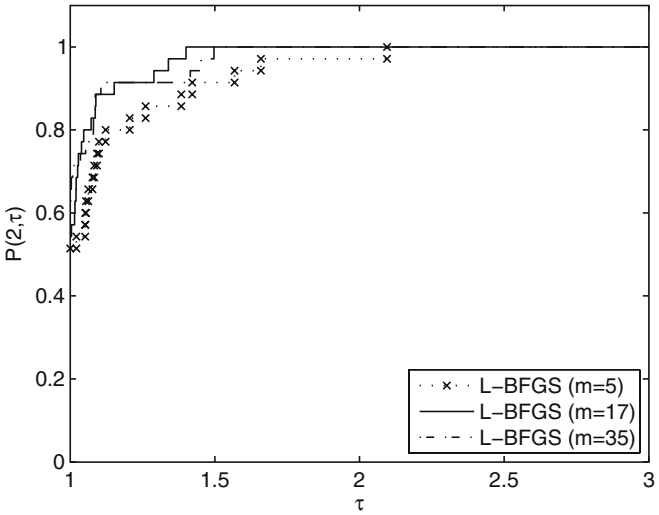

Fig. 5.11 Cumulative distribution ratio of the number of function evaluations: performance of the L-BFGS code for different values of m (cf. Table 5.4)

involved also increases so usually relatively low value of m is recommended as the default value. In general, the limited memory BFGS algorithm is regarded as the most efficient procedure for minimizing functions with large number of variables and which do not have any particular structure of the Hessian matrix. If, for example, the function f is partially separable (see [146] for the definition) then the Newton

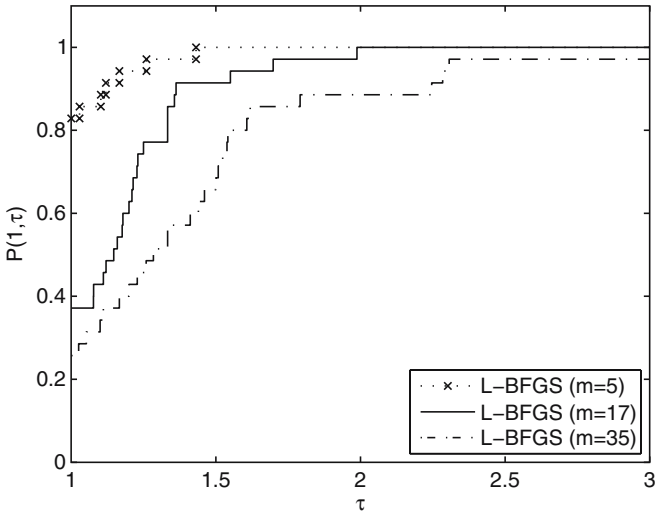

Fig. 5.12 Cumulative distribution ratio of CPU time: performance of the L-BFGS code for different values of m (cf. Table 5.4)

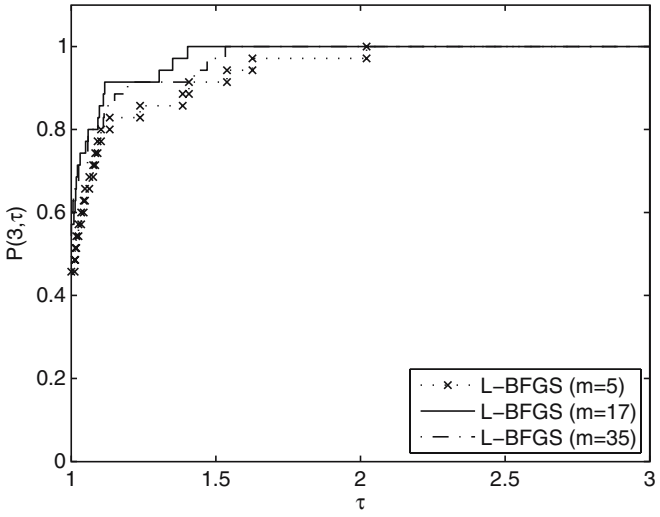

Fig. 5.13 Cumulative distribution ratio of the number of iterations: performance of the L-BFGS code for different values of m (cf. Table 5.4)

method outperforms the limited memory quasi-Newton method by the wide margin. However, the conjugate gradient algorithm is less efficient (in terms of the number of function evaluations) and less robust although the preconditioned conjugate gradient algorithm by Buckley and LeNir is regarded as the strong competitor.

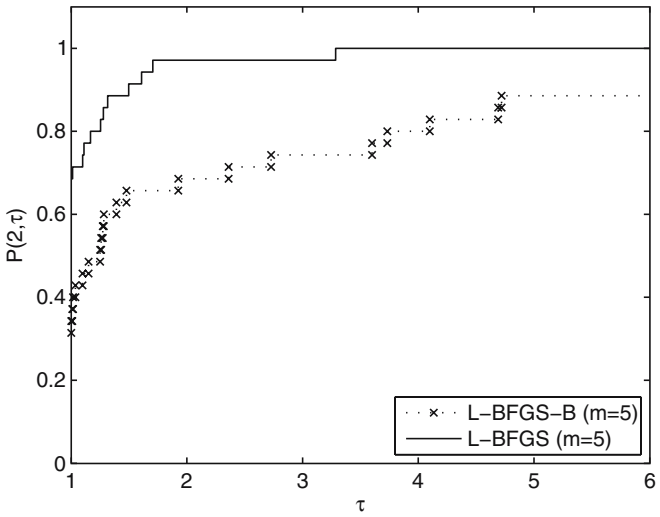

Fig. 5.14 Cumulative distribution ratio of the number of function evaluations: performance of the L-BFGS-B code against L-BFGS code, for $m = 5$ (cf. Tables 5.2 and 5.4)

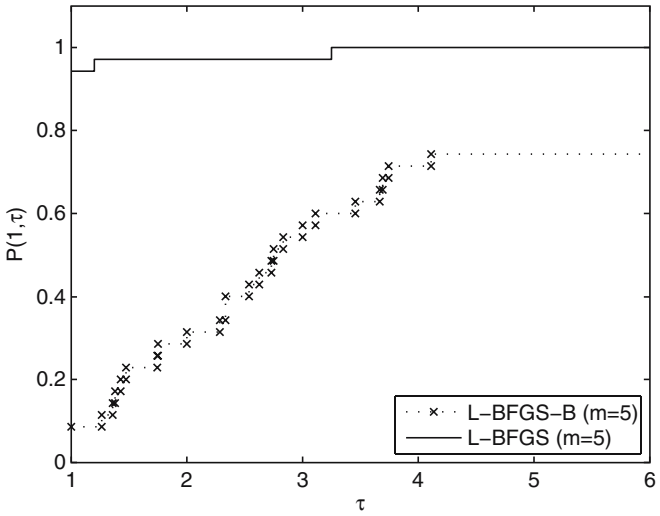

Fig. 5.15 Cumulative distribution ratio of CPU time: performance of the L-BFGS-B code against L-BFGS code, for $m = 5$ (cf. Tables 5.2 and 5.4)

Our presentation of convergence of the limited memory quasi-Newton algorithms is based on [29] where methods from the Broyden family are considered (except the DFP algorithm). Some earlier results on the convergence of the BFGS method are presented in [166]. In [145] Nocedal discusses self-correcting properties of the

BFGS update formula, he also explains reasons why the global convergence analysis does not cover, so far, the DFP method.

The compact representations of quasi-Newton matrices are introduced in [28] where SR1 matrices are also considered. [28] shows how to use the compact representations in constrained optimization and in solving nonlinear equations. Our presentation of the compact representations closely follows [28].

Chapter 6
The Method of Shortest Residuals and Nondifferentiable Optimization

6.1 Introduction

The method of shortest residuals is briefly discussed in Chap. 1. We show there that the method differs from a standard conjugate gradient algorithm only by scaling factors applied to conjugate directions. This is true when problems with quadratics are considered. However, these methods are quite different if applied to nonconvex functions.

A method of shortest residuals was discovered independently by Lemaréchal [118] and Wolfe [209]. Also Hestenes in his book ([100]) claims that the method tuned to quadratics was introduced by him. Lemaréchal and Wolfe were primarily interested in methods for minimizing nondifferentiable convex functions. Lemaréchal derives his method systematically from conjugate direction methods showing that all conjugate direction methods (or Davidon methods using his nomenclature) possess a property (discussed later) which is essential in constructing convergent algorithms for nondifferentiable problems. Wolfe constructs first method for general convex functions referring to a concept of the generalized gradient and then discovers, to his surprise, that his new method finds a solution to a quadratic problem in a number of iterations not exceeding the number of variables. Then, he formally proves that his algorithm is a version of a conjugate direction method.

The method of shortest residuals is thus strongly twinned with algorithms for nondifferentiable optimizations and discussing several versions of the method is not possible without incursions to a field of nondifferentiable optimization. Thus, our presentation of the method of shortest residuals starts with the outline of first algorithms for nondifferentiable convex functions and ends with the description of the globally convergent algorithm for nondifferentiable functions.

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

6.2 The Method of Conjugate Subgradient by Lemaréchal and Wolfe

Algorithms proposed by Lemaréchal and Wolfe have much in common, yet since the method of shortest residuals proposed for differentiable functions is more related to the Wolfe algorithm we will mostly refer to the second method.

The Wolfe algorithm is intended to solve the problem

$$\min_{x\in\mathscr{R}^n} f(x) \tag{6.1}$$

where f is a closed proper, convex function on $\mathscr{R}^n$, i.e. f is convex and lower semicontinuous and never assumes value $-\infty$, although $+\infty$ is allowed.

The following definitions are needed to present a method for the problem (6.1) – they are presented following [187] (cf. Appendix A).

The effective domain of f is the convex set:

$$\mathrm{dom}(f) = \{x \in \mathscr{R}^n : \ f(x) < +\infty\}.$$

For any $x \in \mathrm{dom}(f)$, $d \neq 0$ the directional derivative is defined by

$$f'(x;d) = \lim_{t\downarrow 0} \frac{f(x+td) - f(x)}{t}.$$

Definition 6.1. (i) A vector $g \in \mathscr{R}^n$ is the subgradient of f at x if

$$f(y) \geqslant f(x) + g^T(y-x)$$

for all $y \in \mathscr{R}^n$. The set of all subgradients at x is called the subdifferential of f at x and is denoted by $\partial f(x)$.

(ii) The ε–subgradient of f at x is any vector $g \in \mathscr{R}^n$ satisfying

$$f(y) \geqslant f(x) + g^T(y-x) - \varepsilon$$

for all $y \in \mathscr{R}^n$. The set of all ε–subgradients is called the ε–subdifferential of f at x and is denoted by $\partial_\varepsilon f(x)$.

[187] lists also the basic properties of a convex function in relation to the introduced definitions.

Lemma 6.1. *(i) $f'(x;d)$ is a convex function of d and is closed and proper if x belongs to the relative interior of* $\mathrm{dom}(f)$.

(ii) If x belongs to the relative interior of $\mathrm{dom}(f)$ *then $\partial f(x)$ is nonempty.*

(iii) $\partial f(x)$ is a closed convex set.

(iv) $\partial f(x)$ is nonempty and bounded if and only if x belongs to the relative interior of $\mathrm{dom}(f)$. *In this case $f'(x;d)$ is finite for all d and closed, i.e.*

$$\bar{f}'(x;d) = f'(x;d)$$

for all d. Here, $\bar{f}'(x;\cdot)$ is the closure of $f'(x;\cdot)$ which is the greatest lower semicontinuous function majorized by $f'(x;\cdot)$.

(v)

$$\bar{f}'(x;d) = \sup_{g\in\partial f(x)} g^T d.$$

(vi)

$$\sup_{g\in\partial_\varepsilon f(x)} g^T d = \inf_{t>0} \frac{f(x+td)-f(x)+\varepsilon}{t} \tag{6.2}$$

Since $\partial_\varepsilon f(x)$ is related to small changes of f in a neighborhood of x knowing it can be useful in stating optimality conditions for f at x.

Lemma 6.2. *Assume that $x \in \mathrm{dom}(f)$. Then*

$$f(x) - \inf_y f(y) \leqslant \varepsilon$$

if and only if

$$0 \in \partial_\varepsilon f(x).$$

Proof. From Definition 6.1

$$0 \in \partial_\varepsilon f(x) \Leftrightarrow f(y) \geqslant f(x) - \varepsilon$$

which gives the desired result. □

If $0 \notin \partial_\varepsilon f(x)$ then we can expect that a reduction of f in some direction is possible,

Lemma 6.3. *Assume that $x \in \mathrm{dom}(f)$ and $0 \notin \partial_\varepsilon f(x)$. Then there exists a direction d such that*

$$f(x) - \inf_{t\geqslant 0} f(x+td) > \varepsilon.$$

Proof. Let $\bar{d}$ be the unique vector from $\partial_\varepsilon f(x)$ with the smallest norm. Then, since $\partial_\varepsilon f(x)$ is closed, we have

$$\begin{aligned}
\min_{\|d\|\leqslant 1} \inf_{t>0} \frac{f(x+td)-f(x)+\varepsilon}{t} &= \min_{\|d\|\leqslant 1} \sup_{g\in\partial_\varepsilon f(x)} g^T d \\
&= \sup_{g\in\partial_\varepsilon f(x)} \min_{\|d\|\leqslant 1} g^T d \\
&= \sup_{g\in\partial_\varepsilon f(x)} (-\|g\|) \\
&= -\min_{g\in\partial_\varepsilon f(x)} \|g\|
\end{aligned} \tag{6.3}$$

$$= -\|\bar{d}\| < 0. \tag{6.4}$$

Equation (6.3) follows from the minimax theorem for a bilinear function on the product of two sets, one of which is bounded (cf. e.g. [187]).

Equations (6.2) and (6.4) imply that for

$$s = \frac{\bar{d}}{\|\bar{d}\|}$$

we have

$$f(x) - \inf_{t \geqslant 0} f(x+ts) > 0$$

which proves the lemma. □

Lemmas 6.2 and 6.3 form the basis for the algorithm which finds a minimum of a convex nondifferentiable function. ε–subgradient algorithm presented here is given in [11].

Algorithm 6.1. (The ε–subgradient method)

Parameter: $\xi \in (0,1)$.

1. Choose $x_1 \in \mathrm{dom}(f)$ and $\varepsilon_1 > 0$. Set $k = 1$.
2. Find the smallest nonnegative integer j such that $\varepsilon_{k+1} = \xi^j \varepsilon_k$ and

$$0 \notin \partial_{\varepsilon_{k+1}} f(x_k).$$

   If such j does not exist then STOP.
3. Define d_k as the solution to the problem

$$\min_{p \in \partial_{\varepsilon_{k+1}} f(x_k)} \|p\| \tag{6.5}$$

   by taking $d_k = -p_k$ where p_k solves the problem (6.5).
4. Set $x_{k+1} = x_k + \alpha_k d_k$, where $\alpha_k > 0$ is such that

$$f(x_k) - f(x_{k+1}) > \varepsilon_{k+1}.$$

   Got to Step 2.

The α_k specified in Step 4 can be found according to the rule

$$f(x_k + \alpha_k d_k) = \min_{\alpha > 0} f(x_k + \alpha d_k) \tag{6.6}$$

although that would require many function evaluations.

The convergence of the ε–subgradient method is given in the following theorem.

Theorem 6.1. *Suppose that* $\{x_k\}$ *is generated by* Algorithm 6.1. *Then*
(i)

$$\lim_{k\to\infty} f(x_k) = \inf_x f(x).$$

(ii) If in addition the set $\mathscr{N} = \{\bar{x} \in \mathscr{R}^n : f(\bar{x}) = \inf_x f(x)\}$ *is nonempty and bounded, every limit point of* $\{x_k\}$ *is in* $\mathscr{N}$ *and at least one such point exists.*

Proof. From Lemma 6.3 we know that Step 4 can be executed, i.e. there exists $\alpha_k > 0$ such that $x_{k+1} = x_k + \alpha_k d_k$ and

$$f(x_{k+1}) - f(x_k) < -\varepsilon_{k+1}$$

thus

$$f(x_k) - f(x_1) < -\sum_{j=2}^{k} \varepsilon_j.$$

The above inequality implies that either $\inf_y f(y) = -\infty$ and then $\lim_{k\to\infty} f(x_k) = -\infty$, or $\inf_y f(y) > -\infty$ and

$$\sum_{k=2}^{\infty} \varepsilon_k < \infty. \tag{6.7}$$

If (6.7) holds we also have $\varepsilon_k \to 0$ and there is an infinite number of iterations at which $\varepsilon_{k+1} < \varepsilon_k$. For these iterations

$$0 < f(x_k) - \inf_y f(y) \leqslant \varepsilon_k$$

and, since $\{f(x_k)\}$ is decreasing,

$$\lim_{k\to\infty} f(x_k) = \inf_y f(y).$$

To prove *(ii)* consider the set $\mathscr{M} = \{x \in \mathscr{R}^n : f(x) \leqslant f(x_1)\}$. We can show (see, e.g. [187]) that because $\mathscr{N}$ is nonempty and bounded so is the set $\mathscr{M}$. This implies that there exists at least one accumulation point of $\{x_k\}$ and from *(i)* we know that it belongs to $\mathscr{N}$. This proves *(ii)*. □

The ε–subgradient algorithm is conceptual method in the sense that neither the rule determining $\partial_\varepsilon f(x)$, nor a line search procedure with a finite termination are specified. In general, these two issues cannot be easily resolved. Thus great credit goes to Lemaréchal and Wolfe who show how to make the ε–subgradient algorithm implementable. Before we go to their propositions on resolving these issues we would like to point to a class of problems for which the evaluation of subdifferentials can be relatively easy task. We mention these problems also because their analysis precipitated the development of nondifferentiable optimization.

Consider the function defined as follows

$$f(x) = \max_{i \in I} f_i(x),$$

where I is some finite set. We can show (cf. e.g. [52]) that

$$\partial f(x) = \mathrm{co}\left(\{\nabla f_i(x) : \ i \in I(x)\}\right),$$

where

$$I(x) = \{i \in I : \ f_i(x) = f(x)\}.$$

Here, we assume that functions f_i are continuously differentiable for all $i \in I$ and $\mathrm{co}(\{v_j : \ j \in J\})$ is a convex hull spanned by vectors v_j, $j \in J$ (cf. Appendix A).

Suppose now that f is an arbitrary closed, proper convex function. Wolfe proposes the approximation to $\partial f(x_k)$ by a finite set of subgradients:

$$G_k = \left\{g_{k-p}, g_{k-p+1}, \ldots, g_k\right\},$$

where $g_{k-i} \in \partial f(x_{k-i})$, $i = 0, 2, \ldots, p$ and all x_{k-p}, x_{k-p+1},..., x_{k-1} are sufficiently close to x_k.

Assume that G_k is 'approximately' contained in $\partial f(x_k)$, then direction d_k is determined: first by finding the point of the smallest norm from the set $\mathrm{co}(G_k)$, which we denote by $\mathrm{Nr}\, G_k$, i.e.

$$\mathrm{Nr} G_k = p_k \tag{6.8}$$

and p_k solves the problem

$$\min_{p \in \mathrm{co}(G_k)} \|p\|; \tag{6.9}$$

secondly by taking

$$d_k = -p_k.$$

If our assumption

$$\mathrm{co}\,(G_k) \approx \partial f(x_k) \tag{6.10}$$

is valid and $\|\mathrm{Nr}\ G_k\|$ is small then, according to Lemma 6.2, x_k is a minimizer of f. One of the most important contribution of Lemaréchal and Wolfe is the scheme for building G_k in such a way that (6.10) holds. We will go back to the issue of approximating $\partial f(x_k)$ after stating the Wolfe algorithm.

Consider a line search procedure proposed by Wolfe. It seeks for $\alpha_k > 0$ such that

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu\, \alpha_k \|d_k\|^2 \tag{6.11}$$

$$g_k(\alpha_k)^T d_k \geqslant -\eta \|d_k\|^2, \tag{6.12}$$

where $g_k(\alpha) \in \partial f(x_k + \alpha d_k)$. The Wolfe line search rule mimics that stated by him for differentiable problems – in this case we seek for α_k which satisfies (6.11)–(6.12) with $0 < \mu < \eta < 1$. To see the relation with the Wolfe conditions note that in normal circumstances we have (this will be shown later) $g_k^T d_k = -\|d_k\|^2$ provided that f is differentiable.

We are now ready to state the Wolfe algorithm.

Algorithm 6.2. (The Wolfe conjugate subgradient algorithm)

Parameters: $\varepsilon,\ \delta,\ b > 0, 0 < \mu < \eta < 1$.

1. Choose $x_1 \in \mathscr{R}^n$, $G_1 = \{g_1\} \in \partial f(x_1)$, assume $s_1 = 0$ and set $k = 1$.
2. Calculate

$$d_k = -\mathrm{Nr}\, G_k.$$

   If $\|d_k\| \leqslant \varepsilon$ go to Step 3 (restart), otherwise go to Step 4.
3. If $s_k < \delta$ then STOP. Otherwise set $x_{k+1} = x_k$, $g_{k+1} = g_k$, $G_{k+1} = \{g_{k+1}\}$ and $s_{k+1} = 0$. Go to Step 2.
4. Find $\alpha_k > 0$ and $g_{k+1} \in \partial f(x_k + \alpha_k d_k)$ such that

$$\begin{array}{lll} & g_{k+1}^T d_k \geqslant -\eta \|d_k\|^2 & \text{and either} \\ \text{Case A} & f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu \alpha_k \|d_k\|^2, & \text{or} \\ \text{Case B} & \alpha_k \|d_k\| \leqslant b. & \end{array}$$

   In Case A, set $x_{k+1} = x_k + \alpha_k d_k$. In Case B, set $x_{k+1} = x_k$.
5. Choose $H_k \subset G_k$, set $G_{k+1} = H_k \bigcup \{-d_k, g_{k+1}\}$ and $s_{k+1} = s_k + \|x_{k+1} - x_k\|$. Increase k by one and go to Step 2.

It is not obvious how the Wolfe conjugate subgradient algorithm is related to ε–subgradient method stated in the previous section. This relation is established in Sect. 3 since we think that the detailed analysis of the Wolfe scheme goes beyond the scope of our main interests. Besides, neither the Wolfe algorithm, as it is stated here, is globally convergent, nor efficient algorithms for nondifferentiable optimization are simple modifications of the Wolfe method (see, e.g. Sect. 4, [110]).

Since in the next sections we will show how the Wolfe algorithm can be efficiently adopted to differentiable problems we have to show that the Wolfe method is a conjugate gradient algorithm when applied to quadratics. Furthermore, we want to establish important property of the method, i.e. that it restarts every finite number of iterations since it is the characteristics which has the impact on the performance of the method.

First, we discuss the second issue.

Lemma 6.4. *Suppose that there exists $C < +\infty$ such that for all $x \in \mathrm{dom}(f)$, $\|g\| \leqslant C$ where $g \in \partial f(x)$. Then, the Wolfe conjugate subgradient algorithm resets every a finite number of iterations.*

Proof. We have

$$\begin{aligned}\|d_{k+1}\| &= \left\|\mathrm{Nr}\left\{H_k \bigcup \{-d_k, g_{k+1}\}\right\}\right\| \\ &\leqslant \|\mathrm{Nr}\,\{-d_k, g_{k+1}\}\|.\end{aligned}$$

Finding Nr $\{-d_k, g_{k+1}\|$ requires solving a simple univariate problem:

$$\|\mathrm{Nr}\,\{-d_k, g_{k+1}\}\|^2 = \min_{0\leqslant\theta\leqslant 1} \|-\theta d_k + (1-\theta)g_{k+1}\|^2. \tag{6.13}$$

Assume that θ is not restricted, then its solution is given by

$$\theta_{min} = \frac{\|g_{k+1}\|^2 + g_{k+1}^T d_k}{\|g_{k+1}\|^2 + 2g_{k+1}^T d_k + \|d_k\|^2}. \tag{6.14}$$

It is convenient to represent the numerator and denominator in (6.14) by θ_1 and $\theta_1 + \theta_2$ where

$$\theta_1 = \|g_{k+1}\|^2 + g_{k+1}^T d_k \tag{6.15}$$

$$\theta_2 = \|d_k\|^2 + g_{k+1}^T d_k. \tag{6.16}$$

Using θ_1 and θ_2 we can express the optimal value in (6.13) as

$$\|d_{k+1}\|^2 = \|g_{k+1}\|^2 - \frac{\theta_1^2}{\theta_1 + \theta_2} \tag{6.17}$$

and

$$\theta_{min} = \frac{\theta_1}{\theta_1 + \theta_2},$$

if θ is unrestricted. The solution is well-defined since

$$\theta_1 + \theta_2 = \|g_{k+1} + d_k\|^2 > 0 \tag{6.18}$$

which follows from the fact that $d_k + g_{k+1} \neq 0$ – otherwise we would have $g_{k+1}^T d_k = -\|d_k\|^2$ which is the contradiction to (6.12).

Notice that if $\theta_1 \leqslant 0$ (in particular $\theta_1 = 0$ – the case which due to (6.17) requires special analysis) then the solution to (6.13) is $\theta_{min} = 0$ which implies that

$$\|d_{k+1}\|^2 = \|g_{k+1}\|^2 \leqslant -d_k^T g_{k+1} \leqslant \eta\|d_k\|^2 \tag{6.19}$$

which follows from the definition of θ_1 and the curvature condition.

Observe that

$$\theta_1 - \theta_2 = \frac{\theta_1^2 - \theta_2^2}{\theta_1 + \theta_2} = \|g_{k+1}\|^2 - \|d_k\|^2$$

which together with (6.12) enable us to transform (6.17) into

$$\begin{aligned}\|d_{k+1}\|^2 &= \|d_k\|^2 - \frac{\theta_2^2}{\theta_1+\theta_2}\\ &\leqslant \|d_k\|^2 - (1-\eta)^2\frac{\|d_k\|^2}{\theta_1+\theta_2}.\end{aligned} \tag{6.20}$$

Furthermore, we have

$$\theta_1+\theta_2 = \|g_{k+1}\|^2 + 2d_k^T g_{k+1} + \|d_k\|^2 \leqslant 4C^2$$

which together with (6.19) and (6.20) lead to

$$\|d_{k+1}\|^2 \leqslant \|d_k\|^2 \max\left[\eta, 1-\frac{(1-\eta)^2}{4C^2}\right].$$

Introduce now the number $0<\xi<1$ defined by

$$\xi = \max\left[\eta, 1-\frac{(1-\eta)^2}{4}\right].$$

It follows that the resetting occurs at least every

$$\frac{\log C^2 - \log \varepsilon^2}{\log \xi}$$

iterations. □

Lemma 6.4 states that after a finite number of iterations the norm of d_k is smaller than a prespecified threshold level. Figure 6.1 illustrates the norm reduction of d vector.

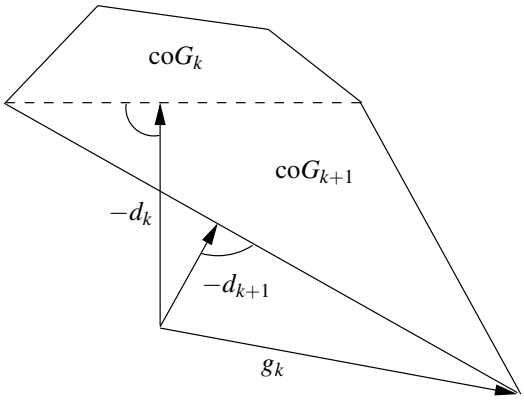

Fig. 6.1 The norm reduction of d vector

If at the same time all points x_j from the previous resetting iterations are close to x_k then we can claim that the zero vector 'almost' belongs to the subdifferential of f at x_k which says that necessary optimality conditions are 'approximately' satisfied at x_k. The full convergence analysis of Algorithm 6.2, without resorting to 'approximate' considerations, we show in Sect. 3.

Now concentrate on the second important issue related to the Wolfe conjugate subgradient algorithm. Is the method a conjugate direction algorithm when applied to a quadratic function? In order to show that we need two auxiliary lemmas which, although technical, put additional light on the specifics of the Wolfe algorithm.

Let

$$P = \{v_1, v_2, \ldots, v_m\}, \tag{6.21}$$

then we have the following optimality conditions for x to be Nr P.

Lemma 6.5. *If P is defined by (6.21), then $x \in$ co (P) is* Nr P *if and only if*

$$x^T v_j \geqslant \|x\|^2 \tag{6.22}$$

for all $1 \leqslant j \leqslant m$.

Proof. Since co (P) is a convex set $y(\theta) = x + \theta(y - x) \in$ co (P) for all $0 \leqslant \theta \leqslant 1$. We will show that $y(\theta)$ cannot be Nr P for any $0 < \theta \leqslant 1$ if (6.22) holds. We have

$$\begin{aligned} \|y(\theta)\|^2 &= (1-\theta)^2\|x\|^2 + 2\theta(1-\theta)y^T x + \theta^2\|y\|^2 \\ &= \|x\|^2 + 2\theta\left(x^T y - \|x\|^2\right) + \theta^2\|y - x\|^2. \end{aligned} \tag{6.23}$$

From (6.23) $\|y(\theta)\|^2 < \|x\|^2$ for small values of θ unless (6.22) holds. Notice that y can be any v_j, $j = 1, 2, \ldots, m$ and x is supposed to have the smallest norm among all vectors from co (P). □

The optimality conditions (6.22) can be equivalently written as $\|v_j\| \cos\theta_j \geqslant \|x\|$ where θ_j is the angle between vectors v_j and x. Figure 6.2 gives the geometric interpretation of the optimality conditions. In Fig. 6.3 vector x does not satisfy optimality conditions ($\|x\| > \|v_1\| \cos\theta_1$ and $\|x\| > \|v_4\| \cos\theta_4$) thus we can construct vector $x + \theta(v_1 - x)$ with small θ which belongs to P and has smaller norm than x.

The next lemma establishes very useful fact concerning the Wolfe algorithm. It says that if all subgradients $\{g_j\}_1^k$ are mutually orthogonal then the algorithm will generate the same directions irrespective of the choice of H_k. In particular the performance of the Wolfe method does not change if the only information from the previous iterations is the vector d_{k-1}.

Lemma 6.6. *Suppose that vectors $v_1, v_2, \ldots, v_k$ are nonnull and mutually orthogonal. Let $u =$ Nr $\{v_1, v_2, \ldots, v_{k-1}\}$ and $w =$ Nr $\{v_1, v_2, \ldots, v_k\}$, then*

$$\begin{aligned} u^T v_j &= \|u\|^2 \neq 0, \ j = 1, \ldots, k-1 \\ w &= \text{Nr } \{u, v_k\}. \end{aligned}$$

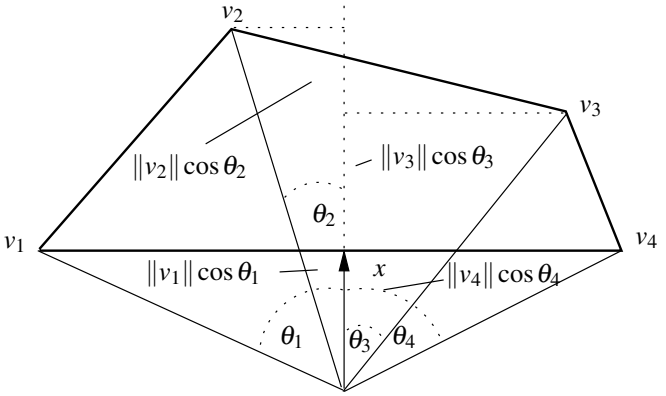

Fig. 6.2 Geometry of the optimality conditions (6.22)

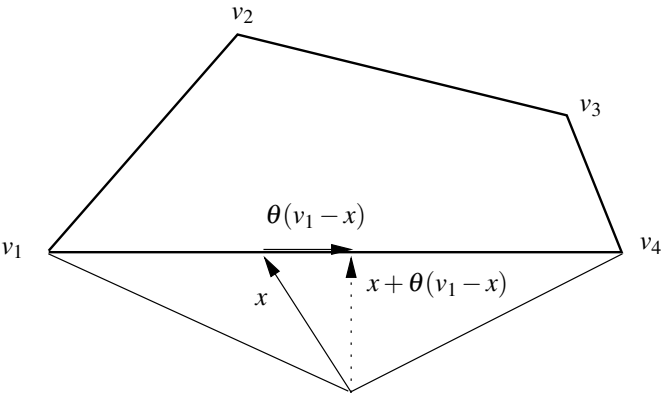

Fig. 6.3 Example of x which does not satisfy (6.22)

Proof. Due to the definition of u

$$u=\sum_{j=1}^{k-1}\lambda_j v_j,\ \ \sum_{j=1}^{k-1}\lambda_j=1, \tag{6.24}$$

$\lambda_j \geqslant 0$, $j=1,2,\ldots,k-1$ and from Lemma 6.5 we also have

$$v_j^T u \geqslant \|u\|^2,\ \ j=1,2,\ldots,k-1. \tag{6.25}$$

Since $\{v_j\}_1^{k-1}$ are mutually orthogonal,

$$u^T v_j=\lambda_j\|v_j\|^2,\ j=1,2,\ldots,k-1 \tag{6.26}$$

and $\|v_j\|^2 \neq 0,\ j = 1,2,\ldots,k-1$ due to our assumptions. From that and (6.24), (6.26) we must have $\|u\| \neq 0$. Furthermore, the following holds

$$\sum_{j=1}^{k-1} \lambda_j \left(u^T v_j - u^T u\right) = \|u\|^2 - \|u\|^2 = 0$$

which implies, from (6.24) and (6.25), that

$$u^T v_j = \|u\|^2,\ j = 1,2,\ldots,k-1$$

and, from (6.26), that

$$\lambda_j = \frac{\|u\|^2}{u^T v_j},\ j = 1,2,\ldots,k-1. \tag{6.27}$$

Analogously, we can write

$$w = \sum_{j=1}^{k} \nu_j v_j,\ \ \sum_{j=1}^{k} \nu_j = 1,\ \ \nu_j \geqslant 0,\ j = 1,2,\ldots,k \tag{6.28}$$
$$\nu_j = \frac{\|w\|^2}{w^T v_j},\ j = 1,2,\ldots,k.$$

Taking this and (6.27) into account we rewrite (6.28) as

$$\begin{aligned} w &= \sum_{j=1}^{k-1} \nu_j v_j + \nu_k v_k \\ &= \sum_{j=1}^{k-1} \frac{w^T v_j}{\|v_j\|^2} v_j + \frac{w^T v_k}{\|v_k\|^2} v_k \\ &= \sum_{j=1}^{k-1} \frac{\|w\|^2}{\|v_j\|^2} v_j + \frac{\|w\|^2}{\|v_k\|^2} v_k \\ &= \sum_{j=1}^{k-1} \left(\frac{\|w\|^2}{\|u\|^2}\right) \lambda_j v_j + \left(\frac{\|w\|^2}{\|v_k\|^2}\right) v_k \\ &= \left(\frac{\|w\|^2}{\|u\|^2}\right) u + \left(\frac{\|w\|^2}{\|v_k\|^2}\right) v_k. \end{aligned}$$

Notice that

$$\frac{\|w\|^2}{\|u\|^2} + \frac{\|w\|^2}{\|v_k\|^2} = \sum_{j=1}^{k-1} \frac{\|w\|^2}{\|u\|^2} \lambda_j + \frac{\|w\|^2}{\|v_k\|^2} = 1.$$

Therefore, $w \in \mathrm{co}(\{u, v_k\})$ and since $u^T v = 0$ we have $w^T v_k = w^T u = \|w\|^2$ thus $w = \mathrm{Nr}\,\{u, v_k\}$. $\square$

Having Lemma 6.6 we can show that the Wolfe conjugate subgradient algorithm is the shortest residuals version of the general conjugate gradient algorithm when applied to the quadratic

$$f(x) = \frac{1}{2}x^T Ax - b^T x. \tag{6.29}$$

Adaptation of the Wolfe algorithm to quadratic problems is as follows.

Algorithm 6.3. (The Wolfe conjugate subgradient algorithm for quadratics)

1. Choose $x_1 \in \mathscr{R}^n$, set $G_1 = \{g_k\}$ and $k = 1$.
2. Calculate

$$d_k = -\mathrm{Nr}\, G_k.$$

   If $d_k^T A d_k \leqslant 0$ then STOP.
3. Set

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T A d_k}$$
$$g_{k+1} = g_k + \alpha_k A d_k. \tag{6.30}$$

   Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
4. If $\|g_{k+1}\| = 0$ then STOP. Otherwise choose $H_k \subset G_k$ and set $G_{k+1} = H_k \bigcup \{g_{k+1}, -d_k\}$. Increase k by one and go to Step 2.

In Step 3 if $d_k^T A d_k \leqslant 0$ then $f(x_k + \alpha d_k) \to -\infty$ as $\alpha \to \infty$. Otherwise α_k is the minimum point of $\phi(\alpha) = f(x_k + \alpha d_k)$.

Theorem 6.2. *Let f be the quadratic function (6.29), and let $\{x_k\}$, $\{d_k\}$ be sequences generated by* Algorithm 6.3. *Then, after a finite number of iterations either $g_k = 0$, or $\lim_{\alpha\to\infty} f(x_k + \alpha d_k) = -\infty$.*

Proof. Suppose that at every iteration we choose $H_k = G_k$. This implies that d_k is determined according to the rule

$$d_k = -\mathrm{Nr}\,\{g_1, g_2, \ldots, g_k\}$$

since all previous d_j, $j = 1, 2, \ldots, k-1$ are convex combinations of $g_1, g_2, \ldots, g_{k-1}$.

First, we prove that vectors $g_1, g_2, \ldots, g_m$ are mutually orthogonal where m is the last iteration performed by the algorithm. The proof is by the induction on k. This is obviously true for $k = 1$, since $g_2^T d_1 = 0$ (from Step 3) and $d_1 = -g_1$.

If vectors $g_1, g_2, \ldots, g_k$ are mutually orthogonal then, due to Lemma 6.6, we have

$$g_i^T d_j = -\|d_j\|^2$$

for $1 \leqslant i \leqslant j \leqslant k$, so that

$$d_j^T (g_{i+1} - g_i) = \alpha_i d_j^T A d_i = 0$$

for all $i < j \leqslant k$. Therefore, $d_1, d_2, \ldots, d_k$ are conjugate directions. In Chap. 1 we have shown that the consequence of that and the condition $g_{k+1}^T d_k = 0$ is

$$g_{k+1}^T d_j = 0$$

for all $1 \leqslant j \leqslant k$. Since, due to Lemma 6.6,

$$d_j = \sum_{i=1}^{j} \lambda_i^j g_i, \quad \lambda_j^j \neq 0$$

for all $1 \leqslant j \leqslant k$, we can write

$$\begin{aligned} 0 = g_{k+1}^T d_j &= g_{k+1}^T \left(\sum_{i=1}^{j} \lambda_i^j g_i \right) \\ &= \lambda_j^j g_{k+1}^T g_j + \sum_{i=1}^{j-1} \lambda_i^j g_{k+1}^T g_i \end{aligned}$$

and the last sum is equal to zero. This proves that $g_1, g_2, \ldots, g_{k+1}$ are mutually orthogonal.

In order to complete the proof we will show that the choice of $H_k \subset G_k$ is irrelevant. For any $H_k \subset G_k$ we have

$$\begin{aligned} \mathrm{Nr}\left\{ H_k \bigcup \{g_{k+1}, -d_k\} \right\} &= \mathrm{Nr}\{g_{k+1}, -d_k\} \\ &= \mathrm{Nr}\{g_1, g_2, \ldots, g_{k+1}\} \\ &= d_{k+1} \end{aligned}$$

since co $(\{g_{k+1}, -d_k\}) \subseteq$ co $(G_{k+1}) \subseteq$ co $(\{g_1, g_2, \ldots, g_{k+1}\})$. Now, it remains to notice that with $H_k = \emptyset$ we have the method of shortest residuals which is a conjugate gradient algorithm as we have proved in Chap. 1. □

6.3 Conjugate Subgradient Algorithm for Problems with Semismooth Functions

Algorithm 6.2 is not globally convergent. It does not contain any mechanism for controlling tolerance parameters ε and δ which should approach zero if x_k tends to a stationary point. One such mechanism is proposed by Mifflin who also extends the Wolfe algorithm in various ways in order to improve its efficiency and make

it applicable to a broader class of problems, not necessarily described by convex functions.

The Mifflin's algorithm is as follows.

Algorithm 6.4. (The Mifflin conjugate subgradient algorithm)

Parameters: positive a_1, a_2, b_1, b_2, μ η such that $a_2 < a_1$, $b_2 \leqslant b_1 < 1$, $\mu < \eta < 1$.

1. Choose $x_1 \in \mathscr{R}^n$. Set $G_1 = \{g_1\} \in \partial f(x_1)$, $\delta_1 = \|g_1\|$ and $k = 1$.
2. Calculate

$$d_k = -\text{Nr}\, G_k.$$

   If $\|d_k\| \leqslant b_2\delta_k$ then replace δ_k by $b_1\delta_k$ and go to Step 3. Otherwise replace δ_k by $\min[\delta_k, \|d_k\|]$ and go to Step 4.
3. Delete $g_j \in \partial f(y_j)$ from G_k if

$$\|x_k - y_j\| > a_1\delta_k.$$

4. Find positive α_R, α_L and $g_R \in \partial f(x_k + \alpha_R d_k)$ such that

$$\alpha_R - \alpha_L \leqslant a_2\delta_{k-1}\frac{1}{\|d_k\|} \tag{6.31}$$

$$f(y_L) - f(x_k) \leqslant -\mu\alpha_L\|d_k\|^2 \tag{6.32}$$

$$g_R^T d_k \geqslant -\eta\|d_k\|^2 \tag{6.33}$$

   where $y_L = x_k + \alpha_L d_k$, $y_R = x_k + \alpha_R d_k$.
5. Set $G_{k+1} = G_k \bigcup \{g_R\}$. Substitute y_L for x_{k+1}, δ_k for δ_{k+1}, increase k by one and go to Step 2.

Mifflin suggests using also $g_L \in \partial f(y_L)$ to improve the approximation of $\partial f(x_k)$ so in Step 5 we would have $G_{k+1} = G_k \bigcup \{g_L, g_R\}$. Notice that if we are able to find $\alpha_L = \alpha_R$ which satisfy (6.31)–(6.32) then we have the modified Wolfe directional minimization rule. However, when f is not convex this is not possible in general.

Another difference concerns the way subgradients from previous iterations are dropped from G_k. They are dropped selectively depending on the distance from the current point to the points at which they are evaluated. This dropping scheme is more selective in comparison to the scheme employed in the Wolfe algorithm where G_k is set to a single subgradient at the resetting iteration.

Algorithm 6.4 is intended for nondifferentiable optimization problems with not necessary convex functions. They are subclass of locally Lipschitz functions as defined in [32] (cf. Appendix A).

Let B be an open subset of $\mathscr{R}^n$ and $f : \mathscr{R}^n \to \mathscr{R}$ be Lipschitz continuous on B, i.e. there exists $L > 0$ such that

$$|f(y) - f(x)| \leqslant L\|y - x\| \tag{6.34}$$

for all $x, y \in B$. If f is Lipschitz continuous on each bounded subset of $\mathscr{R}^n$ then we say that f is locally Lipschitz.

As it is shown in [32] for a locally Lipschitz function f we can introduce the generalized gradient of f which plays in optimization a role similar to that of the subdifferential in case of convex functions.

Let $x \in B$ (as used in (6.34)) and $d \in \mathscr{R}^n$. Define

$$f^0(x;d) = \limsup_{h \to 0,\ t \downarrow 0} \frac{f(x+h+td) - f(x+h)}{t}$$

then the generalized gradient of f at x (or the Clarke subdifferential) is given by

$$\partial_C f(x) = \left\{ x \in \mathscr{R}^n :\ g^T d \leqslant f^0(x;d) \text{ for all } d \in \mathscr{R}^n \right\}.$$

(whenever it does not lead to confusion we use $\partial f(x)$ instead of $\partial_C f(x)$).

The following properties of $f^0(\cdot;\cdot)$ and $\partial_C f(\cdot)$ will be used in this section (cf. Appendix A).

Proposition 6.1. *If f is a locally Lipschitz function then*

(i) $f^0(x;d) = \max\{g^T d :\ g \in \partial f(x)\}$.

(ii) If $\{x_k\}$ converges to x and $g_k \in \partial f(x_k)$ for each k then $\|g_k\| \leqslant L$ (cf. (6.34)) and each accumulation point g of $\{g_k\}$ satisfies $g \in \partial f(x)$, i.e. ∂f is bounded on bounded subsets of B and is upper semicontinuous on B.

If the limit

$$\lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t}$$

exists it is denoted by $f'_+(x;d)$ and is called the directional derivative of f at x in the direction d. We are interested in these locally Lipschitz functions for which directional derivatives exist. One such class of functions was introduced by Mifflin [133].

Definition 6.2. f is semismooth at x if

(i) f is Lipschitz continuous on a ball around x and

(ii) for each $d \in \mathscr{R}^n$ and for any sequences $\{t_k\} \subset \mathscr{R}_+ (\mathscr{R}_+ = \{x \in \mathscr{R} :\ x > 0\})$, $\{\theta_k\} \subset \mathscr{R}^n$ and $\{g_k\} \subset \mathscr{R}^n$ such that $t_k \downarrow 0$, $\theta_k / t_k \to 0 \in \mathscr{R}^n$ and $g_k \in \partial f(x_k + t_k d + \theta_k)$, the sequence $\{g_k^T d\}$ has exactly one accumulation point.

The crucial property of semismooth functions is given next.

Proposition 6.2. *If f is semismooth at x then for each $d \in \mathscr{R}^n$, $f'_+(x;d)$ exists and equals* $\lim_{k \to \infty} g_k^T d$ *where $\{g_k\}$ is any sequence as in* Definition 6.2.

We also have

Proposition 6.3. *If $f : \mathscr{R}^n \to \mathscr{R}$ is convex, then f is locally Lipschitz, $\partial_C f(x) = \partial f(x)$ and f is semismooth on $\mathscr{R}^n$.*

Slightly weaker property of a locally Lipschitz function, than that which defines semismooth functions, is in fact needed to guarantee convergence of the Mifflin algorithm.

Definition 6.3. $f : \mathscr{R}^n \to \mathscr{R}$ is weakly upper semismooth at x if

(i) f is Lipschitz continuous on a ball around x and
(ii) for each $d \in \mathscr{R}^n$ and for any sequences $\{t_k\} \subset \mathscr{R}_+$ and $\{g_k\} \subset \mathscr{R}^n$ such that $t_k \downarrow 0$, $\theta_k/t_k \to 0 \in \mathscr{R}^n$ and $g_k \in \partial f(x_k + t_k d)$ it follows that

$$\liminf_{k\to\infty} g_k^T d \geqslant \limsup_{t_k \downarrow 0} \frac{f(x+t_k d) - f(x)}{t}.$$

Throughout the remainder of this section we will deal with weakly upper semismooth functions. This assumption is mainly needed for the step-size selection procedure, described below, to guarantee that it terminates with the desired result after a finite number of operations.

Algorithm 6.5. (The Mifflin step-size selection procedure)

1. Set $\alpha_L = 0$, $\alpha_N = +\infty$, $\alpha_R = +\infty$, $m = 1$ and choose $\alpha_m > 0$.
2. If α_m satisfies (6.11) set $\alpha_L = \alpha_m$. Otherwise set $\alpha_N = \alpha_m$.
   If α_m satisfies (6.12) set $\alpha_R = \alpha_m$.
3. If $\alpha_R - \alpha_L \leqslant a_2 \delta_{k-1}/\|d_k\|$ then set $y_L = x_k + \alpha_L d_k$, $g_L \in \partial f(x_k + \alpha_L d_k)$, $y_R = x_k + \alpha_R d_k$, $g_R \in \partial f(x_k + \alpha_R d_k)$ and STOP. If $\alpha_N = +\infty$ set α_{m+1} to $2\alpha_m$, otherwise set α_{m+1} to $\frac{1}{2}(\alpha_L + \alpha_N)$ and increase m by one. Go to Step 2.

Lemma 6.7. *Assume that function f is weakly upper semismooth on $\mathscr{S} \subset \mathscr{R}^n$ where $\mathscr{S}$ is the set of all points in $\mathscr{R}^n$ lying within the Euclidean distance of $a_2\|g_1\|$ of*

$$\mathscr{M} = \{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\}.$$

Furthermore, suppose that $x_k \in \mathscr{M}$, $\|d_k\| \neq 0$, $\delta_{k-1} > 0$ and $0 < \mu < \eta < 1$. Then the Mifflin step-size selection procedure either terminates with α_L, α_R and $g_R \in \partial f(x_k + \alpha_L d_k)$ satisfying (6.31)–(6.33), or generates a sequence $\{\alpha_m\}$ such that $\alpha_m \to_{m\to\infty} \infty$ and $f(x_k + \alpha_m d_k) \to_{m\to\infty} -\infty$.

Proof. The proof of the lemma is essentially identical to that of Lemma 2.3. We will focus only on this part of the proof which cannot be straightforwardly copied from the proof of Lemma 2.3.

Suppose that bisection has begun but the search does not terminate. We have that the interval $[\alpha_L, \alpha_N]$ shrinks to the point $\hat{\alpha}$ such that

$$f(x_k + \hat{\alpha} d_k) - f(x_k) \leqslant -\mu \hat{\alpha} \|d_k\|^2. \tag{6.35}$$

Since α_N satisfies (6.11) and $\alpha_N \downarrow \hat{\alpha}$ α_N must take on an infinite number of distinct values greater than $\hat{\alpha}$. Furthermore, α_N fulfill

$$g(\alpha_N)^T d_k \leqslant -\eta \|d_k\|^2,$$

infinitely many times since otherwise the procedure would stop. Thus, we have

$$\liminf_{\alpha_N \downarrow \hat{\alpha}} g(\alpha_N)^T d_k \leqslant -\eta \|d_k\|^2,$$

where $g(\alpha_N) \in \partial f(x_k + \alpha_N d_k)$.

On the other hand, we also have

$$f(x_k + \alpha_N d_k) - f(x_k) > -\mu \alpha_N \|d_k\|^2 \tag{6.36}$$

which combined with (6.35) gives

$$f(x_k + \alpha_N d_k) - f(x_k + \hat{\alpha} d_k) > -\mu (\alpha_N - \hat{\alpha}) \|d_k\|^2.$$

Now we use the fact that f is weakly upper semismooth and $g(\alpha_N) \in \partial f(x_k + \hat{\alpha} d_k + (\alpha_N - \hat{\alpha}) d_k)$ to get

$$\begin{aligned} \liminf_{\alpha_N \downarrow \hat{\alpha}} g(\alpha_N)^T d_k &\geqslant \limsup_{\alpha_N \downarrow \hat{\alpha}} \frac{f(x_k + \alpha_N d_k) - f(x_k + \hat{\alpha} d_k)}{\alpha_N - \hat{\alpha}} \\ &\geqslant -\nu \|d_k\|^2 \end{aligned}$$

which contradicts (6.36) since $\mu < \eta$ and $\|d_k\| \neq 0$. □

The crucial result concerning the global convergence of the Mifflin conjugate subgradient algorithm is stated in the next lemma.

Lemma 6.8. *Suppose that there exists a positive number C such that*

$$\|g\| \leqslant C$$

for all $g \in \partial f(y)$ and $y \in \mathscr{S}$ where $\mathscr{S}$ is introduced in the assumption of Lemma 6.7. *Then, either $\delta_k \downarrow 0$, or $f(x_k) \to -\infty$.*

Proof. The proof of the lemma is given in [133]. It refers to the following property of the direction finding rule

$$\|d_{k+1}\|^2 \leqslant \|d_k\|^2 \max\left[\eta, 1 - \frac{(1-\eta)^2}{4C^2}\right]$$

which can be easily established if we notice that

$$\begin{aligned} \|d_{k+1}\| &= \|\mathrm{Nr}\, G_{k+1}\| = \left\|\mathrm{Nr}\, \left\{G_k \bigcup \{g_R\}\right\}\right\| \\ &\leqslant \|\mathrm{Nr}\, \{-d_k, g_R\}\| \end{aligned}$$

under the assumption that no g_j are deleted from G_k. The proof is based on a contradiction which leads to the conclusion that subgradients g_j are not deleted from sets G_k. □

The global convergence analysis of the algorithm requires new optimality conditions for nondifferentiable problems since now we are dealing with locally Lipschitz functions. The following lemma is due to Clarke [32].

Lemma 6.9. *Suppose that f is a locally Lipschitz function and that $\bar{x}$ is a solution of the problem*

$$\min_{x \in \mathscr{R}^n} f(x),$$

then

$$0 \in \partial f(\bar{x}). \tag{6.37}$$

The global convergence result concerning the algorithm shows that every accumulation of the sequence $\{x_k\}$ is a stationary point in the sense of Lemma 6.9.

Theorem 6.3. *Suppose that the assumptions of* Lemma 6.7 *hold and that the set $\mathscr{M}$ is bounded. Then every accumulation point (and at least one such exists) of a sequence $\{x_k\}$ generated by the Mifflin conjugate subgradient algorithm satisfies (6.37).*

Proof. Since $\mathscr{M}$ is bounded, $\partial f(y)$ are bounded for $y \in \mathscr{M}$. It follows that the assumption of Lemma 6.8 is satisfied.

Since $g_j \in \partial f(y_j) \in \mathscr{R}^n$, from the Caratheodory's theorem, there exists $p_k \leqslant n+1$ such that

$$\mathrm{G_k} \subset \mathrm{co}\left(\bigcup_{j=1}^{p_k} g_j^k\right) \subset \mathrm{co}\left(\bigcup_{j=1}^{p_k} \partial f(y_j^k)\right),$$

where $g_j^k \in G_k$ and y_j^k are such that $g_j^k \in \partial f(y_j^k)$. Since $\mathscr{M}$ is bounded and all $x_k \in \mathscr{M}$ there exists a subsequence $\{x_k\}_{k \in K}$ which has an accumulation point. Suppose that $x_{k \to \infty, k \in K} = \bar{x}$. We can extract a subsequence $\{x_k\}_{k \in K_1}$ from $\{x_k\}_{k \in K}$ ($K_1 \subset K$) and find a positive integer $p \leqslant n+1$ such that

$$\mathrm{Nr}G_k \in \mathrm{co}\left(\bigcup_{j=1}^{p} \partial f(y_j^k)\right) \tag{6.38}$$

for all $k \in K_1$.

The crucial point in the proof is to observe that the mapping $R : \mathscr{M}^p \to \mathscr{R}^n$ defined by

$$R(y_1, y_2, \ldots, y_p) = \mathrm{co}\left(\bigcup_{l=1}^{p} \partial f(y_l)\right)$$

is upper semicontinuous which follows from Proposition 6.1 and the fact that $\mathscr{M}$ is bounded.

On the other hand, from the subgradient deletion rule we have

$$\|x_k - y_l^k\| \leqslant a_1 \delta_k$$

for all $l = 1, 2, \ldots, p$ and $k \in K_1$. Lemma 6.8 and the assumption of the theorem state that $\delta_k \downarrow 0$, thus

$$y_l^k \to_{k\to\infty,\ k\in K_1} \bar{x} \tag{6.39}$$

for all $l = 1, 2, \ldots, p$. Furthermore, due to the restriction $\|\mathrm{Nr}G_k\| \leqslant b_2\delta_k$, we also have

$$\lim_{k\to\infty,\ k\in K_1} \|\mathrm{Nr}G_k\| = 0 \tag{6.40}$$

which results in

$$0 \in \mathrm{co}\left(\bigcup_{l=1}^{p} \partial f(\bar{x})\right) = \partial f(\bar{x})$$

since (6.38), (6.39)–(6.40) hold, the mapping R is semicontinuous and the set $\partial f(\bar{x})$ is convex. □

Mifflin's algorithm is in fact an implementable version of the conjugate subgradient algorithm introduced in the beginning of the chapter. The main drawback of the latter algorithm is that it requires the full specification of the subdifferential at a given point while the former provides the approximation based on a finite number of subgradients evaluated at some previous iterations. Mifflin's algorithm when applied to solve problems described by smooth functions will have a direction rule determined not only by the current gradient but also gradients calculated at previous iterates. It is not surprising that the scheme is related to a conjugate gradient algorithm which gains its strength from linking the current direction with the previous one.

6.4 Subgradient Algorithms with Finite Storage

Conjugate subgradient algorithms presented so far have the drawback that they require storage for subgradients which grows as the iteration count increases. That deficiency can be relatively easily removed from subgradient algorithms but at the cost of losing the conjugacy property of directions generated by these methods. However, to compensate for that, modified algorithms with finite storage guarantee the convergence in a finite number of steps for piecewise linear functions. In this section we present the first such algorithm. For the simplicity of presentation it is stated for convex functions but its extension to the problems described by semismooth functions is possible.

In order to present the algorithm we refer to the following notation introduced in [119]. We define a function $\alpha : \mathscr{R}^n \times \mathscr{R}^n \to \mathscr{R}$ by

$$\alpha(x,y) = f(x) - f(y) - g(y)^T(x-y),$$

where $g(y) \in \partial f(y)$, and introduce

$$\alpha_k^j = \alpha(x_k, y_j)$$

for $j \in J_k = \{1,2,\ldots,k\}$.

Algorithm 6.6. (The subgradient algorithm with finite storage)

Parameter: $\mu \in (0,1)$.

1. Choose $x_1 \in \mathscr{R}^n$ and set $y_1 = x_1 \in \mathscr{R}^n$, $g_1 = g(y_1) \in \partial f(y_1)$, $p_0 = g_1$, $\alpha_1^1 = \varepsilon_0 = \eta_0 = 0$ and $k = 1$.
2. Find the multiplier $\lambda_k \in [0,1]$ which solves the problem

$$\min_{\lambda \in [0,1]} \left[\frac{1}{2} \|(1-\lambda)p_{k-1} + \lambda g_k\|^2 + (1-\lambda)\varepsilon_{k-1} + \lambda \alpha_k^k \right]. \tag{6.41}$$

Set

$$p_k = (1-\lambda_k)\, p_{k-1} + \lambda_k g_k \tag{6.42}$$
$$\eta_k = (1-\mu_k)\, \varepsilon_{k-1} + \lambda_k \alpha_k^k \tag{6.43}$$
$$v_k = -\left(\|p_k\|^2 + \eta_k \right) \tag{6.44}$$
$$d_k = -p_k. \tag{6.45}$$

If $v_k = 0$ then STOP.

3. Set $y_{k+1} = x_k + d_k$ and compute $f(y_{k+1})$, $g_{k+1} \in \partial f(y_{k+1})$. If

$$f(y_{k+1}) \leqslant f(x_k) + \mu v_k$$

set $\alpha_L = 1$, otherwise set $\alpha_L = 0$.

4. Calculate

$$x_{k+1} = x_k + \alpha_L d_k \tag{6.46}$$
$$\alpha_{k+1}^{k+1} = f(x_{k+1}) - f(y_{k+1}) - g_{k+1}^T (x_{k+1} - y_{k+1}) \tag{6.47}$$
$$\varepsilon_k = \eta_k + f(x_{k+1}) - f(x_k) + p_k^T (x_{k+1} - x_k). \tag{6.48}$$

Increase k by one and go to Step 2.

The basic properties of the algorithm can be explained by referring to the lemma proved, for example, in [109].

Lemma 6.10. *Suppose that f is a convex function.*

(i) If $\varepsilon_1 \geqslant 0$ and $g \in \partial f(y)$ for some $y \in \mathscr{R}^n$, then, for any $x \in \mathscr{R}^n$ we have

$$g \in \partial_{\varepsilon_2} f(x) \quad with \quad \varepsilon_2 = \varepsilon_1 + f(x) - f(y) - g^T(x-y) \geqslant 0. \tag{6.49}$$

(ii) If $\varepsilon_i \geqslant 0$ and $g_i \in \partial_{\varepsilon_i} f(x)$ for some $x \in \mathscr{R}$, then for any $\lambda \in [0,1]$ we have

$$g_\lambda = \lambda g_1 + (1-\lambda) g_2 \in \partial_{\varepsilon_\lambda} f(x) \quad with \quad \varepsilon_\lambda = \lambda\varepsilon_1 + (1-\lambda)\varepsilon_2.$$

The lemma leads directly to the following result.

Lemma 6.11. *The variables generated by the algorithm satisfy the following relations*

$$\alpha_k^k \geqslant 0 \quad and \quad g_k \in \partial_{\alpha_k^k} f(x_k) \tag{6.50}$$

$$\eta_k \geqslant 0 \quad and \quad p_k \in \partial_{\eta_k} f(x_k) \tag{6.51}$$

$$\varepsilon_k \geqslant 0 \quad and \quad p_k \in \partial_{\varepsilon_k} f(x_{k+1}) \tag{6.52}$$

for all k.

Proof. (6.50). If $k = 1$ (6.50) obviously holds, furthermore part *(i)* of Lemma 6.10 implies that (6.50) also holds for $k > 1$ if we substitute y_{k+1} for y and x_{k+1} for x and take $\varepsilon_1 = 0$ and $\varepsilon_2 = \alpha_k^k$ in (6.49).

Equation *(6.51)*. In order to prove (6.51) it is sufficient to prove (6.52) and then to use part (ii) of Lemma 6.10. Equation (6.51) is easy to verify for $k = 1$. Suppose that (6.52) is true for $k-1$, then applying Lemma 6.10 with x substituted by x_k, g_1 by g_k, g_2 by p_{k-1}, ε_1 by α_k^k and ε_2 by ε_{k-1}. Then using (6.42)–(6.43) we show (6.51) for k.

Equation *(6.52)*. Part (i) of Lemma 6.10 and (6.47) prove (6.52). □

The convergence of Algorithm 6.6 is established by showing that for some subsequence of $\{x_{k_l}\}$ we have

$$\lim_{l\to\infty} p_{k_l} = 0$$
$$\lim_{l\to\infty} \eta_{k_l} = 0,$$

and if $\{x_{k_l}\}$ has an accumulation point $\bar{x}$ then $0 \in \partial f(\bar{x})$, so $\bar{x}$ is a minimizer of f. The following theorem is proved in [109].

Theorem 6.4. *Suppose that $\{x_k\}$ is generated by* Algorithm 6.6*. Then,*

(i) $\{x_k\}$ is a minimizing sequence, i.e.

$$\lim_{k\to\infty} f(x_k) = \inf_x f(x),$$

(ii) every accumulation point of the sequence $\{x_k\}$ is a minimizer of f.

Algorithm 6.6 even though uses only a limited storage is not efficient since it is based on the aggregate subgradient. If we want to use all subgradients available at the kth iteration our direction finding subproblem is

$$\min_{d,\beta}\left[\frac{1}{2}\|d\|^2+v\right] \tag{6.53}$$

subject to the constraints

$$-\alpha_k^k+g_k^Td\leqslant v \tag{6.54}$$

$$-\alpha_k^j+g_j^Td\leqslant v,\quad j\in J_{k-1}. \tag{6.55}$$

The problem (6.53)–(6.55) can be solved by its dual. Define the function

$$\mathscr{L}\left(d,v,\{\lambda^j\}_{j\in J_k}\right)=\frac{1}{2}\|d\|^2+v+\sum_{j\in J_k}\lambda_j\left[-\alpha_k^j+d^Tg^j-v\right], \tag{6.56}$$

where $\lambda^j\geqslant 0$, $j\in J_k$ are Lagrange's multipliers corresponding to constraints (6.54)–(6.55). Then, problem (6.53)–(6.55) can be stated as

$$\min_{d,v}\max_{\lambda^j\geqslant 0,j\in J_k}\mathscr{L}\left(d,v,\{\lambda^j\}_{j\in J_k}\right). \tag{6.57}$$

Using the minimax theorem the solution of problem (6.53)–(6.55) is reached by solving the problem

$$\max_{\lambda^j\geqslant 0,j\in J_k}\min_{d,v}\mathscr{L}\left(d,v,\{\lambda^j\}_{j\in J_k}\right). \tag{6.58}$$

It is easy to verify that the solution of the inner optimization problem leads to the conditions

$$d=-\sum_{j\in J_k}\lambda^jg_j \tag{6.59}$$

$$1=\sum_{j\in J_k}\lambda^j \tag{6.60}$$

which plugged into the function $\mathscr{L}$ in (6.56) defines the outer problem

$$\min_{\lambda}\left[\frac{1}{2}\left\|\sum_{j\in J_k}\lambda^jg_j\right\|^2+\sum_{j\in J_k}\lambda^j\alpha_k^j\right] \tag{6.61}$$

subject to the constraints

$$\sum_{j\in J_k}\lambda^j=1,\ \ \lambda^j\geqslant 0,\ j\in J_k. \tag{6.62}$$

If $\lambda_k = \{\lambda_k^j\}_{j\in J_k}$ are the solutions to (6.61)–(6.62) then the unique solution to (6.53)–(6.55) can be defined by

$$p_k = \sum_{j\in J_k} \lambda_k^j g_j$$
$$\eta_k = \sum_{j\in J_k} \lambda_k^j \alpha_k^j$$

and eventually by

$$d_k = -p_k$$
$$\beta_k = -\left(\|p_k\|^2 + \eta_k\right)$$

with the complementarity conditions

$$\left(-\alpha_k^j g_j^T d_k - \beta_k\right)\lambda_k^j = 0, \quad j \in J_k.$$

Note that problem (6.41) in Algorithm 6.6 has the same structure as the problem (6.61)–(6.62) with the main difference lying in using one ε_{k-1}–subgradient p_{k-1} instead of α_k^j–subgradients g_j, $j \in J_{k-1}$. Following this path we can show problem (6.41) is equivalent to the problem:

$$\min_{d,v}\left[\frac{1}{2}\|d\|^2 + v\right]$$

subject to the constraints

$$-\alpha_k^k + g_k^T d \leqslant v$$
$$-\varepsilon_{k-1} + p_{k-1}^T d \leqslant v.$$

It is easy to show that p_{k-1} can be regarded as a convex combination of $\{g_j\}_{j\in J_{k-1}}$. Due to the Caratheodory's theorem such a combination can be expressed by a convex combination of at most $n+1$ subgradients from the set $\{g_j\}_{j\in J_{k-1}}$. This implies another version of the direction finding subproblem (6.41), based on the problem (6.55)–(6.62), in which the set J_{k-1} is replaced by the subset

$$\hat{J}_k = \left\{j \in J_{k-1} :\ \lambda_{k-1}^j > 0\right\}:$$

$$\min_{d,v}\left[\frac{1}{2}\|d\|^2 + v\right] \tag{6.63}$$

subject to the constraints

$$-\alpha_k^k + g_k^T d \leqslant v \tag{6.64}$$

$$-\alpha_k^j + g_j^T d \leqslant v, \quad j \in \hat{J}_{k-1}. \tag{6.65}$$

Algorithm 6.6 with the direction finding subproblem (6.63)–(6.65), instead of (6.41), has the property of the finite termination when applied to problems described by piecewise linear functions:

$$f(x) = \max_{i \in I} f_i(x) \tag{6.66}$$

$$f_i(x) = a_i^T x + b_i,\ a_i \in \mathscr{R}^n,\ b_i \in \mathscr{R},\ i \in I, \tag{6.67}$$

where I is a finite set of indices. The result is valid under certain assumption which is satisfied when the Haar condition holds, i.e. for every subset $\hat{I} \subset I(\bar{x})$ the set $\{a_i\}_{i \in \hat{I}}$ has maximal rank. Here, $\bar{x}$ is any minimizer of f,

$$I(x) = \left\{ \tilde{i} \in I;\ \max_{i \in I} f_i(x) = a_{\tilde{i}}^T x + b_{\tilde{i}} \right\}.$$

Theorem 6.5. *Suppose that* Algorithm 6.6 *with the direction finding subproblem (6.63)–(6.65) (instead of (6.41)) is used to minimize the function (6.66)–(6.67) for which the Haar condition is satisfied. Then, it terminates in a finite number of iterations.*

6.5 Notes

The chapter is mainly based on the papers [118], [209] and [133] (see also [94], [134], [160], [12]). For example, Theorem 6.2 is cited and proved, to much extent, after Lemma 2.1 in [118], while Lemma 6.4 refers to Theorem 1 in [209]. Lemma 6.5 is the basis for the first specialized procedure for calculating directions of descent in nondifferentiable optimization – it is presented in [210] together with such algorithm.

Modern codes for nondifferentiable optimization follow the algorithmic approach discussed in Sect. 4. which is more related to the cutting plane method [108] than to a conjugate gradient algorithm. Theoretical and algorithmic aspects of nondifferentiable optimization can be consulted in [110] and [104].

Chapter 7
The Method of Shortest Residuals for Differentiable Problems

7.1 Introduction

In this chapter we consider algorithms for the unconstrained minimization problem:

$$\min_{x\in\mathscr{R}^n} f(x). \tag{7.1}$$

In general, we assume that the function f is continuously differentiable – $f \in \mathscr{C}^1$. The problems (7.1) described by functions $f \in \mathscr{C}^1$ we call *differentiable problems*. To solve this problem we may use conjugate gradient algorithms described in Chap. 2, memoryless quasi-Newton methods discussed in Chap. 3, or limited memory quasi-Newton methods introduced in Chap. 5.

The direction at the kth step of a conjugate gradient algorithm is determined according to the rule:

$$d_k = -g_k + t_k d_{k-1} \tag{7.2}$$

where, e.g.

$$t_k = \frac{g_k^T (g_k - g_{k-1})}{\|g_{k-1}\|^2}, \text{ or } t_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \tag{7.3}$$

We know, from Chap. 6, that we can also apply a conjugate gradient algorithm introduced by Lemaréchal and Wolfe. Their algorithm, when used to solve problem (7.2) with a smooth function, could have the search direction rule

$$d_k = -\mathrm{Nr}\{g_k, -d_{k-1}\}, \tag{7.4}$$

where $\mathrm{Nr}\{a,b\}$, in this particular case, is defined as the point from a line segment spanned by the vectors a and b which has the smallest norm (cf. (6.8)–(6.9)), i.e.

$$\|\mathrm{Nr}\,\{a,b\}\| = \min\{\|\lambda a + (1-\lambda)b\| :\ 0 \leqslant \lambda \leqslant 1\}, \tag{7.5}$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

and $\|\cdot\|$ is the Euclidean norm. Here, while considering a differentiable optimization problem we discuss the simplest version of Lemaréchal and Wolfe algorithms in which the information from previous iterations is conveyed through the direction d_{k-1}. Therefore, we consider not the most efficient version of their algorithm but, on the other hand, our search direction problem is quadratic problem (7.5) we can solve analytically.

The rule (7.4) is discussed in Chap. 6. Here, we give yet another characterization of d_k in terms of g_k and d_{k-1} and Lagrange multipliers associated with the related quadratic problem. Let us consider the problem

$$\min_{(v,d)\in\mathscr{R}^{n+1}} \left[v + \frac{1}{2}\|d\|^2 \right] \tag{7.6}$$

subject to

$$g_k^T d \leqslant v \tag{7.7}$$

$$-d_{k-1}^T d \leqslant v. \tag{7.8}$$

We obtain the solution of this problem by solving its dual

$$\begin{aligned} \max_{0\leqslant\lambda\leqslant 1} \left[-\frac{1}{2}\|\lambda g_k - (1-\lambda)d_{k-1}\|^2 \right] &= -\frac{1}{2}\|\mathrm{Nr}\{g_k, -d_{k-1}\}\|^2 \\ &= -\frac{1}{2}\|\lambda_k g_k - (1-\lambda_k)d_{k-1}\|^2. \end{aligned}$$

Moreover, the optimal value v_k is

$$v_k = -\|d_k\|^2.$$

From (7.7)–(7.8) we can deduce the following properties

$$g_k^T d_k \leqslant -\|d_k\|^2 \tag{7.9}$$

$$-d_{k-1}^T d_k \leqslant -\|d_k\|^2 \tag{7.10}$$

and

$$g_k^T d_k = -\|d_k\|^2 \tag{7.11}$$

$$d_{k-1}^T d_k = \|d_k\|^2 \tag{7.12}$$

if $0 < \lambda_k < 1$ holds.

We know, from Chap. 6, that the method we consider is a conjugate gradient algorithm provided that function f is quadratic and the directional minimization is exact. Wolfe tested his method on problems described by nonconvex functions and found [209] that his version of a conjugate gradient algorithm is twice less efficient than the best implementations of conjugate gradient algorithms. Therefore,

it is natural to ask the questions: If the function f is not quadratic, to which conjugate gradient algorithm, if any, is the Lemaréchal–Wolfe algorithm equivalent? Can we construct other conjugate gradient algorithm based on their scheme?

In order to answer these questions we parameterize the rule (7.4). We introduce the family of algorithms:

$$d_k = -\mathrm{Nr}\{g_k, -\beta_k d_{k-1}\} \tag{7.13}$$

and will work out formulae for β_k for which the process $x_{k+1} = x_k + \alpha_k d_k$, with an exact line search along d_k, becomes a conjugate gradient algorithm.

Notice that if $\beta_k = 1$ then we have the Lemaréchal–Wolfe algorithm. We will show that the Lemaréchal–Wolfe algorithm is equivalent to the Fletcher–Reeves algorithm. Furthermore, we will derive a counterpart of the Polak-Ribière algorithm which is globally convergent under minimal requirements on directional minimization. The results which we present in the next few sections are based on the paper [174].

To complete our introduction we would like to notice that the algorithm with the direction finding subproblem (7.13) has properties similar to (7.9)–(7.12):

$$g_k^T d_k \leqslant -\|d_k\|^2 \tag{7.14}$$

$$-\beta_k d_{k-1}^T d_k \leqslant -\|d_k\|^2 \tag{7.15}$$

and

$$g_k^T d_k = -\|d_k\|^2 \tag{7.16}$$

$$\beta_k d_{k-1}^T d_k = \|d_k\|^2 \tag{7.17}$$

if $0 < \lambda_k < 1$ holds, where λ_k is the Lagrange multiplier of the quadratic problem (7.6)–(7.8) in which the left-hand side of inequality (7.8) is multiplied by β_k.

7.2 General Algorithm

To complete the outline of the general algorithm line search rules have to be specified. Since the proposed algorithm is the adaptation of the Wolfe subgradient algorithm to the differentiable case, these rules are taken from the Wolfe algorithm. According to Lemma 2.4 the following rule is viable in the nonconvex differentiable case: find $\alpha_k > 0$ satisfying

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu\,\alpha_k \|d_k\|^2 \tag{7.18}$$

$$g(x_k + \alpha_k d_k)^T d_k \geqslant -\eta \|d_k\|^2. \tag{7.19}$$

Our general algorithm is stated below.

Algorithm 7.1. (The general method of shortest residuals for nonconvex functions)

Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu, \{\beta_k\}$.

1. Choose $x_1 \in \mathscr{R}^n$, compute:

$$d_1 = -g_1$$

   and set $k = 1$.
2. Find a positive number α_k such that (7.18)–(7.19) are fulfilled. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise calculate:

$$d_{k+1} = -\mathrm{Nr}\left\{g_{k+1}, -\beta_{k+1} d_k\right\}. \tag{7.20}$$

4. Increase k by one and go to Step 2.

The directional minimization is defined by the expressions (7.18)–(7.19). These rules, which lead to inexact minimization, are discussed in Chap. 2 where we also provide an algorithm which finds α_k in a finite number of iterations. Let us observe that these conditions for directional minimization are very similar to the Wolfe conditions. In order to notice that we have to replace $-\|d_k\|^2$ in (7.18) by $g_k^T d_k$. However, in general, we can only guarantee that (7.16) holds.

The step-length conditions (7.18)–(7.19) have been implemented – [173] and work well in practice. Since they are close to the Wolfe conditions it is also straightforward to adapt to rules (7.18)–(7.19) well-tested step-length selection procedures such as that of Moré and Thuente (discussed in Sect. 2.4).

To investigate the convergence of Algorithm 7.1 we begin by providing the lemma which states that the sequence $\{d_k\}$ converges to zero irrespective of a sequence $\{\beta_k\}$. It is as important as Lemma 6.4 in analyzing the convergence of conjugate subgradient algorithms.

Lemma 7.1. *If the direction d_k is determined by (7.20) and the step-size coefficient α_k satisfies (7.18)–(7.19) then:*

$$\lim_{k\to\infty} \|d_k\| = 0, \tag{7.21}$$

or

$$\lim_{k\to\infty} f(x_k) = -\infty. \tag{7.22}$$

Proof. Let us assume that (7.22) is not true. Because $\{f(x_k)\}$ is nonincreasing and bounded from below, it has to be convergent, so that we have

$$f(x_k + \alpha_k d_k) - f(x_k) \to_{k\to\infty} 0. \tag{7.23}$$

Equations (7.23) and (7.18) imply (from the theorem on three sequences) that

$$\mu\alpha_k\|d_k\|^2 \to_{k\to\infty} 0.$$

This is not equivalent to (7.21), thus let us suppose that there exists a set of natural numbers K such that

$$\lim_{k\to\infty,k\in K}\|d_k\| = \|\bar{d}\| \neq 0, \tag{7.24}$$

where, in general, we assume that $\|\bar{d}\|$ can be infinite. Then, from (7.23) and (7.24) we have

$$\lim_{k\to\infty,k\in K}\alpha_k\|d_k\| = 0,$$

so that

$$\lim_{k\to\infty,k\in K}\|x_{k+1} - x_k\| = 0.$$

Using this, (7.19) and (7.14) we get

$$-\eta\|d_k\|^2 \leqslant g_{k+1}^T d_k \leqslant -\frac{1+\eta}{2}\|d_k\|^2$$

for $k \geqslant k_1$ where k_1 is some large number. Obviously this is impossible $\big(\eta \in (0,1)\big)$, therefore (7.21) is true. □

Let us look at the specification of the direction d_k. In general we have the following formula for d_k:

$$d_k = -(1-\lambda_k)g_k + \lambda_k\beta_k d_{k-1} \tag{7.25}$$

where $0 \leqslant \lambda_k \leqslant 1$.

The condition (7.21) is not equivalent to the condition: $\lim_{k\to\infty}\|g_k\| = 0$. This is due to the additional vector $\beta_k d_{k-1}$ in the formula (7.25).

It can happen that (7.21) holds because the vectors d_{k-1} are not appropriately scaled by the β_k. The Wolfe sequence is the example of this situation. If $\beta_k = 1$ then we have shown that $\|d_k\| \leqslant \chi\|d_{k-1}\|$, $\chi \in (0,1)$ irrespective of the magnitude of $\|g_k\|$. Furthermore, for the angle ϕ_k between vectors $-g_k$ and d_{k-1} we can have

$$\lim_{k\in K,k\to\infty}\phi_k = \pi$$

for certain sequence $\{g_k\}_{k\in K}$ and inexact line search (cf. Fig. 7.1).

In each of these situations we shall have (7.21). Thus in order to prove the convergence of Algorithm 7.1 we have to exclude these situations. Later in the chapter we present another convergence analysis of the method of shortest residuals related to Algorithm 7.1. However, convergence properties established then are not as strong

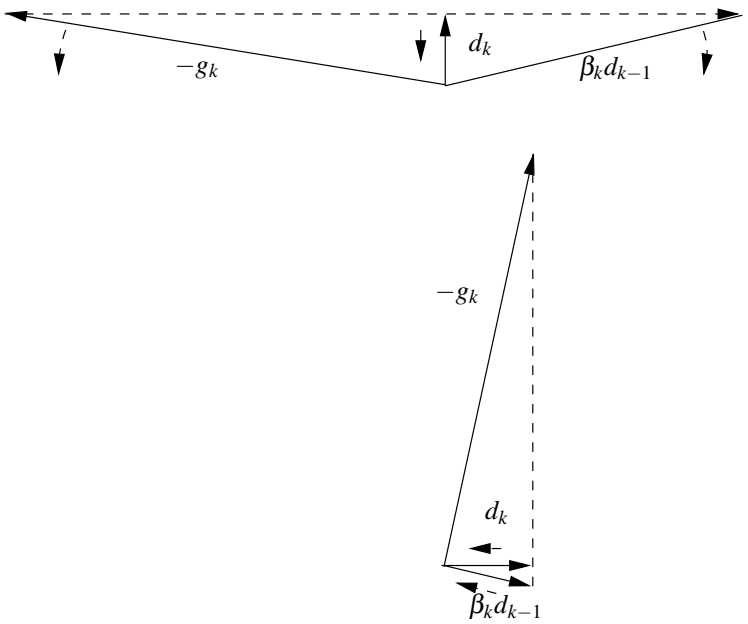

Fig. 7.1 The degenerate cases of d_k

as these presented in this and the next section. In our opinion the convergence analysis we present is unique as far as conjugate gradient algorithms are concerned.

Theorem 7.1. *Assume that* $\{\beta_k\}$ *is such that*

$$\liminf_{k\to\infty}(\beta_k\|d_{k-1}\|) \geqslant \nu_1 \liminf_{k\to\infty}\|g_k\|, \tag{7.26}$$

where ν_1 *is some positive constant. If there exists a number* ν_2 *and an integer* k_1 *such that* $\nu_2 \in (0,1)$*,*

$$g_k^T d_{k-1} \leqslant \nu_2\|g_k\|\|d_{k-1}\|, \tag{7.27}$$

whenever $\lambda_k \in (0,1)$ *and* $k \geqslant k_1$*, then*

$$\lim_{k\to\infty} f(x_k) = -\infty,$$

or every cluster point $\bar{x}$ *of the sequence* $\{x_k\}$ *generated by* Algorithm 7.1 *is such that* $g(\bar{x}) = 0$*.*

Proof. Case (a). Let us suppose that for infinitely many $k \in K_1$, $\lambda_k \in (0,1)$, thus:

$$g_k^T d_k = -\|d_k\|^2 \tag{7.28}$$

$$\beta_k d_{k-1}^T d_k = \|d_k\|^2. \tag{7.29}$$

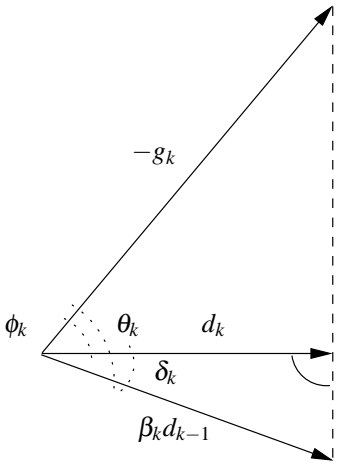

Fig. 7.2 The construction of d_k

Moreover, let us assume that $x_k \to_{k\to\infty, k\in K_1} \bar{x}$,

$$\lim_{k\to\infty, k\in K_1} \|g_k\| = \|g(\bar{x})\| \neq 0. \tag{7.30}$$

Because of this, the equalities (7.28)–(7.29), and since by Lemma 7.1 $\lim_{k\to\infty} \|d_k\| = 0$, we have (cf. Fig. 7.2) for the angle ϕ_k between vectors $-g_k$ and d_k, and for the angle δ_k between vectors $\beta_k d_{k-1}$ and d_k:

$$\begin{aligned} \lim_{k\to\infty, k\in K_1} \cos\phi_k &= \lim_{k\to\infty, k\in K_1} \left\langle -\frac{g_k}{\|g_k\|}, \frac{d_k}{\|d_k\|} \right\rangle \\ &= \lim_{k\to\infty, k\in K_1} \frac{\|d_k\|}{\|g_k\|} = 0 \end{aligned} \tag{7.31}$$

$$\lim_{k\to\infty, k\in K_1} \cos\delta_k = \lim_{k\to\infty, k\in K_1} \frac{\|d_k\|}{\|d_{k-1}\|\beta_k} = 0. \tag{7.32}$$

Since (7.31)–(7.32) are satisfied: $\phi_k \to \pi/2,\ \delta_k \to \pi/2$. Let us consider the angle $\phi_k + \delta_k$. From the usual calculus it follows that

$$\begin{aligned} \lim_{k\to\infty, k\in K_1} \cos(\phi_k + \delta_k) = \lim_{k\to\infty, k\in K_1} \cos\phi_k \cos\delta_k - \\ \lim_{k\to\infty, k\in K_1} \sin\phi_k \sin\delta_k = -1 \end{aligned}$$

(cf. Fig. 7.2), but this implies that

$$\lim_{k\to\infty, k\in K_1} \left\langle \frac{g_k}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle = 1.$$

This contradicts our assumption (7.27) hence we conclude that $g(\bar{x}) = 0$.

Case (b). Now let consider the case $\lambda_k = 1$. If it occurs for infinitely many $k \in K_2$ and

$$x_k \to_{k\to\infty, k\in K_2} \bar{x}, \text{ with } g(\bar{x}) \neq 0$$

we have that there is a ν_4 such that

$$\liminf_{k\to\infty, k\in K_2} \|g_k\| = \nu_4 > 0$$

and by assumptions: $\lambda_k = 1$, (7.26)

$$\liminf_{k\to\infty, k\in K_2} (\beta_k \|d_{k-1}\|) \geqslant \nu_1 \nu_4 > 0.$$

But

$$\lim_{k\to\infty, k\in K_2} \|d_k\| = 0 = \lim_{k\to, k\in K_2} (\beta_k \|d_{k-1}\|) \geqslant \nu_1 \nu_4 > 0$$

and this is impossible.

Case (c). If we have the case $\lambda_k = 0$ for $k \in K_3$ then $-d_k = g_k$ for $k \in K_3$. If

$$x_k \longrightarrow_{k\to\infty, k\in K_3} \bar{x}, \ g(\bar{x}) \neq 0$$

then

$$\lim_{k\to\infty, k\in K_3} \|g_k\| > 0$$

but this is a contradiction to $\lim_{k\to\infty} \|d_k\| = 0$. This completes the proof. □

The condition (7.27) is independent of the choice of the sequence $\{\beta_k\}$ and is related to the directional minimization. Therefore it is not surprising that in some important situations it can be substituted by other, more easily verifiable conditions.

Lemma 7.2. *Let* $\{x_k\}$ *be generated by* Algorithm 7.1, *where* β_k *satisfies (7.26). Then*

$$\lim_{k\to\infty} f(x_k) = -\infty, \text{ or } \lim_{k\to\infty} \|g_k\| = 0$$

if one of the following conditions holds:

(i) there exists $\eta \in (0,1)$ *such that*

$$|g_{k+1}^T d_k| \leqslant \eta \|d_k\|^2,$$

(ii)

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0, \tag{7.33}$$

(iii) function f is locally uniformly convex, i.e. there exists an increasing function d from $\mathscr{R}_+$ into $\mathscr{R}$ such that:

$$d(0) = 0,\ d(t) > 0 \text{ if } t > 0,$$
$$f(\lambda x + (1-\lambda)y) \leqslant \lambda f(x) + (1-\lambda)f(y) - \lambda(1-\lambda)d(\| x - y \|), \tag{7.34}$$

for all $x, y \in \mathscr{R}^n$ and all $\lambda \in [0,1]$. This condition is satisfied in particular if f is strongly convex, in that case $d(t) = \alpha t^2,\ \alpha > 0$.

Proof. First of all we assume that there exists $M < \infty$ such that

$$f(x_k) \geqslant M,\ \forall k, \tag{7.35}$$

otherwise we would have our thesis.

Let us assume that the case *(i)* occurs, and

$$\lim_{k\to\infty, k\in K} \|g_k\| = \alpha > 0 \tag{7.36}$$

for some infinite set K, then in order to prove the lemma we have to show that (7.27) holds provided that $0 < \lambda_k < 1$. This condition can be rewritten as

$$\left\langle \frac{g_k}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle \leqslant \nu$$

for $\nu \in (0,1)$ and all $k \geqslant k_1$. If (7.34) and (7.36) are satisfied then

$$\left\langle \frac{g_k}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle \leqslant \eta \frac{\|d_{k-1}\|}{\|g_k\|} \leqslant \nu \tag{7.37}$$

for some $\nu \in (0,1)$ and all $k \geqslant k_1$ and $k \in K$ since $\|d_k\| \to_{k\to\infty} 0$. Theorem 7.1 leads to the contradiction with (7.36).

For the part *(ii)* of the lemma the proof is carried out along the same lines as for the part *(i)*. Firstly we assume that for some infinite set K we have (7.36). Furthermore, we can write

$$g_k^T d_{k-1} = g_{k-1}^T d_{k-1} + y_k^T d_{k-1}$$

where $y_k = g_k - g_{k-1}$. It follows, from (7.14),

$$\begin{aligned}\left\langle \frac{g_k}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle &= \left\langle \frac{g_{k-1}}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle + \left\langle \frac{y_k}{\|g_k\|}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle \\ &\leqslant -\frac{\|d_{k-1}\|}{\|g_k\|} + \frac{o(\|x_k - x_{k-1}\|)\|d_{k-1}\|}{\|g_k\|\|d_{k-1}\|}\end{aligned}$$

where $o(\|x_{k+1} - x_k\|)$ is such that

$$\lim_{k\to\infty} o(\|x_{k+1} - x_k\|) = 0.$$

It is obvious that there exist $\nu \in (0,1)$ and k_1 such that (7.37) is valid for all $K \ni k \geqslant k_1$.

Now let us assume that the case *(iii)* occurs. Because f is uniformly convex and differentiable, we have (cf. [118])

$$\begin{aligned} &f(y) \geqslant f(x) + g^T(y-x) + d(\|y-x\|) \\ &\text{and } d(\|y-x\|) \to 0 \text{ when } \|y-x\| \to 0. \end{aligned} \tag{7.38}$$

Let us suppose that there exists subsequence $\{g_k\}_{k\in K}$ such that (7.36) is satisfied. In this case, in view of the condition (7.26),

$$g_k^T(x_{k+1} - x_k) = -\alpha_k \|d_k\|^2, \tag{7.39}$$

since $0 < \lambda_k < 1$. Thus, from (7.35), we have

$$f(x_{k+1}) - f(x_k) \to_{k\to\infty} 0$$

and from (7.18)

$$\alpha_k \|d_k\|^2 \to_{k\to\infty} 0.$$

Then it follows, from (7.38), (7.39) and the definition of $d(\cdot)$, that

$$\|x_{k+1} - x_k\| \to_{k\in K} 0.$$

This, as in the proof of part *(ii)*, leads to a contradiction. □

Lemma 7.2 indicates that the assumption (7.27) is necessary because of the rather inexact directional minimization. Here, we must recall that the condition (7.33) plays an important role in conjugate gradient algorithms which are based on the Polak–Ribière rule.

7.3 Versions of the Method of Shortest Residuals

Now we shall give the examples of a sequence $\{\beta_k\}$ which assure that under exact directional minimization the directions $\{d_k\}$ generated by Algorithm 7.1 are conjugate, provided that f is quadratic.

For this purpose we consider the direction finding problem:

$$\min_{0\leqslant\lambda\leqslant 1} \|\lambda\,(-\beta_k d_{k-1}) + (1-\lambda) g_k\|. \tag{7.40}$$

We can solve this problem analytically:

$$\begin{aligned}\|d_k\|^2 &= \min_{0\leqslant\lambda\leqslant 1} \left[\beta_k^2\lambda^2\|d_{k-1}\|^2 - 2\lambda(1-\lambda)\beta_k g_k^T d_{k-1} + \right.\\ &\quad \left.(1-\lambda)^2\|g_k\|^2\right]\\ &= \min_{0\leqslant\lambda\leqslant 1} \left[\lambda^2\left(\beta_k^2\|d_{k-1}\|^2 + 2g_k^T d_{k-1} + \|g_k\|^2\right) + \right.\\ &\quad \left.\lambda\left(-2\beta_k g_k^T d_{k-1} - 2\|g_k\|^2\right) + \|g_k\|^2\right].\end{aligned}$$

Assuming that λ is not restricted we have the solution

$$\lambda_{min} = \frac{\|g_k\|^2 + \beta_k g_k^T d_{k-1}}{\beta_{k-1}^2\|d_{k-1}\|^2 + 2\beta_k g_k^T d_{k-1} + \|g_k\|^2}. \tag{7.41}$$

If, in addition, $g_k^T d_{k-1} = 0$ then

$$\lambda_{min} = \frac{\|g_k\|^2}{\beta_k^2\|d_{k-1}\|^2 + \|g_k\|^2} \tag{7.42}$$

and $\lambda_{min} \in (0,1)$.

In order to derive formulae for β_k which guarantee that $\{d_k\}$ are conjugate suppose that

$$\beta_k = \frac{\gamma_k}{\sqrt{1-\gamma_k^2}}\frac{\|g_k\|}{\|d_{k-1}\|},\ \gamma_k \in (0,1). \tag{7.43}$$

Then, we have

$$\begin{aligned}\lambda_{min} &= \frac{\|g_k\|^2}{\left[\gamma_k^2/(1-\gamma_k^2)\right]\|g_k\|^2 + \|g_k\|^2}\\ &= 1-\gamma_k^2\end{aligned}$$

and, from (7.42),

$$\begin{aligned}\|d_k\|^2 &= \frac{\gamma_k^2}{1-\gamma_k^2}\left(1-\gamma_k^2\right)^2\|g_k\|^2 + \\ &\quad \left(1-1+\gamma_k^2\right)^2\|g_k\|^2\\ &= \gamma_k^2\|g_k\|^2.\end{aligned} \tag{7.44}$$

We notice that γ_k has an important meaning – θ_k, the angle between d_k and g_k, has a cosine:

$$\begin{aligned}\cos\theta_k &= \frac{d_k^T g_k}{\|d_k\|\|g_k\|}\\ &= -\frac{\|d_k\|^2}{\|d_k\|\|g_k\|}\\ &= -\gamma_k. \end{aligned} \tag{7.45}$$

since $\lambda_{min} \in (0,1)$ thus (7.16) applies.

We shall use this to derive a formula for the sequence $\{\beta_k\}$ by forcing directions $\{d_k\}$ to be colinear with the directions generated by standard conjugate gradient algorithms.

Suppose that $\{\underline{d_k}\}$ are obtained by the iterative process

$$\begin{aligned}\underline{d_1} &= -g_1\\ \underline{d_k} &= t_k d_{k-1} - g_k\\ 0 &= g_{k+1}^T \underline{d_k}\\ x_{k+1} &= x_k + \alpha_k \underline{d_k}\end{aligned}$$

with

$$t_k = \frac{\|g_k\|^2}{\|g_{k-1}\|}, \tag{7.46}$$

or

$$t_k = \frac{(g_k - g_{k-1})^T g_k}{\|g_{k-1}\|^2}. \tag{7.47}$$

We select β_k in such a way that cosine of angles between g_k and $\underline{d_k}$ and between g_k and d_k are the same. This can be achieved if we set

$$\gamma_k = \frac{\|g_k\|}{\sqrt{t_k^2\|\underline{d_{k-1}}\|^2 + \|g_k\|^2}} \tag{7.48}$$

provided that

$$\underline{d_{k-1}} = \frac{1}{\gamma_{k-1}^2} d_{k-1} \tag{7.49}$$

holds.

Indeed, we have

$$\begin{aligned}\|\underline{d_k}\|^2 &= t_k^2\|\underline{d_{k-1}}\|^2 + \|g_k\|^2\\ \cos\underline{\theta_k} &= -\gamma_k\end{aligned}$$

where, $\underline{\vartheta}_k$ is the angle between vectors g_k and $\underline{d}_k$, which together with (7.44) imply that

$$\gamma_k^2 = \frac{\|g_k\|^2}{\|\underline{d}_k\|^2} = \frac{\|d_k\|^2}{\gamma_k^2 \|\underline{d}_k\|^2}$$

and eventually

$$\gamma_k^2 \underline{d}_k = d_k.$$

The direct consequence of the above relations is the following theorem.

Theorem 7.2. *If directional minimization is exact, $\{\beta_k\}$ is defined by (7.43) in which γ_k is determined by (7.48)–(7.49) and one of the expressions (7.46) or (7.47), then*

(i) The directions $\{d_k\}$ are colinear with those generated by a conjugate gradient algorithm,

(ii) If t_k is given by (7.47) and $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$ we also have:

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{7.50}$$

Proof. We have

$$\begin{aligned} d_1 &= -g_1 \\ d_{k+1} &= \gamma_{k+1}^2 \left(t_{k+1}\underline{d}_k - g_{k+1}\right) \\ \underline{d}_k &= \frac{1}{\gamma_k^2} d_k \\ 0 &= g_k^T \underline{d}_{k-1}, \end{aligned}$$

$k = 1, 2, \ldots$ and $\gamma_1 = 1$.

Now consider the directions generated by the conjugate gradient method (cf. (7.2)–(7.3)):

$$\begin{aligned} \tilde{d}_1 &= -g_1 \\ \tilde{d}_{k+1} &= t_{k+1}\tilde{d}_k - g_{k+1} \\ 0 &= g_k^T \tilde{d}_{k-1} \end{aligned}$$

for $k = 1, 2, \ldots,$. It is evident that the directions $\{\underline{d}_k\}$ fulfill these conditions and since $\gamma_k^2 \neq 0$, $d_k = \gamma_k^2 \underline{d}_k$. We have thus proved assertion *(i)*. Regarding *(ii)* we recall (7.45). If

$$\liminf_{k\to\infty} \|g_k\| > 0$$

then it can be shown that $\liminf_{k\to\infty} \gamma_k > 0$. But that together with the condition $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$ lead to a contradiction (cf. (7.44) and Lemma 7.1), hence (7.50) holds. □

From (7.48)–(7.49), (7.43) it is evident that we can deliver many formulae for β_k, these described by (7.43) and (7.48)–(7.49) are only examples. Another formula is obtained by setting

$$t_k = \frac{(g_k - g_{k-1})^T g_k}{(g_k - g_{k-1})^T d_{k-1}}$$

in (7.48). This is the Hestenes–Stiefel formula which is equivalent to the Polak–Ribière formula only when directional minimization is exact (cf. Chap. 3).

Since

$$1 - \gamma_k^2 = \frac{t_k^2 \|\underline{d_{k-1}}\|^2}{t_k^2 \|\underline{d_{k-1}}\|^2 + \|g_k\|^2},$$

if we define β_k by (7.43) with γ_k given by (7.48) we obtain

$$\beta_k = \frac{\|g_k\|^2 \gamma_{k-1}^2}{|t_k| \|d_{k-1}\|^2}.$$

If we take t_k defined by (7.46) and recall (7.44) (which is valid when directional minimization is exact) we obtain the Lemaréchal–Wolfe formula

$$\beta_k = 1. \tag{7.51}$$

Similarly we can obtain the formula (cf. (7.47))

$$\beta_k = \frac{\|g_k)\|^2}{|(g_k - g_{k-1})^T g_k|}. \tag{7.52}$$

As (7.51) can be regarded as the Fletcher–Reeves formula, (7.52) can be treated as the Polak–Ribière formula.

7.4 Polak–Ribière Version of the Method of Shortest Residuals

In this section we examine Algorithm 7.1 with the sequence $\{\beta_k\}$ defined by (7.52).

We can prove the following theorem.

Theorem 7.3. *Suppose that there exists $L > 0$ such that*

$$\|g(x) - g(y)\| \leqslant L\|x - y\|, \ \ \forall x, y \in \mathscr{R}^n. \tag{7.53}$$

Then Algorithm 7.1 *gives*

$$\lim_{k \to \infty} f(x_k) = -\infty,$$

or

$$\lim_{k\to\infty} \|g_k\| = 0 \tag{7.54}$$

provided that β_k *is given by (7.52) and one of the following conditions is satisfied*:

(i) There exists $M < \infty$ *such that*

$$\alpha_k \leqslant M, \ \forall\, k;$$

(ii) α_k *is determined by (7.18) and the condition*

$$|g_{k+1}^T d_k| \leqslant \eta \|d_k\|^2.$$

Proof. We have for the formula (7.52):

$$\begin{aligned} \beta_k \|d_{k-1}\| &= \frac{\|g_k\|^2 \|d_{k-1}\|}{|(g_k - g_{k-1})^T g_k|} \\ &\geqslant \frac{\|g_k\|^2 \|d_{k-1}\|}{L\alpha_{k-1}\|g_k\|\|d_{k-1}\|} \\ &\geqslant \frac{\|g_k\|}{LM}. \end{aligned} \tag{7.55}$$

If $f(x_k) \geqslant \tilde{L} > -\infty$ then, because

$$\|x_k - x_{k-1}\| \leqslant M\|d_{k-1}\|$$

and $\|d_k\| \to_{k\to\infty} 0$ (Lemma 7.1), we know from Lemma 7.2 part (i) and (ii) that only violation of the condition (7.26) can destroy our convergence result (7.54). But (7.55) shows that the condition (7.26) is also fulfilled. □

This convergence result is not as restrictive as other results for the conjugate gradient methods (especially for the Polak–Ribière methods) presented in Chap. 2.

The condition (ii) in Theorem 7.3 is very useful in proving convergence of Algorithm 7.1 (with the rule (7.52)) for problems with strictly convex functions.

Theorem 7.4. *If the function* f *is twice continuously differentiable and there exists* $0 < m < M < +\infty$ *such that:*

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2$$

for all $x, z \in \mathscr{R}^n$*, then the sequence* $\{x_k\}$ *generated by* Algorithm 7.1 *with* β_k *calculated in accordance with (7.52) converges to the minimizer of* f.

Proof. Assume first that the direction of descent satisfies the condition:

$$g_k^T d_k = -\|d_k\|^2, \tag{7.56}$$

then we will show that there exists $M_\alpha < +\infty$ such that $\alpha_k \leqslant M_\alpha$ for all k. Indeed, from the Taylor's expansion theorem we have

$$f(x_k + \alpha d_k) = f(x_k) + \alpha g_k^T d_k + \frac{1}{2}(\alpha)^2 d_k^T \nabla^2 f(\xi_k) d_k$$

for some ξ_k. Therefore, if

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu\, \alpha_k \|d_k\|^2$$

then $\alpha_k \leqslant M_\alpha = 2(1-\mu)/m$. Thus, in the first case, there exists $M_\alpha < +\infty$ such that $\alpha_k \leqslant M_\alpha$ for all k for which (7.56) is fulfilled, thus from Theorem 7.3 (since condition (7.53) is obviously satisfied) we have the thesis.

Assume now that condition (7.56) is fulfilled only finitely many times. Thus $d_k = \beta_k d_{k-1}$ for $k \geqslant k_1$. Because $\{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\}$ is a closed bounded set (Lemma I2.8 in [171]), it is easy to show that

$$\sum_{k_1}^{\infty} \|x_{k+1} - x_k\| < \infty,$$

thus

$$\sum_{k=k_1}^{\infty} \alpha_k \beta_k \|d_{k-1}\| < \infty,$$

and

$$\lim_{k\to\infty} \alpha_k \beta_k \|d_{k-1}\| = 0.$$

But

$$\begin{aligned} \alpha_k \beta_k \|d_{k-1}\| &= \frac{\alpha_k \|g_k\|^2}{|\,(g_k - g_{k-1})^T\, g_k|} \|d_{k-1}\| \\ &\geqslant \frac{\alpha_k}{\alpha_{k-1}} \frac{\|g_k\|}{L}. \end{aligned}$$

Therefore, either

$$\lim_{k\to\infty} \|g_k\| = 0,$$

or

$$\lim_{k\to\infty} \frac{\alpha_k}{\alpha_{k-1}} = 0$$

and from the D'Alambert's theorem it follows that

$$\sum_{k=k_1}^{\infty} \alpha_k < \infty.$$

Therefore there exists $M_\alpha < \infty$ such that $\alpha_k \leqslant M_\alpha$ for all $k \geqslant k_1$, and on the basis of Theorem 7.3 we have proved our thesis. □

The algorithm has been presented under the silent assumption that at every iteration we have

$$g_{k+1}^T d_k \neq \|g_{k+1}\| \|d_k\|. \tag{7.57}$$

If (7.57) happens then $d_{k+1} = 0$ and any $\alpha_{k+2} > 0$ will satisfy the modified Wolfe conditions at iteration $k+2$ and we will have $d_{k+2} = -g_{k+2}$.

We end this section by discussing the use of the Armijo step-size rule in Algorithm 7.1, i.e. when the coefficient α_k is determined according to:

$$\begin{aligned} \alpha_k = \arg\max\{\tau^i :\ & f(x_k + \tau^i d_k) - f(x_k) \leqslant -\mu \tau^i \|d_k\|^2, \\ & \tau \in (0,1),\ i = 0,1,\ldots\}. \end{aligned} \tag{7.58}$$

It is easy to show that Algorithm 7.1 with the directional minimization (7.58) instead of (7.18)–(7.19) has all the properties described in this section, in particular it is globally convergent if the sequence $\{\beta_k\}$ is defined by (7.52).

We can prove this fact as follows. If we apply (7.58) as a step-size rule, then we can show that one of the relations (7.21)–(7.22) will hold. From theorem 2.3 we know that $\lim_{k\to\infty} \|d_k\|^2 = 0$. Moreover, because (7.33) is fulfilled, and $\{\beta_k\}$ satisfies condition (7.26), the global convergence follows from Lemma 7.2.

The new algorithm does not require exact directional minimization and is globally convergent without any additional assumptions except (7.53). It seems to be the first such algorithm. Yet we do not recommend the use of (7.58) in Algorithm 7.1, because the conjugate gradient algorithm with this directional minimization can exhibit slow convergence.

7.5 The Method of Shortest Residuals by Dai and Yuan

The method of shortest residuals developed in [174] attracted attention of other researches due to its weak conditions guaranteeing global convergence. Dai and Yuan [43] (see also [177]) propose a version of the method of shortest residuals which applies the Wolfe conditions in the line search instead of their modified version originated in nondifferentiable optimization. But it is not the only contribution of Dai and Yuan. They also show how to prove global convergence of the Fletcher–Reeves version of the method of shortest residuals – in [174] the convergence of this version is not analyzed.

In order to use the standard Wolfe conditions Dai and Yuan modify the direction finding subproblem. d_k is obtained by solving the following univariate quadratic problem:

$$\begin{aligned} \|d_k\|^2 &= \min_\lambda \|(1-\lambda) g_k - \lambda \beta_k d_{k-1}\|^2 \\ &= \|(1-\lambda_k) g_k - \lambda_k \beta_k d_{k-1}\|^2. \end{aligned} \tag{7.59}$$

Observe that λ in problem (7.59) is not restricted. However, whenever $g_k^T d_{k-1} = 0$, then the constraints in problem (7.40) ($0 \leqslant \lambda \leqslant 1$) are not active so both problems are equivalent. This implies that both search directions rules are also the same when applied to quadratics.

We can provide the solution to (7.59) by referring to (7.41):

$$\lambda_k = \frac{\|g_k\|^2 + \beta_k g_k^T d_{k-1}}{\|g_k + \beta_k d_{k-1}\|^2}. \tag{7.60}$$

Observe that we always have

$$g_k^T d_k = -\|d_k\|^2 \tag{7.61}$$

since both g_k and $\beta_k d_{k-1}$ contribute to d_k. Furthermore, the optimal value of (7.59) is given by

$$\|d_k\|^2 = \frac{\beta_k^2 \left(\|g_k\|^2 \|d_{k-1}\|^2 - \left(g_k^T d_{k-1}\right)^2 \right)}{\|g_k + \beta_k g_k^T d_{k-1}\|^2} \tag{7.62}$$

which can be established by recalling the formula for the optimal value of univariate quadratic (7.41):

$$\begin{aligned}
\|d_k\|^2 &= -2\left(\beta_k g_k^T d_{k-1} + \|g_k\|^2\right) \frac{\lambda_k}{2} + \|g_k\|^2 \\
&= \frac{-\left(\|g_k\|^2 + \beta_k g_k^T d_{k-1}\right)^2 + \|g_k\|^2 \|g_k + \beta_k d_{k-1}\|^2}{\|g_k + \beta_k d_{k-1}\|^2} \\
&= \frac{-\beta_k^2 \left(g_k^T d_{k-1}\right)^2 - \|g_k\|^2 g_k^T d_{k-1} - \|g_k\|^4}{\|g_k + \beta_k d_{k-1}\|^2} + \\
&\quad \frac{\beta_k \|g_k\|^2 \|d_{k-1}\|^2 + 2\|g_k\|^2 \beta_k g_k^T d_{k-1} + \|g_k\|^4}{\|g_k + \beta_k d_{k-1}\|^2} \\
&= \frac{\beta_k^2 \left(\|g_k\|^2 \|d_{k-1}\|^2 - \left(g_k^T d_{k-1}\right)^2 \right)}{\|g_k + \beta_k g_k^T d_{k-1}\|^2}
\end{aligned}$$

Notice that if g_k, d_{k-1} are collinear according to (7.62) we can have $d_k = 0$. Therefore, Dai and Yuan (cf. the previous section where the problem is also discussed) propose to define d_k in this situation in the following way:

$$d_k = \begin{cases} -g_k & \text{if } \xi_k < -1 \\ \xi_k g_k & \text{if } -1 \leqslant \xi_k < 0 \\ 0 & \text{if } \xi_k \geqslant 0 \end{cases} \tag{7.63}$$

where ξ_k is such that

$$\beta_k d_{k-1} = \xi_k g_k. \tag{7.64}$$

Dai and Yuan rely on the sequences $\{\beta_k\}$ introduced in [174]:

$$\beta_k = 1 \tag{7.65}$$

$$\beta_k = \frac{\|g_k\|^2}{|g_k^T (g_k - g_{k-1})|} \tag{7.66}$$

with the modification of rule (7.66) which is required for their direction rule. We take

$$d_k = -g_k \text{ if } g_k = g_{k-1}. \tag{7.67}$$

Notice that modification (7.67) is not needed when d_k is calculated through the operator Nr since in this case we have $d_k = -\mathrm{Nr}\{g_k, -\beta_k d_{k-1}\} = -g_k$ due to the fact that we can assume $\beta_k = +\infty$.

Algorithm 7.2. (The Dai–Yuan method of shortest residuals)

Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu, \{\beta_k\}$.

1. Choose $x_1 \in \mathscr{R}^n$, compute:
$$d_1 = -g_1.$$
and set $k = 1$.
2. Find a positive number α_k such that:
$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k \tag{7.68}$$
$$g(x_k + \alpha_k d_k)^T d_k \geqslant \eta g_k^T d_k. \tag{7.69}$$
Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP, otherwise calculate d_{k+1} according to (7.59) and (7.63)–(7.64).
4. Increase k by one and go to Step 2.

Algorithm 7.2 belongs to a class of general algorithms for nonlinear problems discussed in Chap. 2 and since (7.61) holds we can prove the following theorem.

Theorem 7.5. *Suppose that*

(i) f is bounded below in $\mathscr{R}^n$
(ii) that f is continuously differentiable in an open set $\mathscr{N}$ containing the level set $\mathscr{M} = \{x \in \mathscr{R}^n :\ f(x) \leqslant f(x_1)\}$,
(iii) $g(x)$ is Lipschitz continuous – there exists $L > 0$ such that

$$\|g(y) - g(x)\| \leqslant L\|y - x\| \tag{7.70}$$

for all $x,\ y \in \mathscr{N}$.

Then

$$\sum_{k\geqslant 0} \cos^2 \theta_k \|g_k\|^2 < \infty$$

where θ_k is the angle between vectors g_k and d_k.

Proof. The theorem can be proved in the same way as Theorem 2.1 if we notice that d_k is always a direction of descent due to the fact that $d_k \neq 0$. □

Theorem 7.5 implies that for Algorithm 7.2 we have

$$\sum_{k=1}^{\infty} \|d_k\|^2 < +\infty \tag{7.71}$$

thus also

$$\lim_{k\to\infty} \|d_k\| = 0.$$

We start detailed convergence analysis of Algorithm 7.2 from its Fletcher–Reeves version.

Theorem 7.6. *Suppose that the assumptions of* Theorem 7.5 *are satisfied and $\beta_k = 1$ for all k. Then*

$$\liminf_{k\to\infty} \|g_k\| = 0 \tag{7.72}$$

provided that one of the conditions holds

(i) $g_k^T d_{k-1} = 0$ *for all k,*
(ii) α_k *satisfy the strong Wolfe conditions, i.e.*

$$f(x_{k+1}) - f(x_k) \leqslant \mu g_k^T d_k \tag{7.73}$$
$$\left|g_{k+1}^T d_k\right| \leqslant \eta \left|g_k^T d_k\right| \tag{7.74}$$

with $0 < \mu < \eta < 1$,
(iii) α_k *satisfy the Wolfe conditions (7.68)–(7.69) and there exists $M < +\infty$ such that*

$$\alpha_k \leqslant M \tag{7.75}$$

for all k.

Proof. The proof is by the contradiction. Suppose that (7.72) is not true, i.e. there exists $\varepsilon > 0$ such that

$$\|g_k\| \geqslant \varepsilon \tag{7.76}$$

for all k.

Formula (7.62) can be rewritten as

$$\|d_k\|^2 = \|d_{k-1}\|^2 \frac{\|g_k\|^2 - \left(g_k^T d_{k-1}\right)^2 / \|d_{k-1}\|^2}{\|g_k\|^2 + g_k^T d_{k-1} + \|d_{k-1}\|^2}$$

noting that $\beta_k = 1$. This leads to the following equivalent relation

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_{k-1}\|^2}(1 + \tau_k)$$

with

$$\tau_k = \frac{\|d_{k-1}\|^2 + 2g_k^T d_{k-1} + \left(g_k^T d_{k-1}\right)^2 / \|d_{k-1}\|^2}{\|g_k\|^2 - \left(g_k^T d_{k-1}\right)^2 / \|d_{k-1}\|^2}. \tag{7.77}$$

Applying (7.77) recursively we obtain

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_1\|^2} \prod_{l=1}^{k} (1 + \tau_l). \tag{7.78}$$

$\|d_k\|^2 \to_{k\to\infty} 0$ thus the product in (7.78) must be divergent and this leads to the condition

$$\sum_{k=1}^{\infty} \tau_k = +\infty. \tag{7.79}$$

Next we will show that (7.76) contradicts with (7.79). If we can prove that under part *(iii)* of the assumption we have

$$\left|g_k^T d_{k-1}\right| \leqslant c\|d_{k-1}\|^2 \tag{7.80}$$

then all points *(i)–(iii)* can be treated in the same way since exact line searches guarantee (7.80) as well as the curvature condition (7.74) where it is sufficient to take $c = \eta$ (cf. relation (7.61)).

Assume that (7.75) holds. Then, from the Hölder inequality, (7.70) and (7.61) we have

$$\begin{aligned}
\left|g_k^T d_{k-1}\right| &\leqslant \left\| \left(g_k - g_{k-1}\right)^T d_{k-1} + g_{k-1}^T d_{k-1} \right\| \\
&\leqslant \|g_k - g_{k-1}\| \|d_{k-1}\| + \left|g_{k-1}^T d_{k-1}\right| \\
&\leqslant \alpha_k L \|d_{k-1}\|^2 + \|d_{k-1}\|^2 \\
&\leqslant (1 + ML)\|d_{k-1}\|^2
\end{aligned}$$

so (7.80) is satisfied with $c = 1 + LM$.

Equations (7.71), (7.77) and (7.80) imply that for $k \geqslant k_1$

$$\tau_k \leqslant \frac{1 + 2c + c^2}{\varepsilon - \varepsilon_1} \|d_{k-1}\|^2 = c_1 \|d_{k-1}\|^2$$

where $\varepsilon_1 < \varepsilon$ is such that

$$c^2\|d_{k-1}\|^2 \leqslant \varepsilon_1 \tag{7.81}$$

for $k \geqslant k_1$.

Equation (7.81) contradicts with (7.79) because

$$\sum_{k=k_1}^{\infty} \tau_k \leqslant c_1 \sum_{k=k_1}^{\infty} \|d_{k-1}\|^2 < +\infty$$

thus (7.76) is not valid. □

Theorem 7.7. *If the function f is twice continuously differentiable in the neighborhood of $\mathcal{N}$ (cf. the assumption (ii) of* Theorem 7.5*) and there exists $0 < m < M < +\infty$ such that:*

$$m\|z\|^2 \leqslant z^T\nabla^2 f(x)z \leqslant M\|z\|^2,$$

for all $x \in \mathcal{N}$, $z \in \mathcal{R}^n$, then the sequence $\{x_k\}_1^\infty$ generated by Algorithm 7.2 *with $\beta_k = 1$ converges to the minimizer of f.*

Proof. The proof is the same as the first part of the proof of Theorem 7.4 if we take into account that we always have $g_k^T d_k = -\|d_k\|^2$. □

The analysis of the Polak–Ribière version of Algorithm 7.2 can be carried out along the lines of the proof of Theorem 7.1 if we notice that for any k we have (7.61). Therefore, we can prove

Theorem 7.8. *Assume that $\{\beta_k\}$ is such that*

$$\liminf_{k\to\infty}(\beta_k\|d_{k-1}\|) \geqslant \nu_1 \liminf_{k\to\infty}\|g_k\| \tag{7.82}$$

where ν_1 is some positive constant. If there exists a number ν_2 and an integer k_1 such that $\nu_2 \in (0,1)$,

$$g_k^T d_{k-1} \leqslant \nu_2\|g_k\|\|d_{k-1}\|, \tag{7.83}$$

whenever $\lambda_k \in (0,1)$ and $k \geqslant k_1$, then

$$\lim_{k\to\infty} f(x_k) = -\infty,$$

or every cluster point $\bar{x}$ of the sequence $\{x_k\}$ generated by Algorithm 7.2 *is such that $g(\bar{x}) = 0$.*

Proof. First we refer to Lemma 7.1 which can be established for Algorithm 7.2 due to relation (7.61).

As in the proof of Theorem 7.1 we have:

$$g_k^T d_k = -\|d_k\|^2 \tag{7.84}$$

$$\beta_k d_{k-1}^T d_k = \|d_k\|^2. \tag{7.85}$$

Moreover let us assume that $x_k \rightarrow_{k\rightarrow\infty,k\in K_1} \bar{x},\ g(\bar{x}) \neq 0$. From this it follows that

$$\lim_{k\rightarrow\infty,k\in K_1} \|g_k\| \neq 0.$$

Because of this, the equalities (7.84–(7.85), and since by Lemma 7.1 $\lim_{k\rightarrow\infty} \|d_k\| = 0$, we have for the angle ϕ_k between vectors $-g_k$ and d_k, and for the angle δ_k between vectors $\beta_k d_{k-1}$ and d_k:

$$\begin{aligned}\lim_{k\rightarrow\infty,k\in K_1} \cos\phi_k &= \lim_{k\rightarrow\infty,k\in K_1} \left\langle -\frac{g_k}{\|g_k\|}, \frac{d_k}{\|d_k\|} \right\rangle \\ &= \lim_{k\rightarrow\infty,k\in K_1} \frac{\|d_k\|}{\|g_k\|} = 0 \qquad (7.86)\end{aligned}$$

$$\lim_{k\rightarrow\infty,k\in K_1} \cos\delta_k = \lim_{k\rightarrow\infty,k\in K_1} \frac{\|d_k\|}{\|d_{k-1}\|\beta_k} = 0. \qquad (7.87)$$

Since (7.86)–(7.87) are satisfied: $\phi_k \rightarrow \pi/2,\ \delta_k \rightarrow \pi/2$. In contrast to the proof of Theorem 7.1 we have to consider three cases as far as the calculation of the cosine of the angle θ_k between vectors $\beta_k d_{k-1}$ and g_k is concerned. We have the possibilities (cf. Fig. 7.3):

$$\theta_k = \delta_k + \phi_k \qquad (7.88)$$

$$\theta_k = \delta_k - \phi_k \qquad (7.89)$$

$$\theta_k = \phi_k - \delta_k. \qquad (7.90)$$

The case (7.88) is analyzed in as in the proof of Theorem 7.1:

$$\begin{aligned}\lim_{k\rightarrow\infty,k\in K_1} \cos(\phi_k + \delta_k) = &\lim_{k\rightarrow\infty,k\in K_1} \cos\phi_k \cos\delta_k - \\ &\lim_{k\rightarrow\infty,k\in K_1} \sin\phi_k \sin\delta_k = -1.\end{aligned}$$

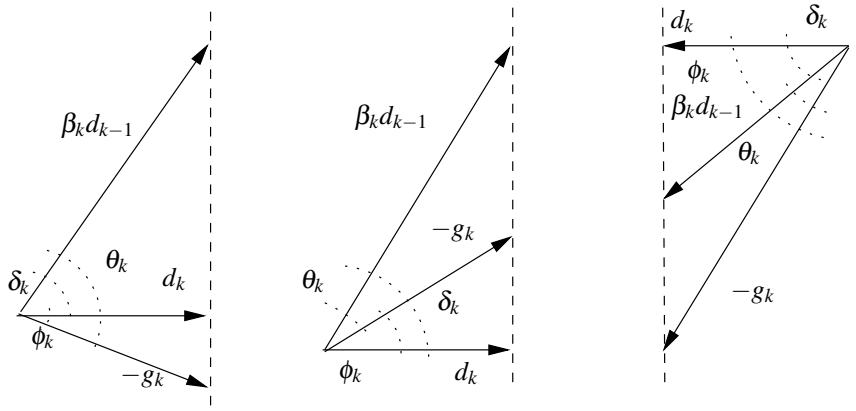

Fig. 7.3 Calculation of d_k

For case (7.89) we have

$$\lim_{k\to\infty,k\in K_1}\cos(\delta_k-\phi_k)=\lim_{k\to\infty,k\in K_1}\cos\phi_k\cos(-\delta_k)-\lim_{k\to\infty,k\in K_1}\sin\phi_k\sin(-\delta_k)=1$$

and analogously for (7.90). Eventually we come to

$$\lim_{k\to\infty,k\in K_1}\left\langle\frac{g_k}{\|g_k\|},\frac{d_{k-1}}{\|d_{k-1}\|}\right\rangle=\pm 1.$$

This contradicts our assumption (7.83) hence we conclude that $g(\bar{x})=0$. □

Condition (7.83) does not have to be checked if the assumptions of Lemma 7.2 hold.

Lemma 7.3. *Let* $\{x_k\}$ *be generated by* Algorithm 7.2*, where* β_k *satisfies (7.82). Then*

$$\lim_{k\to\infty}f(x_k)=-\infty,\ \text{or}\ \lim_{k\to\infty}\|g_k\|=0$$

if one of the conditions (ii)–(iii) stated in Lemma 7.2*, or*

(i) there exists $\eta\in(0,1)$ *such that*

$$\left|g_{k+1}^T d_k\right|\leqslant\eta\left|g_k^T d_k\right|, \tag{7.91}$$

holds

Combining Theorem 7.6 and Lemma 7.2 leads to the analog of Theorem 7.6.

Theorem 7.9. *Suppose that there exists* $L>0$ *such that*

$$\|g(x)-g(y)\|\leqslant L\|x-y\|,\ \ \forall x,y\in\mathscr{R}^n.$$

Then Algorithm 7.2 *gives*

$$\lim_{k\to\infty}f(x_k)=-\infty,$$

or

$$\lim_{k\to\infty}\|g_k\|=0$$

provided that β_k *is given by (7.66) and one of the following conditions is satisfied:*

(i) there exists $M<\infty$ *such that*

$$\alpha_k\leqslant M,\ \forall\, k;$$

(ii) α_k *is determined by the strong Wolfe conditions.*

Finally, we provide the global convergence of Algorithm 7.2 when applied to convex functions.

Theorem 7.10. *Suppose that assumptions of* Theorem 7.4 *are fulfilled then the sequence* $\{x_k\}$ *generated by* Algorithm 7.2 *with* β_k *calculated in accordance with (7.66) converges to the minimizer of* f.

7.6 Global Convergence of the Lemaréchal–Wolfe Algorithm Without Restarts

Now we are ready to refine convergence analysis of the Lemaréchal–Wolfe algorithm. In the previous chapter we establish its convergence when restarts are taken every finite number of iterations. That convergence analysis borrows some ideas from nondifferentiable optimization along the lines of Wolfe and Miffiin reasoning. Having new approach of Dai and Yuan to the method of shortest residuals we can show that the Lemaréchal–Wolfe algorithm is convergent also in the absence of restarts.

Theorem 7.11. *Suppose that the assumptions of* Theorem 7.5 *are satisfied. Consider* Algorithm 7.1 *with* $\beta_k = 1$. *Then we have*

$$\liminf_{k\to\infty} \|g_k\| = 0,$$

if the line search satisfies one of the conditions (i)–(iii) stated in Theorem 7.6 *with the condition (ii) replaced by:*

(ii) there exists $c_1 > 0$ *such that*

$$|g_k^T d_{k-1}| \leqslant c_1 \|d_{k-1}\|^2 \text{ for all } k.$$

Proof. Following Theorem 2.2, since for any k d_k is a direction of descent, we can show that

$$\lim_{k\to\infty} \|d_k\|^2 = 0. \tag{7.92}$$

Assume that

$$\liminf_{k\to\infty} \|g_k\| \neq 0.$$

Then there exists k_1 such that for all $k \geqslant k_1$

$$\|g_k\| \geqslant \gamma > 0.$$

Moreover, we have

$$g_k^T d_k \leqslant -\|d_k\|^2$$

and

$$g_k^T d_k = -\|d_k\|^2 \text{ if } 0 < \lambda_k < 1 \tag{7.93}$$

$$d_k = d_{k-1} \text{ if } \lambda_k = 0 \tag{7.94}$$

$$d_k = -g_k \text{ if } \lambda_k = 1. \tag{7.95}$$

In case (7.93) we also have

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_{k-1}\|^2}(1+\tau_k) \tag{7.96}$$

where

$$\tau_k = \frac{\|d_{k-1}\|^2 + 2g_k^T d_{k-1} + \left(g_k^T d_{k-1}\right)^2/\|d_{k-1}\|^2}{\|g_k\|^2 - \left(g_k^T d_{k-1}\right)^2/\|d_{k-1}\|^2} \tag{7.97}$$

(cf. Sect. 5).

When $\lambda_k = 1$ we can write

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_{k-1}\|^2}(1+\tau_k) \tag{7.98}$$

with

$$\tau_k = \frac{\|d_{k-1}\|^2 - \|g_k\|^2}{\|g_k\|^2}. \tag{7.99}$$

This case, according to (7.92), can happen only a finite number of times. Therefore, we can assume that there exists k_2 such that for all $k \geqslant k_2$ $\lambda_k \neq 1$.

When $\lambda_k = 0$ we can write

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_{k-1}\|^2}(1+\tau_k) \tag{7.100}$$

where $\tau_k = 0$.

From (7.96)–(7.100) we have

$$\frac{1}{\|d_k\|^2} = \frac{1}{\|d_{k_2-1}\|^2} \prod_{l=k_2+1}^{k} (1+\tau_k)$$

where $\tau_k \geqslant 0$.

$\|d_k\|^2 \to_{k\to\infty} 0$ gives

$$\prod_{l=k_2+1}^{\infty} (1+\tau_l) = \infty$$

which implies that

$$\sum_{l=k_2+1}^{\infty} \tau_l = \infty \tag{7.101}$$

(i). If $g_k^T d_{k-1} = 0$ then for $k \geqslant k_2$ either $\tau_k = 0$, or

$$\tau_k = \frac{\|d_{k-1}\|^2}{\|g_k\|^2}.$$

We then have

$$\sum_{l=k_2+1}^{\infty} \tau_k \leqslant \frac{1}{\gamma^2}, \quad \sum_{l=k_2+1}^{\infty} \|d_{k-1}\|^2 < \infty$$

which is contradictory with (7.101).

(ii). We have

$$|g_k^T d_{k-1}| \leqslant c_1 \|d_{k-1}\|^2$$

for $c_1 = \eta$.

Since $\|d_k\| \to_{k\to\infty} 0$ thus there exists $k_3 \geqslant k_2$ such that

$$\|d_k\| \leqslant \frac{\sqrt{2}\gamma}{2c_1}$$

for $k \geqslant k_3$.

Therefore,

$$\tau_k \leqslant \frac{1 + 2c_1 + c_1^2}{\gamma} \|d_{k-1}\|^2.$$

Thus, we also have $\sum_{l=k_2+1}^{\infty} \tau_k < \infty$.

(iii). In this case we have

$$\begin{aligned} |g_{k+1}^T d_k| &= |\left(g_{k+1} - g_k\right)^T d_k + g_k^T d_k| \\ &\leqslant \|g_{k+1} - g_k\| \|d_k\| + |g_k^T d_k| \\ &\leqslant \alpha_k L \|d_k\|^2 + \|d_k\|^2 \\ &\leqslant (1 + \alpha_k L) \|d_k\|^2 \end{aligned}$$

(for $0 < \lambda_k < 1$). This leads to the conclusion stated in *(ii)*. □

Based on the above result we can also prove the following theorem.

Theorem 7.12. *If the assumptions of* Theorem 7.4 *are satisfied then the sequence* $\{x_k\}$ *generated by* Algorithm 7.1 *with* $\beta_k = 1$ *converges to the minimizer of* f.

Proof. In the proof of Theorem 7.4 we show that there exists $M_\alpha < \infty$ such that $\alpha_k \leqslant M_\alpha$. The proof follows then from part *(iii)* of Theorem 7.11 and the fact that the minimizer is also a point of attraction for the sequence $\{x_k\}$ (cf. Theorem 2.4). □

7.7 A Counter-Example

Global convergence results from previous sections are valid under the assumption that step-sizes are selected by the strong Wolfe conditions, or under the Wolfe conditions if they are uniformly bounded. Dai and Yuan in [43] construct the function with the aim of showing that the latter condition cannot be dispensed.

Following [43] we consider the function

$$f(x,y) = \begin{cases} 0, & \text{if } (x,y) = (0,0) \\ \frac{1}{4}\lambda\left(\frac{x}{\sqrt{x^2+y^2}}\right)(x+y)^2 + \frac{1}{4}\lambda\left(\frac{y}{\sqrt{x^2+y^2}}\right)(x-y)^2, & \text{otherwise} \end{cases}$$

where

$$\lambda(t) = \begin{cases} 1, & \text{if } |t| > \frac{\sqrt{3}}{2} \\ \frac{1}{2} + \frac{1}{2}\sin(b_1|t| + b_2), & \text{if } \frac{\sqrt{2}}{2} \leqslant |t| \leqslant \frac{\sqrt{3}}{4} \\ 0, & \text{if } |t| < \frac{\sqrt{2}}{2} \end{cases} \tag{7.102}$$

We assume that $b_1 = 2(\sqrt{3}+\sqrt{2})\pi$ and $b_2 = -(5/2+\sqrt{6})\pi$. One can check that the Frechet derivative of f exists for all $(x,y) \in \mathscr{R}^2$ (in order to show that for the point $(x,y) = (0,0)$ we have to refer to the definition of the Frechet derivative). Furthermore, one can show that ∇f is Lipschitz continuous and that $(0,0)$ is the point at which minimum of f is achieved. However, one can also show that

$$\lim_{x \to \bar{x}, y \to -\bar{x}} f(x,y) = 0 \tag{7.103}$$

$$\lim_{x \to \bar{x}, y \to \bar{x}} f(x,y) = 0 \tag{7.104}$$

for any $\bar{x} \neq 0$ (cf. Figs. 7.4–7.5).

Based on the function f we want to construct sequences $\{x_k\}$, $\{d_k\}$ which fulfill

$$x_k = \|x_k\| \begin{bmatrix} \cos\left(\frac{k-1}{2}\pi\right) \\ \sin\left(\frac{k-1}{2}\pi\right) \end{bmatrix} \tag{7.105}$$

$$\|x_k\| = \|x_{k-1}\| \tan\left(\frac{\pi}{4} - \tau_{k-1}\right), \ \tau_{k-1} \in \left(0, \frac{\pi}{8}\right) \tag{7.106}$$

$$g_k = \frac{1}{2}\|x_k\| \begin{bmatrix} \cos\left(\frac{k-1}{2}\pi + \frac{\pi}{4}\right) \\ \sin\left(\frac{k-1}{2}\pi + \frac{\pi}{4}\right) \end{bmatrix} \tag{7.107}$$

$$d_k = \|g_k\| \sin\tau_k \begin{bmatrix} \cos\left(\frac{k}{2}\pi + \frac{\pi}{4} + \tau_k\right) \\ \sin\left(\frac{k}{2}\pi + \frac{\pi}{4} + \tau_k\right) \end{bmatrix}. \tag{7.108}$$

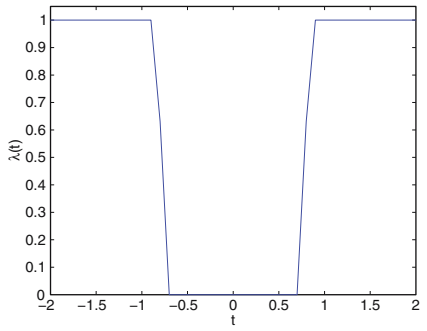

Fig. 7.4 The λ function

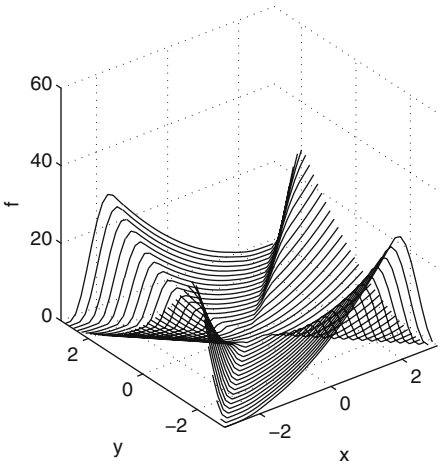

Fig. 7.5 The Dai–Yuan function

Suppose now that we can choose initial point x_1 in such a way that (7.105)–(7.108) hold for $k = 2$. Then, according to Dai and Yuan, (7.105) and (7.108) and the definition of f justify the existence of $\alpha_k > 0$ for which we have

$$x_{k+1} = \|x_{k+1}\| \begin{bmatrix} \cos \frac{k}{2}\pi \\ \sin \frac{k}{2}\pi \end{bmatrix} \tag{7.109}$$

$$\|x_{k+1}\| = \|x_k\| \tan\left(\frac{\pi}{4} - \tau_k\right) \tag{7.110}$$

which implies that

$$g_{k+1} = \frac{1}{2}\|x_{k+1}\| \begin{bmatrix} \cos\left(\frac{k}{2}\pi + \frac{\pi}{4}\right) \\ \sin\left(\frac{k}{2}\pi + \frac{\pi}{4}\right) \end{bmatrix}. \tag{7.111}$$

Simple calculations show that $g_{k+1}^T g_k = 0$ which implies that $\beta_k = 1$ irrespective of the rule for β_k used.

Next we refer to the rule

$$d_{k+1} = -(1-\lambda_k)g_{k+1} + \lambda_k d_k, \tag{7.112}$$

with λ_k defined by (7.60), which can be restated as

$$d_{k+1} = (1-\lambda_k)\left(-g_{k+1} + \xi_k d_k\right) \tag{7.113}$$

where

$$\xi_k = \frac{\lambda_k}{1-\lambda_k}.$$

Then, due to

$$g_{k+1}^T d_k = \|g_{k+1}\|\|d_k\|\cos\tau_k \tag{7.114}$$

$$\|g_{k+1}\| = \|g_k\|\tan\left(\frac{\pi}{4} - \tau_k\right) \tag{7.115}$$

$$\|d_k\| = \|g_k\|\sin\tau_k \tag{7.116}$$

and the equality

$$\tan(\alpha+\beta) = \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta} \tag{7.117}$$

we have

$$\begin{aligned}
\xi_k &= \frac{\|g_k\|^2 + g_{k+1}^T d_k}{\|d_k\|^2 + g_{k+1}^T d_k} \\
&= \frac{\tan^2\left(\frac{\pi}{4} - \tau_k\right) + \cos\tau_k \tan\left(\frac{\pi}{4} - \tau_k\right)\sin\tau_k}{\cos\tau_k \tan\left(\frac{\pi}{4} - \tau_k\right)\sin\tau_k + \sin^2\tau_k} \\
&= \frac{\tan\left(\frac{\pi}{4} - \tau_k\right)}{\sin\tau_k}\chi_k
\end{aligned}$$

where

$$\chi_k = \cos\tau_k - \sin^3\tau_k + \cos\tau_k\sin^2\tau_k,$$

since, according to (7.117)

$$\tan\left(\frac{\pi}{4} - \tau_k\right) = \frac{\cos\tau_k - \sin\tau_k}{\cos\tau_k + \sin\tau_k} \tag{7.118}$$

and

$$\begin{aligned}\chi_k &= \frac{\tan\left(\frac{\pi}{4}-\tau_k\right)+\sin\tau_k\cos\tau_k}{\tan\left(\frac{\pi}{4}-\tau_k\right)\cos\tau_k+\sin\tau_k}\\ &= \frac{(\cos\tau_k-\sin\tau_k)/(\cos\tau_k\sin\tau_k)+\sin\tau_k\cos\tau_k}{\sin\tau_k+\cos\tau_k\,(\cos\tau_k-\sin\tau_k)/(\cos\tau_k\sin\tau_k)}\\ &= \frac{\cos\tau_k-\sin\tau_k+\sin\tau_k\cos^2\tau_k+\cos\tau_k\sin^2\tau_k}{\sin\tau_k\cos\tau_k+\sin^2\tau_k+\cos^2\tau_k-\sin\tau_k\cos\tau_k}.\end{aligned}$$

This, due to (7.113), implies

$$d_{k+1} = \upsilon_k\left(\begin{bmatrix}\cos\left(\frac{k+2}{2}\pi+\frac{\pi}{4}\right)\\ \sin\left(\frac{k+2}{2}\pi+\frac{\pi}{4}\right)\end{bmatrix}+\chi_k\begin{bmatrix}\cos\left(\frac{k}{2}\pi+\frac{\pi}{4}+\tau_k\right)\\ \sin\left(\frac{k}{2}\pi+\frac{\pi}{4}+\tau_k\right)\end{bmatrix}\right) \tag{7.119}$$

where

$$\upsilon_k = (1-\lambda_k)\tan\left(\frac{\pi}{4}-\tau_k\right)\|g_k\|. \tag{7.120}$$

Since $\chi_k \geqslant 1$ for all $\tau_k \in (0,\pi/8)$, on the basis of (7.119), Dai and Yuan claim that

$$\frac{d_{k+1}}{\|d_{k+1}\|} = \begin{bmatrix}\cos\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\\ \sin\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\end{bmatrix}$$

with

$$\tau_{k+1} \in \left[-\frac{\pi}{4}+\tau_k, \frac{\tau_k}{2}\right].$$

From the relation $g_{k+1}^T d_{k+1} < 0$ we also have

$$\tau_{k+1} \in \left(0, \frac{\tau_k}{2}\right] \tag{7.121}$$

thus

$$\tau_{k+1} \in \left(0, \frac{\pi}{8}\right).$$

By the way d_{k+1} is constructed, $g_{k+1}^T d_{k+1} = -\|d_{k+1}\|^2$ which leads to

$$\begin{aligned}d_{k+1}^T g_{k+1} = \|d_{k+1}\|\|g_{k+1}\| \times &\\ \left(\cos\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\cos\left(\frac{k}{2}\pi+\frac{\pi}{4}\right)+\right.&\\ \left.\sin\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\sin\left(\frac{k}{2}\pi+\frac{\pi}{4}\right)\right)&\end{aligned} \tag{7.122}$$

and to

$$\begin{aligned}\|d_{k+1}\|\|g_{k+1}\|\cos\left(\frac{\pi}{2}+\tau_{k+1}\right) &= -\sin\tau_{k+1}\|d_{k+1}\|\|g_{k+1}\| \\ &= -\|d_{k+1}\|^2\end{aligned}$$

and eventually to

$$d_{k+1} = \|g_{k+1}\|\sin\tau_{k+1}\begin{bmatrix}\cos\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\\ \sin\left(\frac{k+1}{2}\pi+\frac{\pi}{4}+\tau_{k+1}\right)\end{bmatrix}.$$

Therefore, relations (7.105)–(7.108) are also valid for $k+1$ and by the induction (7.105)–(7.108) are true for any $k \geqslant 2$. The Wolfe curvature condition is satisfied since $g_{k+1}^T d_k > 0$ (cf. (7.114)).

Consider now the Wolfe descent condition. We have

$$-\alpha_k g_k^T d_k = \alpha_k\|d_k\| = \|d_k\|\|x_{k+1}-x_k\|. \tag{7.123}$$

But, according to (7.106) and (7.118),

$$\begin{aligned}\|x_{k+1}-x_k\| &= \|x_k\|\left\|\tan\left(\frac{\pi}{4}-\tau_k\right)\begin{bmatrix}\cos\frac{k}{2}\pi\\ \sin\frac{k}{2}\pi\end{bmatrix}-\begin{bmatrix}\cos\frac{k-1}{2}\pi\\ \sin\frac{k-1}{2}\pi\end{bmatrix}\right\| \\ &= \frac{\|x_k\|}{\cos\tau_k+\sin\tau_k}\left\|\begin{bmatrix}\cos\left(\frac{k}{2}\pi-\tau_k\right)-\sin\left(\frac{k}{2}\pi+\tau_k\right)\\ \sin\left(\frac{k}{2}\pi-\tau_k\right)+\sin\left(\frac{k}{2}+\tau_k\right)\end{bmatrix}\right\| \\ &= \frac{\sqrt{2}\|x_k\|}{\cos\tau_k+\sin\tau_k}\end{aligned}$$

and (7.123) can be stated as

$$-\alpha_k g_k^T d_k = \frac{2\sqrt{2}\|g_k\|\|d_k\|}{\cos\tau_k+\sin\tau_k} \tag{7.124}$$

which implies

$$\begin{aligned}\alpha_k &= \frac{2\sqrt{2}\|g_k\|}{(\cos\tau_k+\sin\tau_k)\|d_k\|} \\ &= \frac{2\sqrt{2}}{(\cos\tau_k+\sin\tau_k)\sin\tau_k}.\end{aligned} \tag{7.125}$$

Furthermore, due to the definition f, we have

$$f(x_{k+1})-f(x_k) = \frac{1}{2}\left(\|g_{k+1}\|^2-\|g_k\|^2\right). \tag{7.126}$$

Due to (7.110)–(7.111) and (7.118) we can write

$$\begin{aligned}\|g_k\|^2 - \|g_{k+1}\|^2 &= \|g_k\|^2 \left(1 - \tan^2\left(\frac{\pi}{4} - \tau_k\right)\right) \\ &= \|g_k\|^2 \frac{4\cos\tau_k \sin\tau_k}{(\cos\tau_k + \sin\tau_k)^2}\end{aligned}$$

which together with (7.108) and (7.126) give

$$\begin{aligned}f(x_{k+1}) - f(x_k) &= -\frac{2\cos\tau_k \|g_k\| \|d_k\|}{(\cos\tau_k + \sin\tau_k)^2} \\ &= \delta_k \alpha_k g_k^T d_k \end{aligned} \tag{7.127}$$

with

$$\delta_k = \frac{\cos\tau_k}{\sqrt{2}(\cos\tau_k + \sin\tau_k)}. \tag{7.128}$$

Since (7.121) holds we obtain

$$\lim_{k\to\infty} \tau_k = 0 \tag{7.129}$$

therefore we can choose k_1 in such a way that $\delta_k > \mu$ for all $k \geqslant k_1$ which together with (7.127)–(7.128) imply that the Wolfe descent condition holds for all $k \geqslant k_1$.

From (7.106) and (7.121) we deduce

$$\|x_k\| = \|x_2\| \prod_{l=2}^{k-1} \tan\left(\frac{\pi}{4} - \tau_l\right)$$

thus

$$\lim_{k\to\infty} \|x_k\| = \gamma \|x_2\|$$

with

$$\gamma = \prod_{l=2}^{\infty} \tan\left(\frac{\pi}{4} - \tau_l\right) > 0.$$

This implies that any accumulation point of $\{x_k\}$ is not equal to $(0,0)$.

On the other hand, from (7.125) and (7.129), we have

$$\lim_{k\to\infty} \alpha_k = +\infty.$$

This shows that condition $\alpha_k \leqslant M < +\infty$ is essential in establishing global convergence of the method of shortest residuals if the Wolfe conditions are applied in directional minimization. However, the counter-example does not address the stationary points indicated by (7.103)–(7.104).

7.8 Numerical Experiments: First Comparisons

First, we show Algorithm 7.1 performance on standard, difficult problems listed below. Some other numerical results obtained on problems from the CUTE collection are discussed later in the chapter. The *ad hoc* implementation was done on SUN SPARC 1 workstation in FORTRAN using *double precision* accuracy. In order to verify Algorithm 7.1 with the formula for β_k given by (7.52) we run it on several standard test problems.

1. Rosenbrock function [188]: $f = 100(x_2 - x_1^2)^2 + (1-x_1)^2$, $x \in \mathscr{R}^2$.
2. Extended Rosenbrock function [191]: $f = \sum_{i=2}^{10}\{100(x_i - x_{i-1}^2)^2 + (1-x_i)^2\}$, $x \in \mathscr{R}^{10}$.
3. Powell function: [164] $f = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4, x \in \mathscr{R}^4$.
4. Cube function [103]: $f = 100(x_2 - x_1^3)^2 + (1-x_1)^2$, $x \in \mathscr{R}^2$.
5. Beale function [103]: $f = \sum_{i=1}^{3}(c_i - x_1(1-x_2^i))^2$, $c_1 = 1.5$, $c_2 = 2.25$, $c_3 = 2.625$, $x \in \mathscr{R}^2$.
6. Wood function [34]: $f = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1\{(x_2-1)^2 + (x_4-1)^2\} + 19.8(x_2-1)(x_4-1)$, $x \in \mathscr{R}^4$.
7. Watson function [117]: $f = \sum_{i=1}^{30}\{\sum_{j=1}^{10}(j-1)x_j y_i^{j-2} - (\sum_{j=1}^{10} x_j y_i^{j-1})^2 - 1\}^2$, $y_i = (i-1)/29$, $x \in \mathscr{R}^{10}$.
8. Oren–Spedicato function [199]: $f = \{\sum_{i=1}^{20}(ix_i^2 + c)\}^r$, $c = 0$, $r = 2$, $x \in \mathscr{R}^{20}$.

We use the line search rules (7.18)–(7.19) supplemented by a quadratic interpolation search. Our directional minimization was in fact procedure described in [117] with different parameters, every time this procedure finished the conditions (7.18)–(7.19) were checked with parameters: $\mu = 0.0001$, $\eta = 0.9$. If these conditions were not satisfied, we applied the algorithm described in the proof of Mifflin's line search procedure given in the previous chapter and applied in the proof of Lemma 2.4.

The results of our computations, following [174], are shown in Table 7.1, where ITER means the number of iterations, IFUN the number of function evaluations and IGRAD the number of gradient evaluations. Our stopping criterion was

$$|(g_k)_i| \leqslant 1.0^{-5},\ i = 1,\ldots,n.$$

The numbers in the first column are initial points for Algorithm 7.1.

In the last two columns we show the results (if available) for two efficient methods based on the conjugate gradient approach (which use the same stopping criterion). The method described in [117] requires as many gradient calculations as the number of iterations, while for the method in [191] (CONMIN) this number is equal to the number of function evaluations. The first number in the bracket is the number of iterations, the second is the number of function evaluations. The symbol "FAIL" denotes that the method failed to converge. It should be noted that the method presented in [191] requires three additional vectors to calculate a direction of descent.

Table 7.1 Performance of Algorithm 7.1 – small problems

Problem	ITER	IFUN	IGRAD	[117]	[191]
Rosenbrock					
(−1.2, 1.)	38	127	56	∼	∼
Extended Rosenbrock					
(−1.2,1,...,1.)	59	130	59	(22,138)	FAIL
Powell					
(−3,−1.,0.,1.)	64	156	65	∼	(64,160)
Cube					
(−1.2,1.)	42	106	43	(13,169)	∼
Beale					
(0.,0.)	26	69	27	(7,60)	∼
Wood					
(−3.,1.,−3.,1.)	106	248	108	∼	(113,195)
(−3.,−1.,−3.,−1.)	67	148	67	(20,134)	(101,235)
(−1.2,1.,−1.2,1.)	145	353	152	∼	(93,219)
(−1.2,1.,1.2,1.)	98	232	99	∼	(48,118)
Watson					
(0.,0.,...,0.)	18	39	18	(383,3416)	FAIL
Oren-Spedicato					
(1.,1.,...,1.)	20	46	20	(15,99)	∼

In order to verify the effectiveness of our algorithm we also have tested it on problems from the CUTE collection [13]. We tried it on two sets of problems – the first is described in Table 3.1, and the other one in Table 7.4.

Algorithms 7.1 and 7.2 have been implemented. In both implementations we used directional minimization procedure by Moré and Thuente described in Chap. 2– Algorithm 7.2 required several obvious modifications of the procedure due to the fact that its direction minimization rules are different from standard ones employed by Algorithm 7.2.

The procedure discussed in Chap. 2 finds the step-size $\alpha_k > 0$, in a finite number of iterations, such that the strong Wolfe conditions (7.68), (7.91) are satisfied. We replaced these conditions by our directional minimization conditions (7.18)–(7.19). It is not difficult to show that the Moré–Thuente procedure modified in this way finds $\alpha_k > 0$ satisfying (7.18)–(7.19) in a finite number of steps. We did also similar changes to the Moré–Thuente procedure in order to use it in Algorithm 7.2.

The stopping criterion used to get numerical results reported from now on in this chapter was

$$\|g_k\| / \max(1, \|x_k\|) \leqslant 10^{-5}$$

which is employed in L-BFGS algorithm [125].

In figures and tables presented below by A1-weak Algorithm we mean Algorithm 7.1 which employs conditions (7.18)–(7.19). A1-strong Algorithm is Algorithm 7.1 in which line search rule (7.19) is replaced by

$$\left|g(x_k + \alpha_k d_k)^T d_k\right| \leqslant \eta \|d_k\|^2.$$

A2-weak Algorithm is Algorithm 7.2 with the Wolfe conditions as line search rules, while A2-strong Algorithm is Algorithm 7.2 with the strong Wolfe conditions.

The comparison of the performance of two algorithms is given in Tables 7.2–7.3. The problems, on which we performed these tests are presented in Table 3.1.

Table 7.2 Numerical results for CG+, Algorithm 7.1 and Algorithm 7.2 on problems from Table 3.1

Problem	CG+ (method = 2)			A1 strong Wolfe			A2 strong Wolfe		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	8	29	0.01	17	53	0.01	11	30	0.01
BRYBND	57	129	0.36	28	68	0.20	28	68	0.19
CHAINWOO	FAIL	FAIL	FAIL	376	727	4.27	361	707	4.22
DIXON3DQ	10000	20006	25.91	2474	2939	5.31	2369	2793	5.42
DQDRTIC	5	15	0.02	5	15	0.03	5	15	0.02
DQRTIC	12	45	0.02	11	44	0.01	13	49	0.01
EIGENALS	1233	2471	0.34	615	1132	0.16	740	1355	0.19
EXTROSNB	3723	7602	0.02	2886	5546	0.03	2538	4941	0.03
FLETCHBV	2630	9547	1.92	2964	8193	1.66	3115	8524	1.73
FLETCHCR	773	1560	0.05	684	1176	0.05	739	1225	0.03
FMINSURF	789	1583	8.58	731	960	5.70	747	982	5.95
GENHUMPS	2412	5165	5.89	2741	5052	5.28	2765	5034	5.36
GENROSE	1115	2258	0.31	1128	1982	0.31	1136	2011	0.33
HILBERTA	6	14	0.01	6	14	0.01	6	14	0.01
LIARWD	15	39	0.16	19	57	0.22	26	71	0.30
MANCINO	11	27	1.66	10	24	1.48	11	26	1.59
MOREBV	60	121	0.16	27	46	0.06	27	46	0.06
NONCVXU2	1238	2483	1.16	3308	4712	2.36	2880	4045	2.05
NONCVXUN	FAIL	FAIL	FAIL	2965	3672	20.44	2983	3693	21.01
NONDIA	5	27	0.13	32	106	0.41	10	30	0.12
NONDQUAR	541	1140	1.94	317	670	1.14	305	638	1.11
POWELLSG	141	340	0.55	97	219	0.34	62	135	0.20
POWER	40	89	0.01	37	78	0.01	37	80	0.01
QUARTC	22	78	0.09	12	52	0.06	26	91	0.12
SCHMVETT	34	76	0.84	29	64	0.61	29	64	0.59
SENSORS	25	59	0.66	21	50	0.44	21	50	0.53
SPARSINE	4412	8829	4.55	10067	18120	8.88	9199	16473	9.34
SPMSRTLS	186	378	2.73	146	271	1.73	146	271	2.03
SROSENBR	9	26	0.05	12	35	0.06	7	20	0.05
TOINTGSS	4	20	0.12	4	20	0.09	3	10	0.03
TQUARTIC	11	32	0.08	20	69	0.17	28	85	0.25
TRIDIA	1154	2311	3.47	3491	4283	7.70	3550	4337	9.52
VAREIGVL	305	621	2.84	167	282	1.16	173	290	1.44
WOODS	287	628	1.75	93	205	0.56	78	189	0.59

Table 7.3 Numerical results for CG+, Algorithm 7.1 and Algorithm 7.2 on problems from Table 3.1

Problem	CG+ (method = 3)			A1–weak Wolfe			A2–weak Wolfe		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	8	28	0.01	25	78	0.01	17	45	0.01
BRYBND	28	70	0.17	36	80	0.22	33	76	0.22
CHAINWOO	FAIL	FAIL	FAIL	272	517	3.05	327	590	3.56
DIXON3DQ	10000	20006	25.72	1201	1693	2.89	953	1364	2.44
DQDRTIC	5	15	0.02	5	15	0.02	5	15	0.02
DQRTIC	11	44	0.01	18	64	0.02	18	59	0.01
EIGENALS	1233	2471	0.33	517	877	0.11	526	866	0.12
EXTROSNB	3085	7901	0.03	1516	2732	0.02	1315	2393	0.02
FLETCHBV	2941	10754	2.14	1988	3705	0.75	2767	5333	1.09
FLETCHCR	773	1560	0.05	755	1154	0.03	730	1102	0.05
FMINSURF	789	1583	8.56	915	1315	7.73	972	1364	8.16
GENHUMPS	2780	5793	6.88	2240	3635	4.14	2903	4190	4.95
GENROSE	1115	2258	0.31	1530	2387	0.38	1437	2230	0.36
HILBERTA	6	14	0.01	6	14	0.01	6	14	0.01
LIARWD	14	44	0.17	37	106	0.42	25	63	0.27
MANCINO	10	25	1.53	10	24	1.47	11	26	1.59
MOREBV	60	121	0.16	23	35	0.05	25	38	0.05
NONCVXU2	1238	2483	1.16	1697	2406	1.24	1542	2176	1.16
NONCVXUN	FAIL	FAIL	FAIL	2295	3202	17.94	2015	2766	15.86
NONDIA	6	26	0.11	31	107	0.44	10	30	0.11
NONDQUAR	738	1589	2.34	354	647	1.14	301	587	1.05
POWELLSG	195	437	0.61	78	174	0.30	70	155	0.25
POWER	42	88	0.01	42	85	0.01	42	85	0.01
QUARTC	13	58	0.06	29	94	0.13	37	101	0.12
SCHMVETT	34	76	0.70	42	71	0.69	41	70	0.69
SENSORS	25	59	0.56	29	59	0.84	28	57	0.81
SPARSINE	4412	8829	4.06	9995	14757	7.38	6459	9583	4.86
SPMSRTLS	186	378	2.36	172	270	1.78	169	269	1.80
SROSENBR	8	26	0.05	12	35	0.08	15	38	0.06
TOINTGSS	4	20	0.09	4	20	0.11	3	10	0.05
TQUARTIC	9	29	0.08	36	95	0.23	21	55	0.15
TRIDIA	1154	2311	2.98	4626	6673	11.34	5325	7787	13.81
VAREIGVL	305	620	2.45	197	293	1.20	153	234	0.98
WOODS	231	485	1.19	108	258	0.66	101	231	0.61

The presented below efficiency profiles show that the method of shortest residuals is competitive to CG+ code which is the efficient implementation of the conjugate gradient algorithm introduced in [78] and described in Sect. 2.9. The option of CG+ code 'method' refers to the version of a conjugate gradient algorithm. 'method = 2' means the Polak–Ribière version while 'method = 3' assumes that the modified Polak–Ribière formula – (2.124)–(2.124) is used. Both Algorithms 7.1 and 7.2 are superior to the code CG+ with respect to CPU time and the number of function evaluations with the slightly greater efficiency recorded by Algorithm 7.2 (Figs.7.6–7.8).

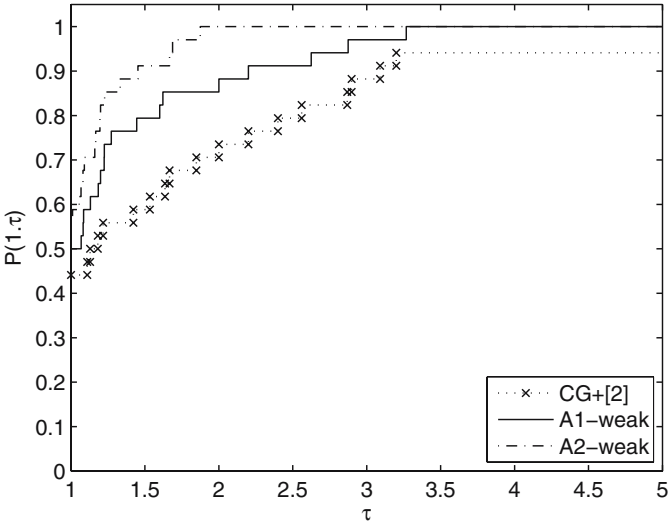

Fig. 7.6 Cumulative distribution ratio of CPU time: comparison of the proposed A1–weak Algorithm and A2–weak Algorithm against the CG (method = 2)–problems described in Table 3.1

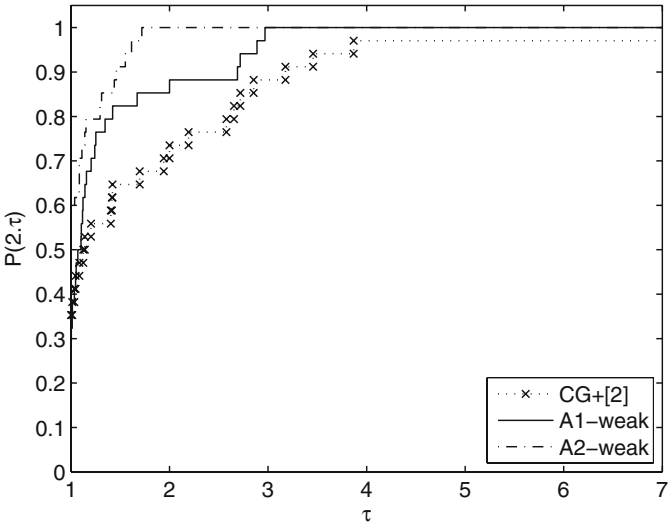

Fig. 7.7 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed A1–weak Algorithm and A2–weak Algorithm against the CG (method = 2)–problems described in Table 3.1

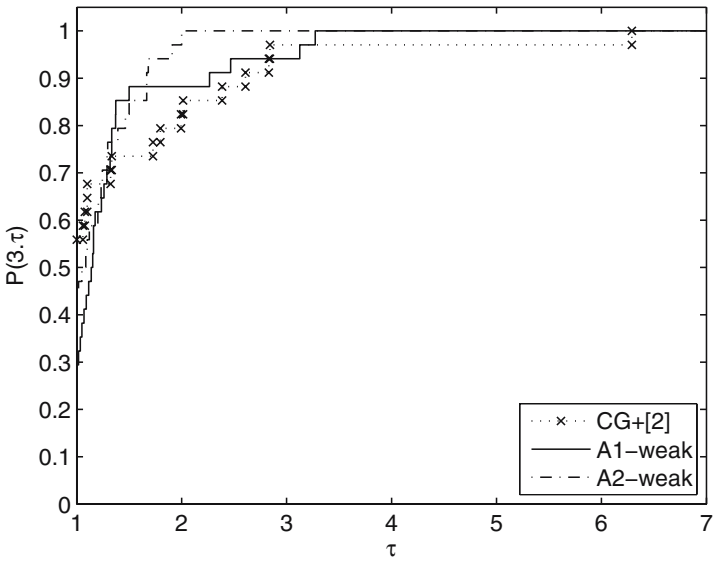

Fig. 7.8 Cumulative distribution ratio of the iteration number: comparison of the proposed A1–weak Algorithm and A2–weak Algorithm against the CG (method = 2)–problems described in Table 3.1

Table 7.4 Dimensions of unconstrained problems used for testing

Problems	Dimension
EXTROSNB, HILBERTA	10
MANCINO, SENSORS,	100
GENROSE	500
BROWNAL, FLETCHCR, SPARSINE,	1000
EIGENALS	2550
DQDRTIC, DQRTIC, GENHUMPS, MOREBV, SINQUAD, VAREIGVL	5000
BROYDN7D, BRYBND, CHAINWOO, DIXON3DQ, FLETCHBV, LIARWD, NONCVXUN, NONDIA, NONCVXU2, NONDQUAR, POWER, POWELLSG, QUARTIC, SCHMVETT, SPMSRTLS, SROSENBR, TOINTGS, TQUARTIC, TRIDIA, WOODS	10000
FMINSURF	15625

The results presented in Tables 7.2–7.3 are drawn from the paper [183] (see also [184]). Below we also present results obtained by the implementations of Algorithm 7.1, Algorithm 7.2, CONMIN and CG+ codes on the problems from the CUTE collection – they are shown in Tables 7.5, 7.6.

Table 7.5 Numerical results for CG+, Algorithm 7.1 and Algorithm 7.2 on problems from Table 7.4

Problem	CG+ (method = 3)			A2 weak Wolfe			A1 weak Wolfe		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	3	37	0.54	3	23	0.34	2	29	0.43
BROYDN7D	2791	5631	79.54	2818	4754	68.38	3038	4229	61.08
BRYBND	70	156	0.87	48	106	0.61	76	149	0.85
CHAINWOO	FAIL	FAIL	FAIL	371	727	3.40	317	588	2.74
DIXON3DQ	10000	20006	29.76	100139	124422	239.85	6441	9258	16.29
DQDRTIC	5	15	0.02	6	17	0.03	6	17	0.03
DQRTIC	13	53	0.04	19	72	0.06	23	72	0.06
EIGENALS	29048	58102	1020.52	14338	24102	423.89	25752	37288	656.73
EXTROSNB	4492	9128	0.04	2909	5659	0.02	1585	2936	0.02
FLETCHBV	7752	15526	78.90	15093	17501	97.75	3035	4110	22.06
FLETCHCR	4328	8681	2.56	4475	6914	2.26	6720	9773	3.26
FMINSURF	915	1835	16.34	1240	1570	14.99	2427	2427	23.04
GENHUMPS	7408	15053	48.81	7587	13145	44.26	7238	9885	33.08
GENROSE	1117	2261	0.33	1119	1982	0.32	1523	2370	0.37
HILBERTA	6	14	0.01	6	14	0.01	6	14	0.01
LIARWD	14	44	0.16	26	71	0.25	37	106	0.38
MANCINO	10	25	0.31	11	26	0.32	11	26	0.32
MOREBV	60	121	0.17	58	90	0.14	46	69	0.11
NONCVXU2	5100	10210	46.71	10194	13430	66.88	4407	6213	39.07
NONCVXUN	30274	60560	276.19	41397	51246	256.36	7028	9885	47.27
NONDIA	6	26	0.11	10	30	0.14	33	111	0.48
NONDQUAR	1445	2963	4.82	917	1838	3.35	475	915	1.63
POWELLSG	409	972	1.65	153	340	0.65	295	680	1.25
POWER	380	769	0.88	456	720	1.06	617	902	1.29
QUARTC	13	58	0.08	27	93	0.14	29	94	0.15
SCHMVETT	34	76	0.72	35	74	0.71	35	75	0.72
SENSORS	25	50	0.48	21	50	0.41	30	61	0.50
SPARSINE	4412	8829	4.66	11650	20808	11.49	10636	15737	8.73
SPMSRTLS	186	378	1.85	183	334	1.72	218	340	1.76
SROSENBR	8	26	0.05	10	29	0.06	13	37	0.07
TOINTGSS	4	20	0.09	3	10	0.05	4	20	0.09
TQUARTIC	9	29	0.07	20	69	0.18	36	95	0.24
TRIDIA	1155	2313	3.42	4185	5144	9.96	5274	7514	13.26
VAREIGVL	305	620	2.55	391	547	2.35	300	449	1.89
WOODS	231	485	1.21	91	216	0.58	113	260	0.68

7.9 Numerical Experiments: Comparison to the Memoryless Quasi-Newton Method

This section presents the comparison of the implementations of several versions of the method of shortest residuals with the efficient implementation of the memoryless quasi–Newton algorithm presented in Chap. 3 – we refer to it as CONMIN code. Recorded data are shown in Table 7.6 which is the basis for the efficiency profiles presented in Figs. 7.9–7.20. CONMIN uses three more vectors than the Polak–Ribiére version of a conjugate gradient algorithm so it is not surprising that it is significantly more efficient than the method of shortest residuals and CG+ code. In Table 7.7 we compare CONMIN, CG+ and L-BFGS codes.

Table 7.6 Numerical results for CONMIN and CG+ on problems from Table 7.4

Problem	CONMIN			CG+ (method = 2)			CG+ (method = 3)		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	2	4	0.08	3	37	0.53	3	37	0.54
BROYDN7D	2842	5725	87.01	2791	5637	79.52	2791	5631	79.54
BRYBND	33	66	0.45	70	156	0.86	70	156	0.87
CHAINWOO	210	426	2.53	30766	63605	287.05	FAIL	FAIL	FAIL
DIXON3DQ	5647	11441	28.51	10000	20006	29.59	10000	20006	29.76
DQDRTIC	7	15	0.04	5	15	0.02	5	15	0.02
DQRTIC	13	27	0.03	15	56	0.04	13	53	0.04
EIGENALS	5564	11190	197.83	29048	58102	1014.06	29048	58102	1020.52
EXTROSNB	1139	2318	0.02	22668	45461	0.20	4492	9128	0.04
FLETCHBV	7624	15484	100.92	7752	15526	78.40	7752	15526	78.90
FLETCHCR	5372	10854	4.56	4328	8681	2.54	4328	8681	2.56
FMINSURF	950	1926	21.25	915	1835	16.28	915	1835	16.34
GENHUMPS	9169	18580	69.72	7588	15471	50.00	7408	15053	48.81
GENROSE	1118	2263	0.48	1117	2261	0.33	1117	2261	0.33
HILBERTA	6	13	0.01	6	14	0.01	6	14	0.01
LIARWD	14	30	0.14	15	39	0.15	14	44	0.16
MANCINO	9	19	0.24	11	27	0.34	10	25	0.31
MOREBV	53	108	0.23	60	121	0.17	60	121	0.17
NONCVXU2	4032	8153	49.22	5100	10210	46.64	5100	10210	46.71
NONCVXUN	8351	16909	101.31	30724	60560	273.77	30274	60560	276.19
NONDIA	8	19	0.19	5	27	0.12	6	26	0.11
NONDQUAR	394	802	2.36	1394	2825	4.60	1445	2963	4.82
POWELLSG	60	121	0.34	784	1787	3.07	409	972	1.65
POWER	464	937	2.40	380	769	0.89	380	769	0.88
QUARTC	14	29	0.07	22	78	0.12	13	58	0.08
SCHMVETT	23	50	0.71	39	76	0.71	34	76	0.72
SENSORS	23	50	0.41	25	59	0.48	25	50	0.48
SPARSINE	5377	10833	7.39	4412	8829	4.68	4412	8829	4.66
SPMSRTLS	185	373	2.38	186	378	1.84	186	378	1.85
SROSENBR	7	15	0.04	9	26	0.05	8	26	0.05
TOINTGSS	6	13	0.07	4	20	0.09	4	20	0.09
TQUARTIC	9	30	0.09	11	32	0.08	9	29	0.07
TRIDIA	3702	7497	22.02	1155	2313	3.43	1155	2313	3.42
VAREIGVL	297	599	2.88	305	621	2.54	305	620	2.55
WOODS	55	116	0.41	296	644	1.61	231	485	1.21

7.10 Numerical Experiments: Comparison to the Limited Memory Quasi-Newton Method

In this section implementations of several versions of the method of shortest residuals are compared to L-BFGS code which is the implementation of the limited memory quasi–Newton algorithm introduced in [125]. Table 7.8 and Figs. 7.21–7.23 clearly indicate on L-BFGS as a clear winner both in terms of CPU time and the number of function evaluations. However, L-BFGS uses $2m+4$ vectors (in our calculations $m=5$) in contrast to 4 in the case of the method of shortest residuals.

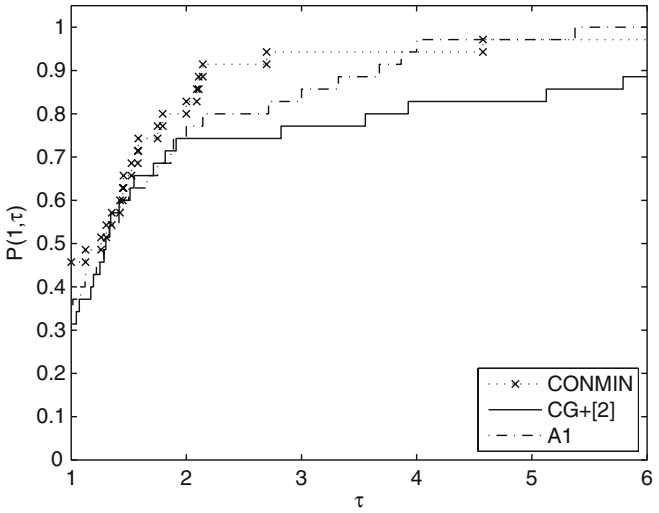

Fig. 7.9 Cumulative distribution ratio of CPU time: comparison of the proposed A1–weak Algorithm, CG+ (with method = 2) and CONMIN – problems described in Table 7.4

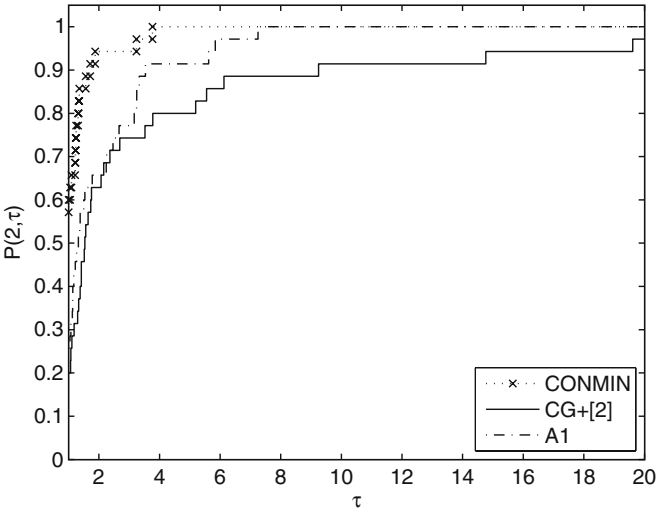

Fig. 7.10 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed A1–weak Algorithm, CG+ (with method = 2) and CONMIN – problems described in Table 7.4

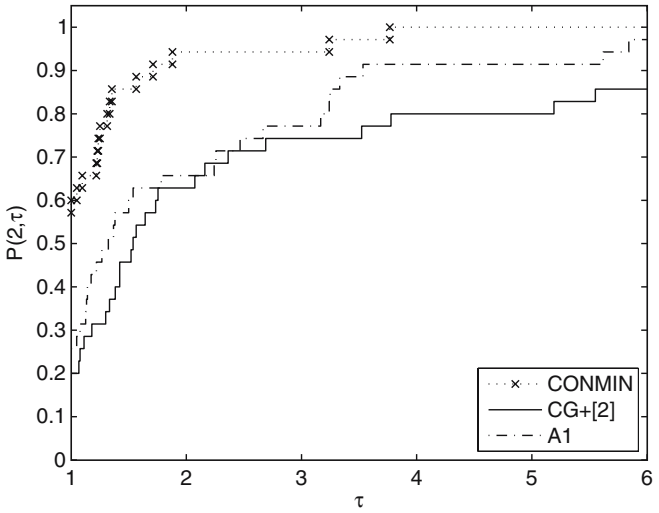

Fig. 7.11 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed A1–weak Algorithm, CG+ (with method = 2) and CONMIN – problems described in Table 7.4

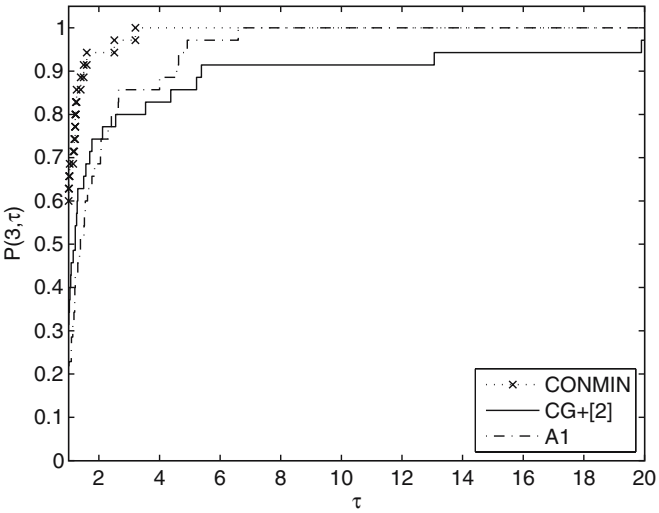

Fig. 7.12 Cumulative distribution ratio of the number of iterations: comparison of the proposed A1–weak Algorithm, CG+ (with method = 2) and CONMIN – problems described in Table 7.4

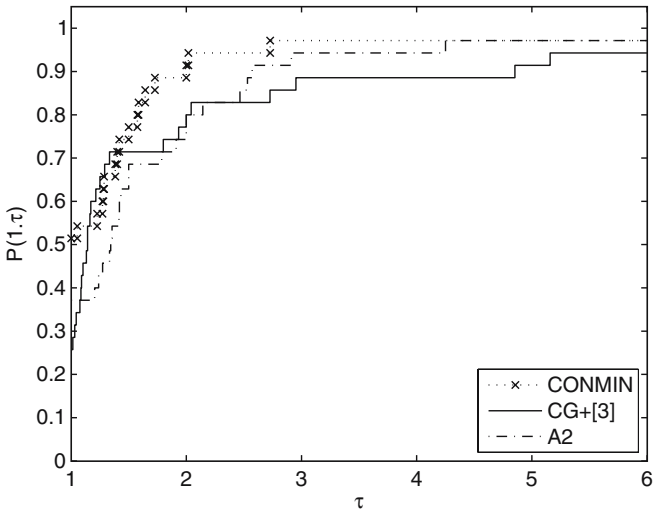

Fig. 7.13 Cumulative distribution ratio of CPU time: comparison of the proposed A2–weak Algorithm, CG+ (with method = 3) and CONMIN – problems described in Table 7.4

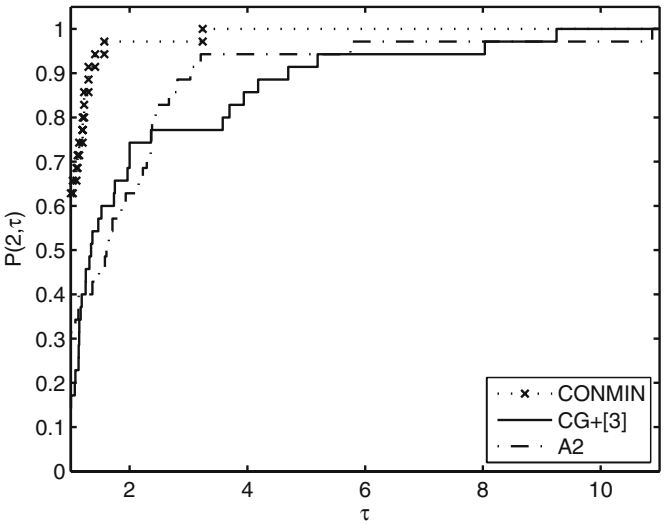

Fig. 7.14 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed A2–weak Algorithm, CG+ (with method = 3) and CONMIN – problems described in Table 7.4

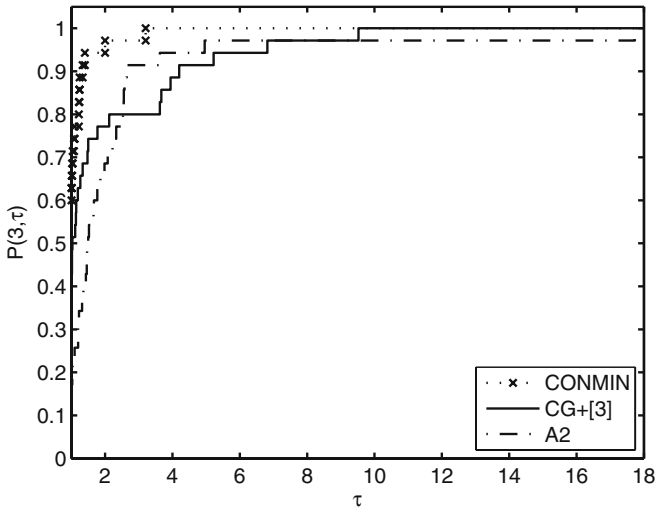

Fig. 7.15 Cumulative distribution ratio of the number of iterations: comparison of the proposed A2–weak Algorithm, CG+ (with method = 3) and CONMIN – problems described in Table 7.4

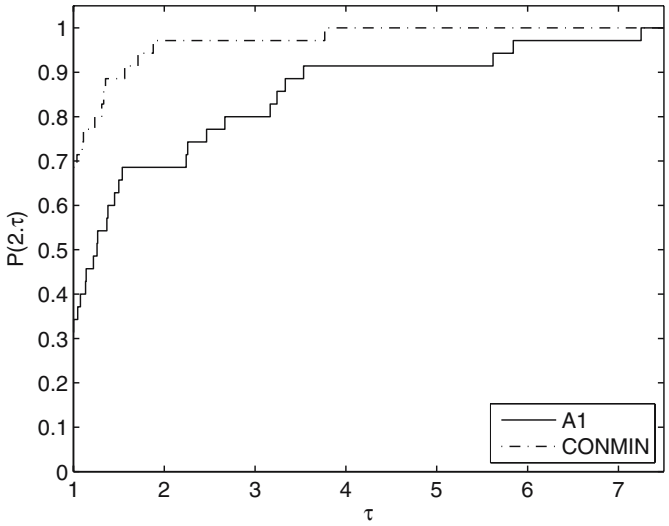

Fig. 7.16 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm and CONMIN

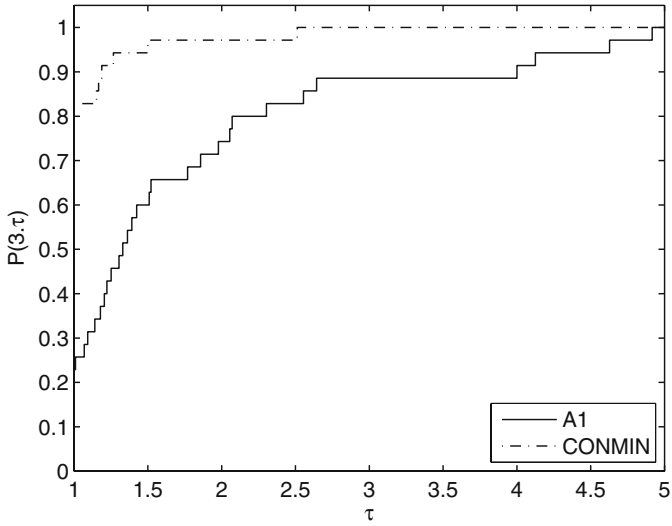

Fig. 7.17 Cumulative distribution ratio of the number of function iterations: comparison of A1–weak Algorithm and CONMIN

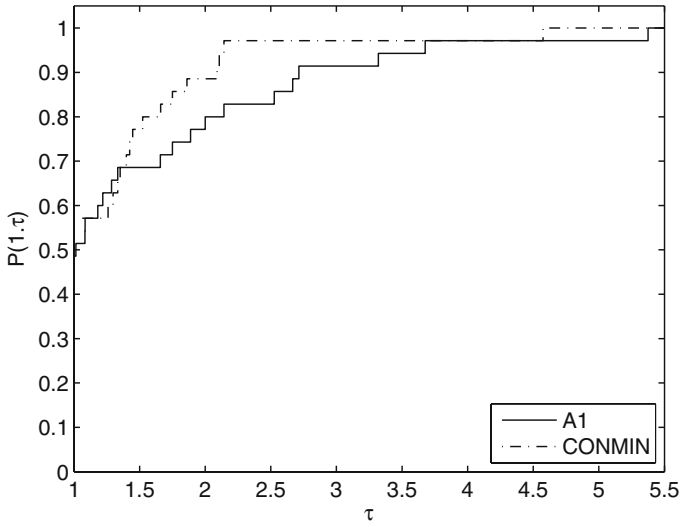

Fig. 7.18 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm and CONMIN

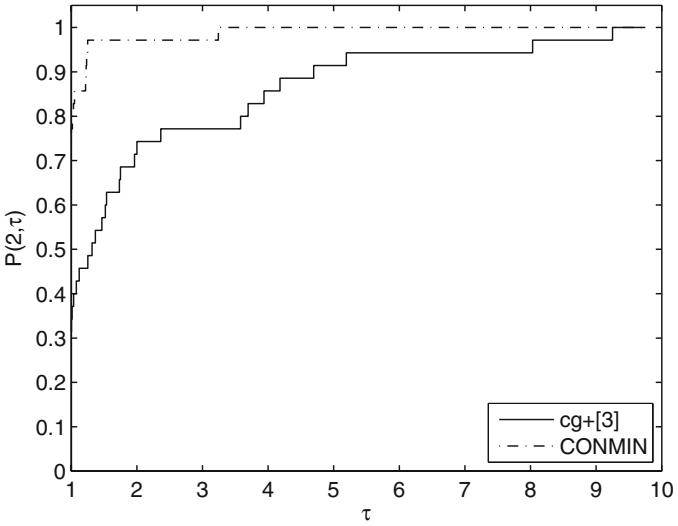

Fig. 7.19 Cumulative distribution ratio of the number of function evaluations: comparison of CG+ (with method = 3) and CONMIN

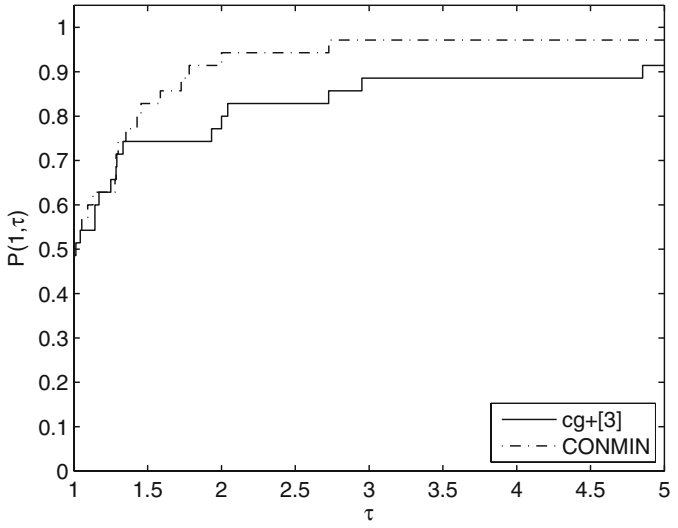

Fig. 7.20 Cumulative distribution ratio of CPU time: comparison of CG+ (with method = 3) and CONMIN

Table 7.7 Numerical results: comparison of CONMIN to CG+ and L-BFGS on problems described in Table 7.4

Problem	CONMIN		CG+ method = 3		L-BFGS $m = 5$	
	IF	CPU	IF	CPU	IF	CPU
BROWNAL	4	0.08	37	0.54	15	0.23
BROYDN7D	5725	87.01	5631	79.54	3158	48.92
BRYBND	66	0.45	156	0.87	52	0.35
CHAINWOO	426	2.53	FAIL	FAIL	468	2.70
DIXON3DQ	11441	28.51	20006	29.76	10715	30.73
DQDRTIC	15	0.04	15	0.02	22	0.44
DQRTIC	27	0.03	53	0.04	47	0.06
EIGENALS	11190	197.83	58102	1020.52	9938	180.51
EXTROSNB	2318	0.02	9128	0.04	1076	0.01
FLETCHBV	15484	100.92	15526	78.90	10102	65.34
FLETCHCR	10854	4.56	8681	2.56	5685	2.24
FMINSURF	1926	21.25	1835	16.34	1192	13.32
GENHUMPS	18580	69.72	15053	48.81	11001	40.29
GENROSE	2263	0.48	2261	0.33	1234	0.24
HILBERTA	13	0.01	14	0.01	31	0.01
LIARWD	30	0.14	44	0.16	28	0.13
MANCINO	19	0.24	25	0.31	15	0.19
MOREBV	108	0.23	121	0.17	45	0.10
NONCVXU2	8153	49.22	10210	46.71	4013	23.93
NONCVXUN	16909	101.31	60560	276.19	12879	76.24
NONDIA	19	0.19	26	0.11	23	0.13
NONDQUAR	802	2.36	2963	4.82	558	1.62
POWELLSG	121	0.34	972	1.65	73	0.22
POWER	937	2.40	769	0.88	451	1.14
QUARTC	29	0.07	58	0.08	48	0.13
SCHMVETT	50	0.71	76	0.72	48	0.52
SENSORS	50	0.41	50	0.48	23	0.19
SPARSINE	10833	7.39	8829	4.66	8376	5.36
SPMSRTLS	373	2.38	378	1.85	213	1.35
SROSENBR	15	0.04	26	0.05	20	0.06
TOINTGSS	13	0.07	20	0.09	26	0.14
TQUARTIC	30	0.09	29	0.07	27	0.10
TRIDIA	7497	22.02	2313	3.42	3150	9.00
VAREIGVL	599	2.88	620	2.55	22	0.10
WOODS	116	0.41	485	1.21	121	0.42

7.11 Numerical Experiments: Comparison to the Hager–Zhang Algorithm

We also compare the method of shortest residuals with the code CG-DESCENT discussed in Chap. 2 and introduced in [91]. The tests were run on the set of problems presented in Table 7.4 and are given in Table 7.9.

Table 7.8 Numerical results–comparison of Algorithm 7.1 and Algorithm 7.2 to L-BFGS code on problems from Table 7.4

Problem	A1 weak Wolfe		A2 weak Wolfe		L-BFGS $m = 5$	
	IF	CPU	IF	CPU	IF	CPU
BROWNAL	29	0.43	23	0.34	15	0.23
BROYDN7D	4229	61.08	4754	68.38	3158	48.92
BRYBND	149	0.85	106	0.61	52	0.35
CHAINWOO	588	2.74	727	3.40	468	2.70
DIXON3DQ	9258	16.29	124422	239.85	10715	30.73
DQDRTIC	17	0.03	17	0.03	22	0.44
DQRTIC	72	0.06	72	0.06	47	0.06
EIGENALS	37288	656.73	24102	423.89	9938	180.51
EXTROSNB	2936	0.02	5659	0.02	1076	0.01
FLETCHBV	4110	22.06	17501	97.75	10102	65.34
FLETCHCR	9773	3.26	6914	2.26	5685	2.24
FMINSURF	2427	23.04	1570	14.99	1192	13.32
GENHUMPS	9885	33.08	13145	44.26	11001	40.29
GENROSE	2370	0.37	1982	0.32	1234	0.24
HILBERTA	14	0.01	14	0.01	31	0.01
LIARWD	106	0.38	71	0.25	28	0.13
MANCINO	26	0.32	26	0.32	15	0.19
MOREBV	69	0.11	90	0.14	45	0.10
NONCVXU2	6213	39.07	13430	66.88	4013	23.93
NONCVXUN	9885	47.27	51246	256.36	12879	76.24
NONDIA	111	0.48	30	0.14	23	0.13
NONDQUAR	915	1.63	1838	3.35	558	1.62
POWELLSG	680	1.25	340	0.65	73	0.22
POWER	902	1.29	720	1.06	451	1.14
QUARTC	94	0.15	93	0.14	48	0.13
SCHMVETT	75	0.72	74	0.71	48	0.52
SENSORS	61	0.50	50	0.41	23	0.19
SPARSINE	15737	8.73	20808	11.49	8376	5.36
SPMSRTLS	340	1.76	334	1.72	213	1.35
SROSENBR	37	0.07	29	0.06	20	0.06
TOINTGSS	20	0.09	10	0.05	26	0.14
TQUARTIC	95	0.24	69	0.18	27	0.10
TRIDIA	7514	13.26	5144	9.96	3150	9.00
VAREIGVL	449	1.89	547	2.35	22	0.10
WOODS	260	0.68	216	0.58	121	0.42

The efficiency profiles for conjugate gradient algorithms are shown in Figs. 7.24–7.26. They clearly indicate that the code CG-DESCENT is superior, in terms of the CPU time when compared to the other discussed conjugate gradient algorithms. However it is comparable to the other algorithms as far as the number of function evaluations is concerned. It seems that the Hager–Zhang line search rules, which take care of numerical stability, does work on the chosen set of problems. However, we have to mention that the step-size selection algorithm used in CG-DESCENT is different from that implemented in Moré–Thuente code which is incorporated in Algorithm 7.1, and from the procedure applied in CG+ code.

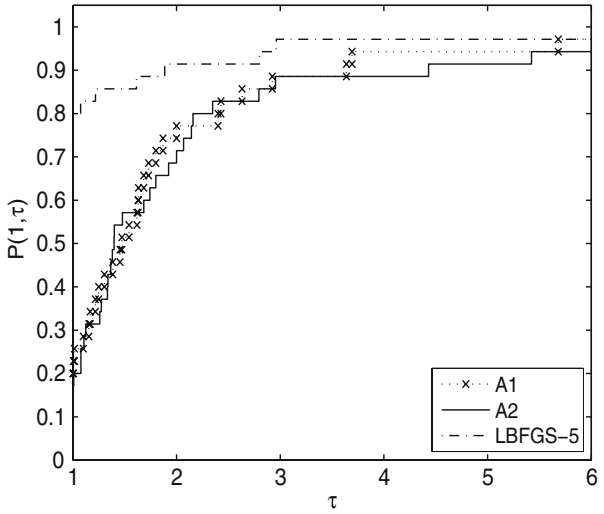

Fig. 7.21 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm, A2–weak Algorithm and L-BFGS

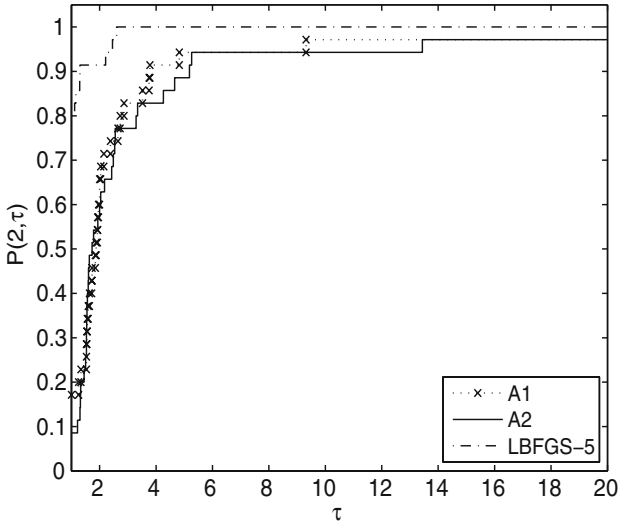

Fig. 7.22 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, A2–weak Algorithm and L-BFGS

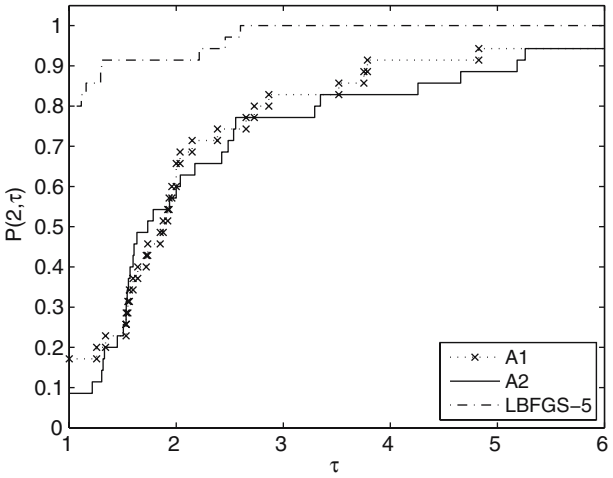

Fig. 7.23 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, A2–weak Algorithm and L-BFGS

On Figs. 7.27–7.29 we also show how the code CG-DESCENT compares to L-BFGS code. These results are in line with those presented in [91] although we use the different stopping criterion and our set of problems is the subset of that in [91]–CG-DESCENT is superior to L-BFGS as far as CPU time is concerned but lags behind it when the number of function evaluations is taken into account.

We also did experiments with CG-DESCENT code in which the Wolfe conditions as line search rules are applied. According to [91] this version of CG-DESCENT code should give much worse results in comparison to the version which employs the Hager–Zhang rules. However, on our test problems the difference between these two versions of the code are negligible if we take into account both criteria: CPU time and the number of function evaluations. In order to put these results in the proper perspective, we have to admit that we used our standard stopping criterion based on the gradient Euclidean norm and Hager and Zhang relied on the gradient Chebyshev norm while reporting their results in [91].

Our results are reported in Table 7.10 and the efficiency profiles are presented in Figs. 7.30–7.31.

7.12 Notes

In this chapter we have presented a family of conjugate gradient algorithms. They originate in the algorithms by Wolfe and Lemaréchal for nondifferentiable problems. It has been shown that it is possible to construct a general algorithm that includes as a special case the Wolfe–Lemaréchal algorithm. Moreover we have

Table 7.9 Numerical comparison of CG-DESCENT and CG+ codes with the implementation of A1–weak Algorithm on problems given in Table 7.4.

Problem	CG-DESCENT			A1–weak			CGg+ method = 3		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	3	5	0.08	2	29	0.43	3	37	0.54
BROYDN7D	2874	5727	55.78	2842	4229	61.08	2791	5631	79.54
BRYBND	33	62	0.28	33	149	0.85	70	156	0.87
CHAINWOO	287	558	2.06	210	588	2.74	0	0	0.00
DIXON3DQ	10001	20001	24.38	5647	9258	16.29	10000	20006	29.76
DQDRTIC	8	15	0.02	7	17	0.03	5	15	0.02
DQRTIC	28	55	0.03	13	72	0.06	13	53	0.04
EIGENALS	6247	12477	154.23	5564	37288	656.73	29048	58102	1020.52
EXTROSNB	5268	8947	0.03	1139	2936	0.02	4492	9128	0.04
FLETCHBV	20020	40020	179.84	7624	4110	22.06	7752	15526	78.90
FLETCHCR	7433	14144	3.30	5372	9773	3.26	4328	8681	2.56
FMINSURF	925	1733	11.22	950	2427	23.04	915	1835	16.34
GENHUMPS	6738	13384	36.09	9169	9885	33.08	7408	15053	48.81
GENROSE	1303	2556	0.29	1118	2370	0.37	1117	2261	0.33
HILBERTA	10	15	0.01	6	14	0.01	6	14	0.01
LIARWD	41	60	0.20	14	106	0.38	14	44	0.16
MANCINO	11	21	0.24	9	26	0.32	10	25	0.31
MOREBV	58	113	0.13	53	69	0.11	60	121	0.17
NONCVXU2	5247	10493	38.07	4032	6213	39.07	5100	10210	46.71
NONCVXUN	30205	60409	218.30	8351	9885	47.27	30274	60560	276.19
NONDIA	16	22	0.10	8	111	0.48	6	26	0.11
NONDQUAR	1201	2356	3.19	394	915	1.63	1445	2963	4.82
POWELLSG	319	540	0.84	60	680	1.25	409	972	1.65
POWER	385	769	0.73	464	902	1.29	380	769	0.88
QUARTC	29	57	0.08	14	94	0.15	13	58	0.08
SCHMVETT	36	56	0.51	23	75	0.72	34	76	0.72
SENSORS	40	55	0.46	23	61	0.50	25	50	0.48
SPARSINE	6289	12577	5.02	5377	15737	8.73	4412	8829	4.66
spmsrtls	199	387	1.44	185	340	1.76	186	378	1.85
SROSENBR	17	26	0.05	7	37	0.07	8	26	0.05
TOINTGSS	4	7	0.03	6	20	0.09	4	20	0.09
TQUARTIC	60	75	0.18	9	95	0.24	9	29	0.07
TRIDIA	1158	2315	2.86	3702	7514	13.26	1155	2313	3.42
VAREIGVL	306	611	1.78	297	449	1.89	305	620	2.55
WOODS	257	426	0.93	55	260	0.68	231	486	1.21
Total	107056	211074	742.78	62779	126734	939.94	112028	225073	1624.46

demonstrated that the Wolfe–Lemaréchal algorithm can be regarded as a version of the Fletcher–Reeves conjugate gradient algorithm.

We have derived a conjugate gradient algorithm which is equivalent to the Polak–Ribière algorithm if a directional minimization is exact. This algorithm is globally convergent under weak conditions on the directional minimization.

It would be interesting to explore the ideas presented in this chapter in the context of nondifferentiable optimization. Some preliminary results presented in [173] are encouraging.

The chapter gives several comparisons of versions of the method of shortest residuals to other conjugate gradient algorithms. We conclude it by Tables 7.11 and 7.12 which illustrate the influence of used storage (and the preconditioning in the case of the limited memory quasi-Newton method) on the performance of conjugate gradient algorithms – Figs. 7.32–7.36 address the appropriate efficiency profiles. The code L-BFGS is clearly superior to both CONMIN and CG+, however CONMIN performance is close to that of L-BFGS.

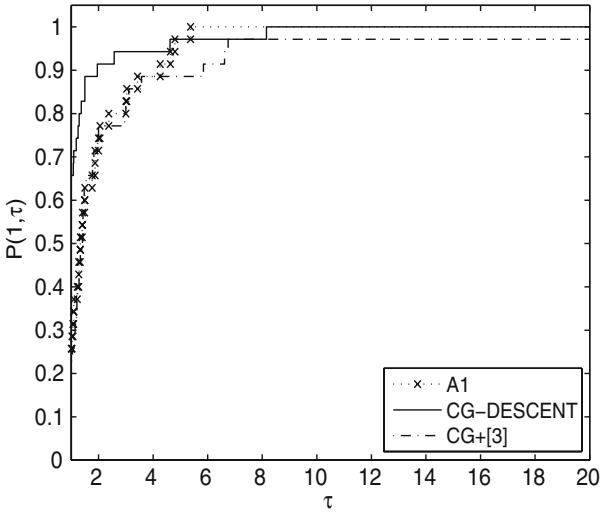

Fig. 7.24 Cumulative distribution ratio of CPU time: comparison of the proposed A1–weak Algorithm, CG-DESCENT and CG+ (with method = 3) codes

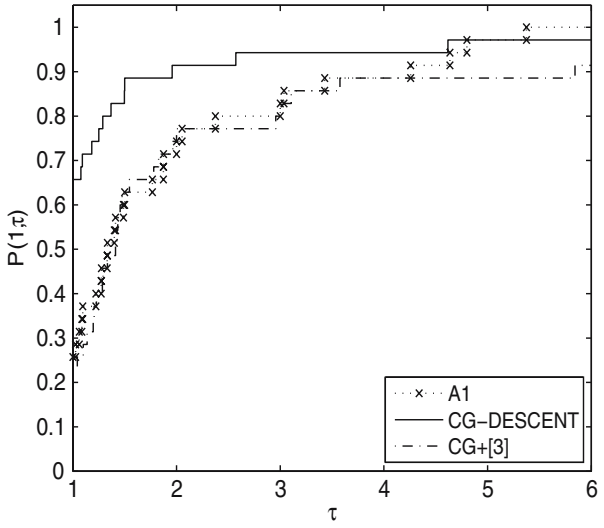

Fig. 7.25 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm, CG-DESCENT and CG+ (with method = 3) codes

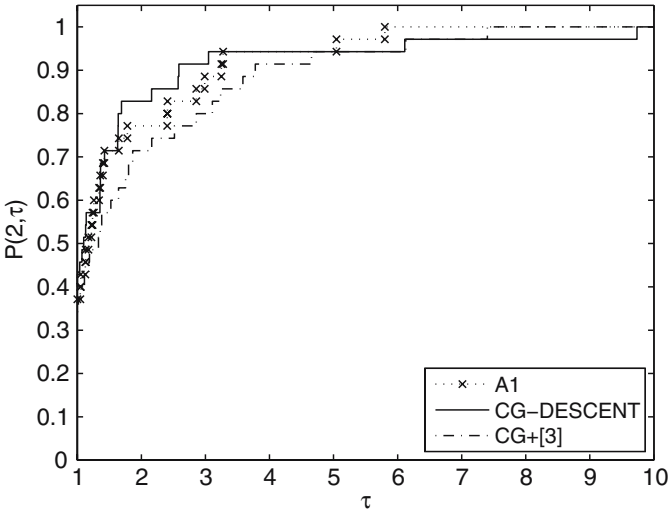

Fig. 7.26 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, CG-DESCENT and CG+ (with method = 3) codes

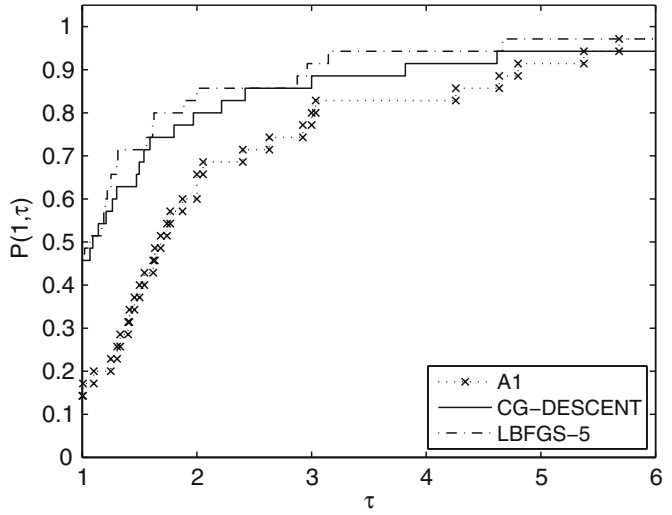

Fig. 7.27 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm, CG-DESCENT and L-BFGS (with $m = 5$) codes

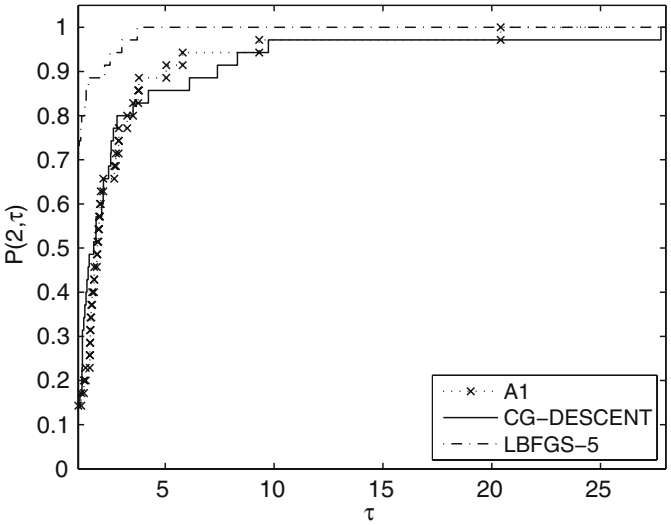

Fig. 7.28 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, CG-DESCENT and L-BFGS (with $m = 5$) codes

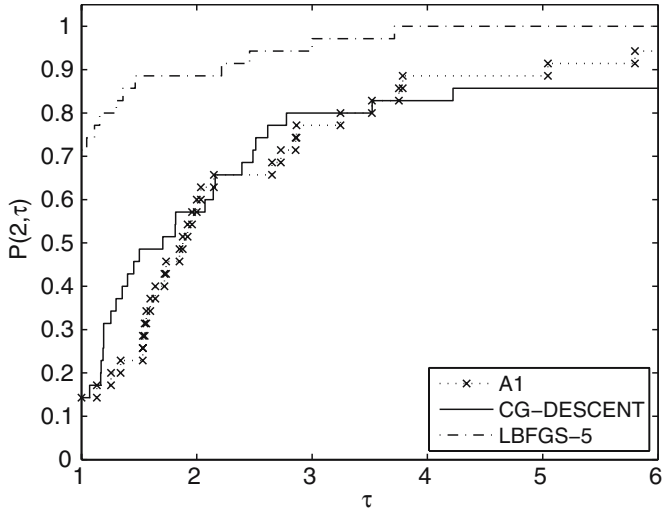

Fig. 7.29 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, CG-DESCENT and L-BFGS (with $m = 5$) codes

Table 7.10 Numerical comparison of CG-DESCENT and CG-DESCENT-W (with the Wolfe conditions) codes with the implementation of A1–weak Algorithm on problems given in Table 7.4

Problem	CG-DESCENT-W			CG-DESCENT			A1–weak Algorithm		
	IT	IF	CPU	IT	IF	CPU	IT	IF	CPU
BROWNAL	3	5	0.08	3	5	0.08	2	29	0.43
BROYDN7D	2908	5795	56.60	2874	5727	55.78	2842	4229	61.08
BRYBND	33	62	0.28	33	62	0.28	33	149	0.85
CHAINWOO	290	575	2.10	287	558	2.06	0	0	0.00
DIXON3DQ	10003	20003	24.41	10001	20001	24.38	5647	9258	16.29
DQDRTIC	8	15	0.02	8	18	0.02	7	17	0.03
DQRTIC	28	55	0.03	28	55	0.03	13	72	0.06
EIGENALS	7727	15439	190.48	2561	5109	63.30	5564	37288	656.73
EXTROSNB	5274	9088	0.04	16	25	0.01	1139	2936	0.02
FLETCHBV	20020	40020	179.84	20020	40020	179.84	7624	4110	22.06
FLETCHCR	7121	13733	3.20	1141	2121	0.50	5372	9773	3.26
FMINSURF	949	1742	11.43	925	1733	11.22	950	2427	23.04
GENHUMPS	9158	18134	48.14	5077	10062	27.80	9169	9885	33.08
GENROSE	1282	2530	0.78	525	1024	0.12	1118	2370	0.37
HILBERTA	17	29	0.01	10	15	0.01	6	14	0.01
LIARWD	42	62	0.20	41	60	0.20	14	106	0.38
MANCINO	11	21	0.24	11	21	0.24	9	26	0.32
MOREBV	58	113	0.13	58	113	0.13	53	69	0.11
NONCVXU2	5247	10493	38.07	5247	10493	38.07	4032	6213	39.07
NONCVXUN	30205	60409	218.47	10001	20001	72.48	8351	9885	47.27
NONDIA	16	22	0.10	16	22	0.10	8	111	0.48
NONDQUAR	1221	2382	3.25	1201	2356	3.19	394	915	1.63
POWELLSG	119	220	0.34	319	540	0.84	60	680	1.25
POWER	385	769	0.73	385	769	0.73	464	902	1.29
QUARTC	29	57	0.08	29	57	0.08	14	94	0.15
SCHMVETT	36	56	0.51	36	56	0.51	23	75	0.72
SENSORS	40	55	0.46	40	55	0.46	23	61	0.50
SPARSINE	6289	12577	5.03	1001	2001	0.79	5377	15737	8.73
SPMSRTLS	199	387	1.44	199	387	1.44	185	340	1.76
SROSENBR	17	26	0.05	17	26	0.05	7	37	0.07
TOINTGSS	4	7	0.03	4	7	0.03	6	20	0.09
TQUARTIC	43	58	0.13	60	75	0.18	9	95	0.24
TRIDIA	1158	2315	2.86	1158	2315	2.86	3702	7514	13.26
VAREIGVL	306	611	1.78	306	611	1.78	297	449	1.89
WOODS	329	554	1.22	257	426	0.93	55	260	0.68
Total	110575	218419	792.56	63895	126926	490.52	62569	126146	937.20

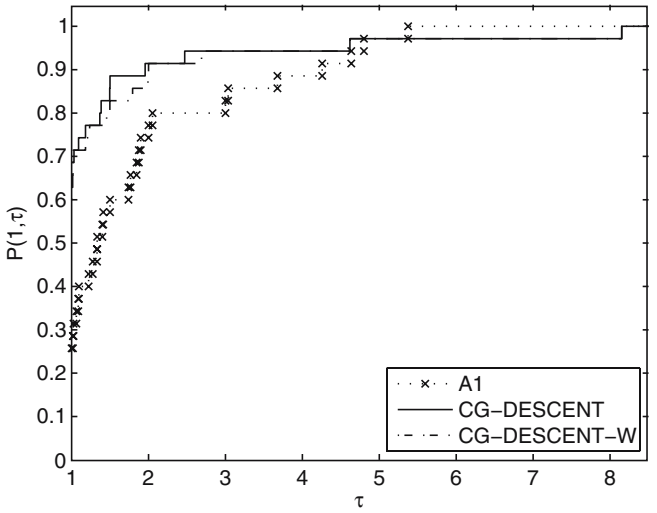

Fig. 7.30 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm, CG-DESCENT and CG-DESCENT-W codes

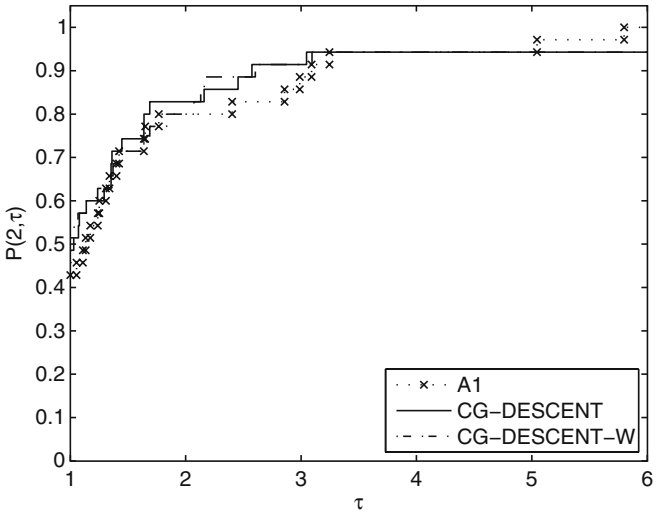

Fig. 7.31 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm, CG-DESCENT and CG-DESCENT-W codes

Table 7.11 Numerical results: comparison of CONMIN to CG+ and L-BFGS on problems from Table 7.4

Problem	CONMIN		CG+ method = 3		L-BFGS $m = 5$	
	IF	CPU	IF	CPU	IF	CPU
BROWNAL	4	0.08	37	0.54	15	0.23
BROYDN7D	5725	87.01	5631	79.54	3158	48.92
BRYBND	66	0.45	156	0.87	52	0.35
CHAINWOO	426	2.53	*F*	*F*	468	2.70
DIXON3DQ	11441	28.51	20006	29.76	10715	30.73
DQDRTIC	15	0.04	15	0.02	22	0.44
DQRTIC	27	0.03	53	0.04	47	0.06
EIGENALS	11190	197.83	58102	1020.52	9938	180.51
EXTROSNB	2318	0.02	9128	0.04	1076	0.01
FLETCHBV	15484	100.92	15526	78.90	10102	65.34
FLETCHCR	10854	4.56	8681	2.56	5685	2.24
FMINSURF	1926	21.25	1835	16.34	1192	13.32
GENHUMPS	18580	69.72	15053	48.81	11001	40.29
GENROSE	2263	0.48	2261	0.33	1234	0.24
HILBERTA	13	0.01	14	0.01	31	0.01
LIARWD	30	0.14	44	0.16	28	0.13
MANCINO	19	0.24	25	0.31	15	0.19
MOREBV	108	0.23	121	0.17	45	0.10
NONCVXU2	8153	49.22	10210	46.71	4013	23.93
NONCVXUN	16909	101.31	60560	276.19	12879	76.24
NONDIA	19	0.19	26	0.11	23	0.13
NONDQUAR	802	2.36	2963	4.82	558	1.62
POWELLSG	121	0.34	972	1.65	73	0.22
POWER	937	2.40	769	0.88	451	1.14
QUARTC	29	0.07	58	0.08	48	0.13
SCHMVETT	50	0.71	76	0.72	48	0.52
SENSORS	50	0.41	50	0.48	23	0.19
SPARSINE	10833	7.39	8829	4.66	8376	5.36
SPMSRTLS	373	2.38	378	1.85	213	1.35
SROSENBR	15	0.04	26	0.05	20	0.06
TOINTGSS	13	0.07	20	0.09	26	0.14
TQUARTIC	30	0.09	29	0.07	27	0.10
TRIDIA	7497	22.02	2313	3.42	3150	9.00
VAREIGVL	599	2.88	620	2.55	22	0.10
WOODS	116	0.41	485	1.21	121	0.42

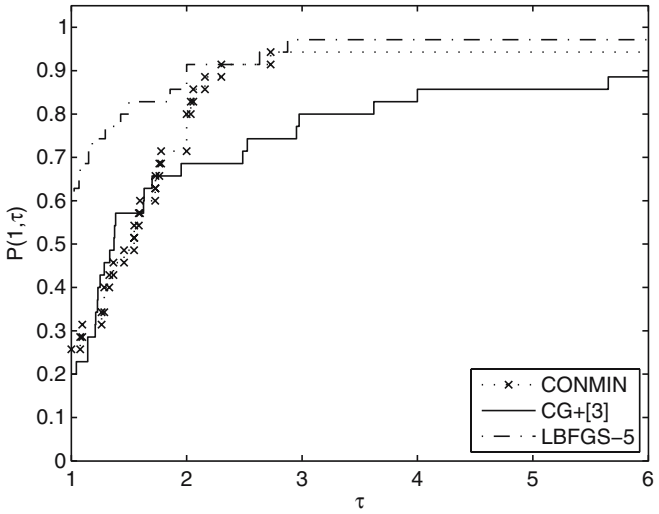

Fig. 7.32 Cumulative distribution ratio of CPU time: comparison of CONMIN, CG+ (method = 3) and L-BFGS with $m = 5$

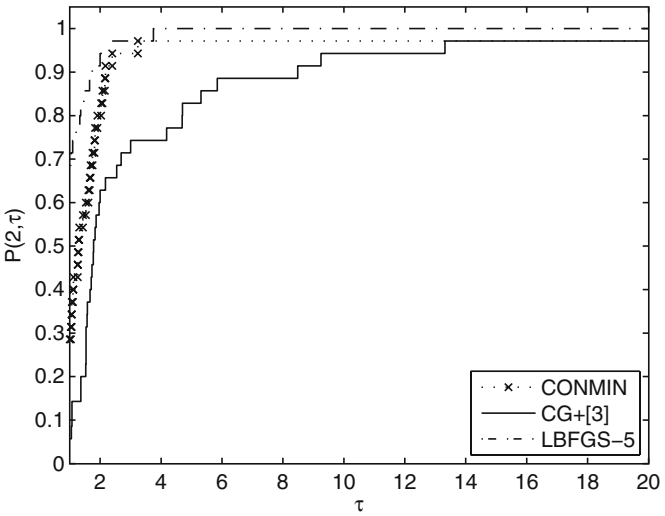

Fig. 7.33 Cumulative distribution ratio of the number of function evaluations: comparison of CONMIN, CG+ (method = 3) and L-BFGS with $m = 5$

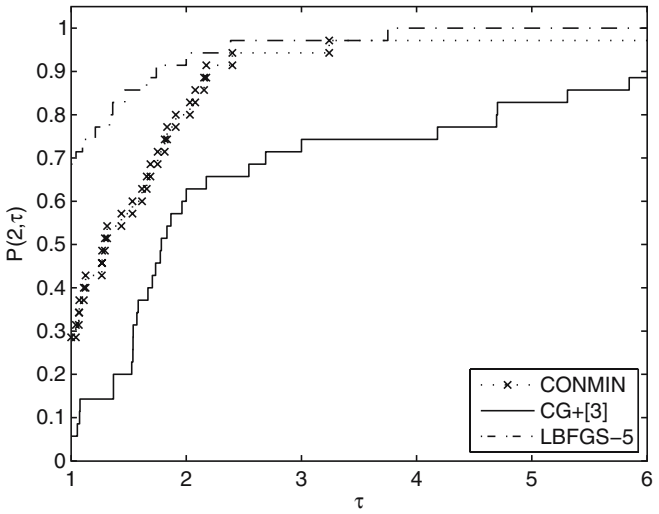

Fig. 7.34 Cumulative distribution ratio of the number of function evaluations: comparison of CONMIN, CG+ (method = 3) and L-BFGS with $m = 5$

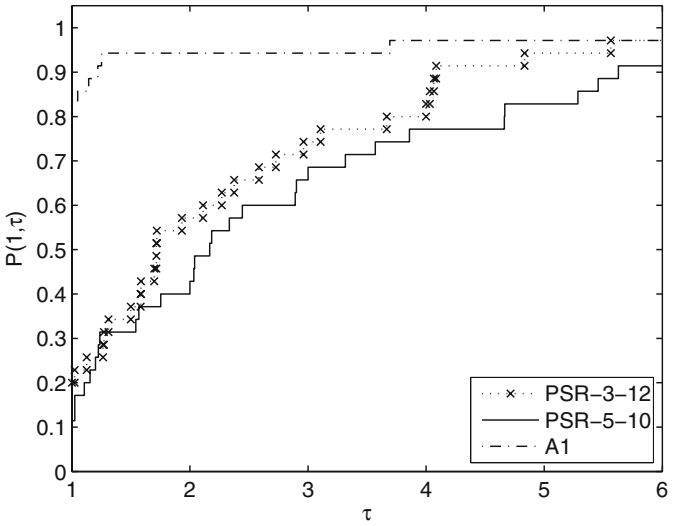

Fig. 7.35 Cumulative distribution ratio of CPU time: comparison of A1–weak Algorithm and Algorithm 8.3 denoted as PSR (cf. Table 7.12)

Table 7.12 Numerical results: comparison of Algorithm 7.1 and Algorithm 8.3 (denoted as PSR) on problems from Table 7.4

Problem	PSR-3-12		A1–weak Algorithm		PSR-5-10	
	IF	CPU	IF	CPU	IF	CPU
BROWNAL	27	0.41	29	0.43	27	0.42
BROYDN7D	3784	80.03	4229	61.08	3965	95.73
BRYBND	362	4.73	149	0.85	69	1.02
CHAINWOO	398	4.7	588	2.74	373	5.94
DIXON3DQ	5870	50.61	9258	16.29	12207	156.99
DQDRTIC	31	0.11	17	0.03	29	0.14
DQRTIC	40	0.29	72	0.06	40	0.48
EIGENALS	27870	537.43	37288	656.73	25845	524.93
EXTROSNB	1519	0.03	2936	0.02	1364	0.04
FLETCHBV	5134	65.35	4110	22.06	7128	120.39
FLETCHCR	8060	7.74	9773	3.26	7433	9.46
FMINSURF	1289	25.95	2427	23.04	1078	28.5
GENHUMPS	8675	56.78	9885	33.08	8822	72.23
GENROSE	1714	0.84	2370	0.37	1616	1.07
HILBERTA	82	0.01	14	0.01	23	0.01
LIARWD	43	0.38	106	0.38	33	0.42
MANCINO	22	0.28	26	0.32	22	0.28
MOREBV	64	0.3	69	0.11	58	0.33
NONCVXU2	4103	49.71	6213	39.07	4957	79.42
NONCVXUN	6182	74.94	9885	47.27	9851	156.76
NONDIA	16	0.13	111	0.48	16	0.15
NONDQUAR	756	6.66	915	1.63	510	6.29
POWELLSG	253	2.15	680	1.25	136	1.53
POWER	600	5.2	902	1.29	570	7.26
QUARTC	41	0.61	94	0.15	41	1
SCHMVETT	64	0.91	75	0.72	63	1.11
SENSORS	50	0.41	61	0.5	51	0.42
SPARSINE	11599	14.82	15737	8.73	8927	15.3
SPMSRTLS	286	3.4	340	1.76	236	3.59
SROSENBR	25	0.28	37	0.07	22	0.37
TOINTGSS	22	0.19	20	0.09	24	0.22
TQUARTIC	39	0.38	95	0.24	36	0.56
TRIDIA	3855	34.27	7514	13.26	3655	47.33
VAREIGVL	30	0.22	449	1.89	23	0.22
WOODS	454	4.16	260	0.68	227	3.17
Total	93359	1034.41	126734	939.94	99477	1343.08

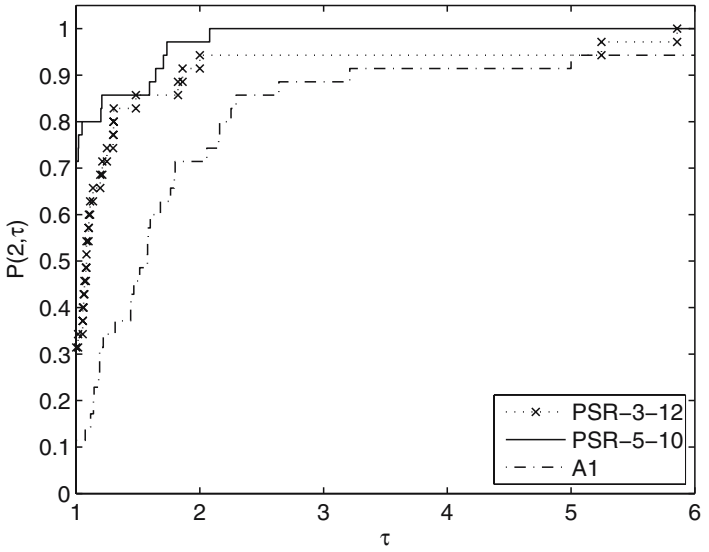

Fig. 7.36 Cumulative distribution ratio of the number of function evaluations: comparison of A1–weak Algorithm and Algorithm 8.3 denoted as PSR (cf. Table 7.12)

Table 7.12 concentrates on the performance of Algorithm 7.1 and Algorithm 8.3 (discussed in the next chapter) which is a preconditioned version of Algorithm 7.1. Figures 7.35–7.36 show the corresponding efficiency profiles – Algorithm 8.3 is denoted as PSR-m-k, the first number in the description corresponds to the number of pairs used in BFGS matrices while the second number says for how many iterations these preconditioned matrices are applied. Since Algorithm 8.3 uses more storage not surprisingly that it is superior to the method of shortest residuals.

Chapter 8
The Preconditioned Shortest Residuals Algorithm

8.1 Introduction

Due to strong convergence properties exhibited by the method of shortest residuals it is tempting to extend it by introducing its preconditioned version. The idea behind preconditioned conjugate gradient algorithm is to transform the decision vector by linear transformation D such that after the transformation the nonlinear problem is *easier* to solve – eigenvalues of Hessian matrices of the objective function of the new optimization problem are more clustered (see Chap. 1 for the discussion of how eigenvalues clustering influences the behavior of conjugate gradient algorithms).

If $\hat{x}$ is transformed x (or x is transformed $\hat{x}$ as in Chap. 4):

$$\hat{x} = Dx$$

then our minimization problem will become

$$\min_{\hat{x}} \left[\hat{f}(\hat{x}) = f(D^{-1}\hat{x})\right]$$

and for this problem the search direction will be defined as follows

$$\hat{d}_k = -\mathbf{Nr}\{\hat{g}(\hat{x}_k), -\hat{\beta}_k \hat{d}_{k-1}\} \tag{8.1}$$

Since we want to avoid minimizing $\hat{f}$ with respect to $\hat{x}$ we need expressing the above search direction rule in terms of f and x. First of all, notice that

$$\hat{g}(\hat{x}) = D^{-T} g(x)$$

therefore we can write

$$\hat{d}_k = -\mathbf{Nr}\{D^{-T} g(D^{-1} x_k), -\hat{\beta}_k \hat{d}_{k-1}\}. \tag{8.2}$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

If we multiply both sides of (8.2) by D^{-1} we will get

$$d_k = -\lambda_k D^{-1} D^{-T} g_k + (1 - \lambda_k)\,\hat{\beta}_k d_{k-1}, \tag{8.3}$$

where $0 \leqslant \lambda_k \leqslant 1$ and either

$$\hat{\beta}_k = 1$$

for the Fletcher–Reeves version, or

$$\begin{aligned}\hat{\beta}_k &= \frac{\|\hat{g}_k\|^2}{|\,(\hat{g}_k - \hat{g}_{k-1})^T\,\hat{g}_k|} \\ &= \frac{g_k^T D^{-1} D^{-T} g_k}{|\,(g_k - g_{k-1})^T D^{-1} D^{-T} g_k|}\end{aligned}$$

for the Polak–Ribière version.

The (8.3) can be stated as

$$d_k = -\lambda_k H g_k + (1 - \lambda_k)\,\hat{\beta}_k d_{k-1}, \tag{8.4}$$

where $H = D^{-1} D^{-T}$. This suggests that D should be chosen in such a way that $D^T D$ is an approximation to $\nabla^2 f(\bar{x})$ where $\bar{x}$ is a solution of problem (7.1).

Moreover, D should be such that systems of linear equations

$$D^T \hat{g}_k = g_k \tag{8.5}$$

$$D d_k = \hat{d}_k, \tag{8.6}$$

which we have to solve at every iteration are *easy* to solve.

It is straightforward to show that for rule (8.1) properties similar to (7.14)–(7.15) hold:

$$g_k^T d_k \leqslant -\|\hat{d}_k\|^2 \tag{8.7}$$

$$-\hat{\beta}_k d_{k-1}^T H^{-1} d_k \leqslant -\|\hat{d}_k\|^2 \tag{8.8}$$

and

$$g_k^T d_k = -\|\hat{d}_k\|^2,\ \hat{\beta}_k d_{k-1}^T H^{-1} d_k = \|\hat{d}_k\|^2,$$

if $0 < \lambda_k < 1$.

The aim of the chapter is to present two versions of the preconditioned conjugate gradient algorithm. They differ by the way the scaling matrices are built and then used in conjugate gradient iterations. One version follows the strategy proposed by Buckley and LeNir, the other can be regarded as the extension of the limited memory quasi-Newton method. We discuss convergence properties of these methods and provide some numerical results which show that the proposed approach can lead to viable and competitive numerical algorithms.

8.2 General Preconditioned Method of Shortest Residuals

The scaling matrix D should be changed every fixed number of iterations to guarantee that it is as close as possible to $\nabla^2 f(x_k)$. For the simplicity of presentation we assume that it is changed at every iteration. Since $\hat{d}_k$ is calculated in the space determined by D_k, and $\hat{d}_k$ is expressed by $\hat{d}_{k-1}$ we need the additional notation

$$\hat{d}^c_{k-1} = D_k d_{k-1}. \tag{8.9}$$

Thus, our general algorithm is as follows.

Algorithm 8.1. (The preconditioned method of shortest residuals)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\varepsilon > 0$, $\{\hat{\beta}_k\}$, $\{D_k\}$, $D_k \in \mathscr{R}^{n\times n}$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, set

$$d_1 = -g_1$$

and $k = 1$.

2. Find a positive number α_k such that the following conditions are fulfilled:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu\alpha_k \|\hat{d}_k\|^2 \tag{8.10}$$

$$g(x_k + \alpha_k d_k)^T d_k \geqslant -\eta \|\hat{d}_k\|^2. \tag{8.11}$$

Substitute $x_k + \alpha_k d_k$ for x_{k+1}.

3. If $\|g_{k+1}\| = 0$ then STOP, otherwise evaluate d_{k+1} by solving

$$D^T_{k+1}\hat{g}_{k+1} = g_{k+1} \tag{8.12}$$

$$\hat{d}^c_k = D_{k+1} d_k \tag{8.13}$$

$$\hat{d}_{k+1} = -\mathbf{Nr}\{\hat{g}_{k+1}, -\hat{\beta}_{k+1}\hat{d}^c_k\} \tag{8.14}$$

$$D_{k+1} d_{k+1} = \hat{d}_{k+1}. \tag{8.15}$$

Increase k by one and go to Step 2.

The directional minimization is defined by the expressions (8.10)–(8.11). These rules, which lead to inexact minimization, were taken from the algorithms for nondifferentiable problems (cf. Chap. 6). Let us observe that our conditions for directional minimization are very similar to those well-known in the literature. In order to notice that we have to replace $-\|\hat{d}_k\|^2$ in (8.10) by $g_k^T d_k$ which holds when $0 < \lambda_k < 1$. Indeed, we have

$$\begin{aligned} g_k^T d_k &= \left(D_k^T \hat{g}_k\right)^T D_k^{-1} \hat{d}_k \\ &= -\|\hat{d}_k\|^2. \end{aligned} \tag{8.16}$$

Furthermore, if we do not impose the restriction on λ $(0 \leqslant \lambda \leqslant 1)$ we could employ the Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k$$
$$g(x_k + \alpha_k d_k)^T d_k \geqslant \eta g_k^T d_k.$$

This version of a preconditioned conjugate gradient algorithm would be in the spirit of the method of shortest residuals discussed in Sect. 7.6.

A procedure which finds α_k satisfying (8.10)–(8.11), in the finite number of operations, can be easily constructed.

Lemma 8.1. *There exists a procedure which finds α_k satisfying (8.10)–(8.11) in a finite number of operations, or produces $\alpha_k \to \infty$ such that $f(x_k + \alpha_k d_k) \to -\infty$.*

Proof. The proof of this lemma can be carried out along the lines outlined in the proof of Lemma 2.4, if we notice (8.16) and

$$g(x_k + \alpha_k d_k)^T d_k = \hat{g}_{k+1}^T \hat{d}_k, \tag{8.17}$$

where in (8.17) $\hat{g}_{k+1}$ is defined through the matrix D_k. □

The following lemma plays crucial role in proving global convergence of Algorithm 8.1.

Lemma 8.2. *Suppose that*

(i) There exists $L < \infty$ such that

$$\|g(y) - g(x)\| \leqslant L\|y - x\|$$

for all x, y from a bounded set,

(ii) There exist d_l, d_u such that $0 < d_l < d_u < \infty$ and

$$d_l \|z\|^2 \leqslant z^T D_k^T D_k z \leqslant d_u \|z\|^2 \tag{8.18}$$

for all $z \in \mathscr{R}^n$ and k.

If the direction d_k is determined by (8.14), the step-size coefficient α_k satisfies (8.10)–(8.11), then

$$\lim_{k \to \infty} \|d_k\| = 0 \tag{8.19}$$

or

$$\lim_{k \to \infty} f(x_k) = -\infty. \tag{8.20}$$

Proof. Assume that $\{f(x_k)\}$ is bounded. In the first step of the proof we will show that

$$\lim_{k \to \infty} \|\hat{d}_k\|^2 = 0. \tag{8.21}$$

Equation (8.21) follows from the following considerations. We have

$$g_{k+1}^T d_k \geqslant -\eta \|\hat{d}_k\|^2,$$

thus

$$\begin{aligned}(g_{k+1} - g_k)^T d_k &\geqslant -\eta \|\hat{d}_k\|^2 - g_k^T d_k \\ &\geqslant (1-\eta)\|\hat{d}_k\|^2 \end{aligned} \tag{8.22}$$

Since the assumption *(i)* is satisfied

$$(g_{k+1} - g_k)^T d_k \leqslant \alpha_k \hat{L} \|\hat{d}_k\|^2$$

for some $\hat{L} < \infty$, which together with (8.22) imply that

$$\alpha_k \geqslant \frac{(1-\eta)}{L}.$$

From directional minimization rule we also have

$$\begin{aligned} f(x_k + \alpha_k d_k) - f(x_k) &\geqslant \mu \alpha_k \|\hat{d}_k\|^2 \\ &\geqslant \mu \frac{(1-\eta)}{L} \|\hat{d}_k\|^2 \end{aligned}$$

which implies

$$L[f(x_k + \alpha_k d_k) - f(x_k)] / [(1-\eta)\mu] \geqslant \|\hat{d}_k\|^2.$$

Eventually we will get

$$\infty > L \sum_{k=0}^{\infty} [f(x_k + \alpha_k d_k) - f(x_k)] / [(1-\eta)\mu] \geqslant \sum_{k=0}^{\infty} \|\hat{d}_k\|^2$$

since f is bounded. This proves

$$\sum_{k=0}^{\infty} \|\hat{d}_k\|^2 < \infty \tag{8.23}$$

and (8.21).

From (8.23) we also have

$$\lim_{k \to \infty} \|\hat{d}_k\| = 0. \tag{8.24}$$

Due to the assumption *(ii)* we can show that the following estimates are valid:

$$\sigma_n(D_k) \geqslant \sqrt{d_l}, \quad \sigma_1(D_k) \leqslant \sqrt{d_u}$$

where $\sigma_1(D_k)$ and $\sigma_n(D_k)$ are the largest and the smallest singular values of the matrix D_k in its SVD decomposition (cf. Appendix B).

We also have

$$\|D_k^{-1}\|_2 = \frac{1}{\sigma_n(D_k)},$$

where $\|D_k^{-1}\|_2$ is the spectral norm of D_k^{-1}. This implies

$$\|d_k\| \leqslant \|D_k^{-1}\|_2 \|\hat{d}_k\| \leqslant \frac{1}{\sqrt{d_l}} \|\hat{d}_k\|. \tag{8.25}$$

Equation (8.19) follows from (8.24) and (8.25). □

In order to go from (8.19) to the condition

$$\lim_{k\to\infty} \|g_k\| = 0$$

we have to pay special attention to the situation when the vectors d_{k-1} are not appropriately scaled by the $\hat{\beta}_k$, and when for the angle between $-g_k$ and d_k–φ_k we have $\lim_{k\in K, k\to\infty} \varphi_k = \pi$ for certain sequence $\{g_k\}_{k\in K}$ and inexact line search.

In each of these situations we shall have (8.19). Thus, in order to prove the convergence of Algorithm 8.1 we have to exclude these situations.

Theorem 8.1. *Suppose that assumptions* (i)–(ii) *of* Lemma 8.2 *are satisfied and that* $\{\hat{\beta}_k\}$ *is such that*

$$\liminf_{k\to\infty} \left(\hat{\beta}_k \|d_{k-1}\| \right) \geqslant \nu_1 \liminf_{k\to\infty} \|g_k\| \tag{8.26}$$

where ν_1 *is some positive constant. If there exists a number* ν_2 *such that* $\nu_2 \|D_k^{-1}\|_2 \|D_k\|_2 \in (0,1)$ *and*

$$g_k^T d_{k-1} \leqslant \nu_2 \|g_k\| \|d_{k-1}\|, \text{ whenever } \lambda_k \in (0,1), \tag{8.27}$$

then $\lim_{k\to\infty} f(x_k) = -\infty$, *or every cluster point* $\bar{x}$ *of the sequence* $\{x_k\}$ *generated by* Algorithm 8.1 *is such that* $g(\bar{x}) = 0$.

Proof. Case (a). Let us suppose that for infinitely often $k \in K_1$, $\lambda_k \in (0,1)$, thus:

$$\hat{g}_k^T \hat{d}_k = -\|\hat{d}_k\|^2 \text{ and } \hat{\beta}_k \left(\hat{d}_{k-1}^c\right)^T \hat{d}_k = \|\hat{d}_k\|^2. \tag{8.28}$$

Furthermore, assume that $x_k \to_{k\to\infty, k\in K_1} \bar{x}$, $g(\bar{x}) \neq 0$. From this it follows that $\lim_{k\to\infty, k\in K_1} \|g_k\| \neq 0$. Because of this, the equalities (8.28), and since by Lemma 8.2 $\lim_{k\to\infty} \|\hat{d}_k\| = 0$, for the angle φ_k between $-\hat{g}_k$ and $\hat{d}_k$, and for the angle δ_k between $\hat{d}_{k-1}^c$ and $\hat{d}_k$ we have

$$\lim_{k\to\infty, k\in K_1} \cos\varphi_k = \lim_{k\to\infty, k\in K_1} \left\langle -\frac{\hat{g}_k}{\|\hat{g}_k\|}, \frac{\hat{d}_k}{\|\hat{d}_k\|} \right\rangle$$

$$= \lim_{k\to\infty, k\in K_1} \frac{\|\hat{d}_k\|^2}{\|\hat{g}_k\|} = 0 \tag{8.29}$$

$$\lim_{k\to\infty, k\in K_1} \cos\delta_k = \lim_{k\to\infty, k\in K_1} \frac{\|\hat{d}_k\|}{\|\hat{d}^c_{k-1}\|\hat{\beta}_k} = 0 \tag{8.30}$$

which follows from the fact that

$$\|\hat{d}^c_{k-1}\| = \|D_k d_{k-1}\| \geqslant \sqrt{d_l}\|d_{k-1}\|.$$

Since (8.29), (8.30) are satisfied: $\varphi_k \to \pi/2,\ \delta_k \to \pi/2$ and

$$\lim_{k\to\infty, k\in K_1} \cos(\varphi_k + \delta_k) = \lim_{k\to\infty, k\in K_1} \cos\varphi_k \cos\delta_k - \lim_{k\to\infty, k\in K_1} \sin\varphi_k \sin\delta_k = -1$$

(cf. Fig. 8.1), but this implies that

$$\lim_{k\to\infty, k\in K_1} \left\langle \frac{\hat{g}_k}{\|\hat{g}_k\|}, \frac{\hat{d}^c_{k-1}}{\|\hat{d}^c_{k-1}\|} \right\rangle = 1.$$

This contradicts our assumption (8.27) since

$$\begin{aligned} \hat{g}_k^T \hat{d}^c_{k-1} &= g_k^T d_{k-1} \\ &\leqslant \nu_2 \|g_k\| \|d_{k-1}\| \\ &= \nu_2 \|D_k^T \hat{g}_k\| \|D_k^{-1} \hat{d}^c_{k-1}\| \\ &\leqslant \nu_2 \|D_k\|_2 \|D_k^{-1}\|_2 \|\hat{g}_k\| \|\hat{d}^c_{k-1}\| \\ &< \|\hat{g}_k\| \|\hat{d}^c_{k-1}\| \end{aligned}$$

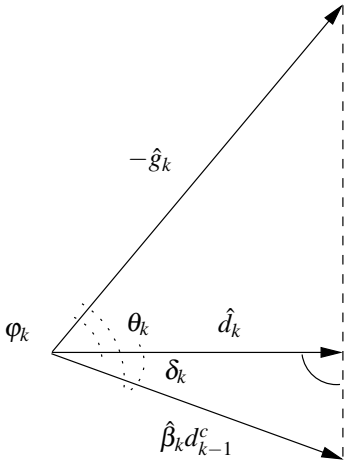

Fig. 8.1 Calculation of $\hat{d}_k$

since $\nu_2\|D_k\|_2\|D_k^{-1}\|_2 \in (0,1)$. Hence we conclude that $g(\bar{x}) = 0$.

Case (b). Now let consider the case $\lambda_k = 0$. If it occurs infinitely often for $k \in K_2$ and $x_k \to_{k\to\infty, k\in K_2} \bar{x},\ g(\bar{x}) \neq 0$ we have that there is a number ν_4 such that

$$\liminf_{k\to\infty, k\in K_2} \|g_k\| = \nu_4 > 0 \tag{8.31}$$

and by assumption $\lambda_k = 0$, and (8.26)

$$\liminf_{k\to\infty, k\in K_2} \left(\hat{\beta}_k \|d_{k-1}\|\right) \geqslant \nu_1\nu_4 > 0. \tag{8.32}$$

But $\lim_{k\to\infty, k\in K_2} \|d_k\| = 0 = \lim_{k\to, k\in K_2} \left(\hat{\beta}_k\|d_{k-1}\|\right) \geqslant \nu_1\nu_4 > 0$ and this is impossible.

Case (c). If we have the case $\lambda_k = 1$ for $k \in K_3$ then $-d_k = g_k$ for $k \in K_3$. If $x_k \to_{k\to\infty, k\in K_3} \bar{x},\ g(\bar{x}) \neq 0$ then $\lim_{k\to\infty, k\in K_3} \|g_k\| > 0$ but this is a contradiction to $\lim_{k\to\infty} \|d_k\| = 0$. This completes our proof. □

8.3 Globally Convergent Preconditioned Conjugate Gradient Algorithm

In this section we examine Algorithm 8.1 with the sequence $\{\hat{\beta}_k\}$ defined by

$$\hat{\beta}_k = \frac{\|\hat{g}_k\|^2}{\left|\left(\hat{g}_k - \hat{g}_{k-1}^c\right)^T \hat{g}_k\right|}, \tag{8.33}$$

where

$$\hat{g}_{k-1}^c = D_k g_{k-1}. \tag{8.34}$$

The following theorem shows global convergence of Algorithm 8.1 with $\{\hat{\beta}_k\}$ defined by (8.33)–(8.34).

Theorem 8.2. *Suppose that the assumptions* (i)–(ii) *of* Lemma 8.2 *are satisfied, then* Algorithm 8.1 *gives*

$$\lim_{k\to\infty} f(x_k) = -\infty, \text{ or } \lim_{k\to\infty} \|g_k\| = 0 \tag{8.35}$$

provided that:

(1) β_k is given by (8.33),
(2) there exists $M < \infty$ such that $\alpha_k \leqslant M$, $\forall k$.

Proof. From the assumptions *(i)–(ii)* of Lemma 8.2, there exists positive L for which we have

$$\begin{aligned}\hat{\beta}_k\|d_{k-1}\| &= \frac{\|\hat{g}_k\|^2\|d_{k-1}\|}{\left|\left(\hat{g}_k-\hat{g}^c_{k-1}\right)^T\hat{g}_k\right|}\\ &\geqslant \frac{\|\hat{g}_k\|^2\|d_{k-1}\|}{\|\hat{g}_k-\hat{g}^c_{k-1}\|\|\hat{g}_k\|}\\ &\geqslant \frac{\sqrt{d_l}\|g_k\|^2\|d_{k-1}\|}{\sqrt{d_u}L\alpha_{k-1}\|g_k\|\|d_{k-1}\|}\\ &\geqslant \frac{\sqrt{d_l}\|g_k\|}{\sqrt{d_u}LM}.\end{aligned}$$

If $f(x_k)\geqslant\tilde{L}>-\infty$ then, because $\|x_k-x_{k-1}\|\leqslant M\|d_{k-1}\|$ and $\|d_k\|\to_{k\to\infty}0$ we have

$$\lim_{k\to\infty}\|x_{k+1}-x_k\|=0.$$

Suppose that for some infinite set K we have

$$\lim_{k\to\infty,k\in K}\|g_k\|=\alpha>0 \tag{8.36}$$

Thus for every $\nu\in(0,1)$ such that $\nu\|D_k\|_2\|D_k^{-1}\|_2\in(0,1)$ we have

$$g_k^T d_{k-1}>\nu\|g_k\|\|d_{k-1}\|.$$

But for sufficiently large $k\in K$, since $\|x_{k+1}-x_k\|\to_{k\to\infty}0$, we will achieve the relation (from Theorem 8.1, and (8.17))

$$\begin{aligned}0<\alpha\nu &\leqslant \lim_{k\to\infty,k\in K}\left\langle g_k,\frac{d_{k-1}}{\|d_{k-1}\|}\right\rangle\\ &= \lim_{k\to\infty,k\in K}\left\langle g_{k-1},\frac{d_{k-1}}{\|d_{k-1}\|}\right\rangle\\ &\leqslant \lim_{k\to\infty,k\in K}\left(-\frac{\|\hat{d}_{k-1}\|^2}{\|d_{k-1}\|}\right)=0.\end{aligned}$$

This is impossible, thus (8.36) cannot happen. □

Following the proof of Theorem 7.4 we can also show the global convergence of Algorithm 8.1 applied to problems with strictly convex functions.

Theorem 8.3. *Suppose that the assumption* (ii) *of* Lemma 8.2 *is satisfied, the function f is twice continuously differentiable and there exist positive m, M such that:*

$$m\|z\|^2\leqslant z^T\nabla^2 f(x)z\leqslant M\|z\|^2 \tag{8.37}$$

for all $x,\ z \in \mathscr{R}^n$ *then the sequence* $\{x_k\}$ *generated by* Algorithm 8.1 *with* β_k *calculated in accordance with (8.33) converges to the minimizer of* f.

8.4 Scaling Matrices

In the previous section we showed that for a given sequence of nonsingular matrices $\{D_k\}$, where D_k satisfies (8.18), the preconditioned conjugate gradient algorithm is globally convergent. In addition, on the matrices D_k we should impose the condition that the matrix $H_k = D_k^{-1}D_k^{-T}$ is such that H_k^{-1} is as close as possible to the Hessian $\nabla^2 f(x_k)$. Furthermore, H_k should be easily factorized as $D_k^{-1}D_k^{-T}$ where D_k is a nonsingular matrix.

We present the preconditioned conjugate gradient algorithm based on the BFGS updating formula. Suppose that B_k is an approximation of $\nabla^2 f(x_k)$ then the update of B_k to an approximation of the Hessian at the point x_{k+1} is given by the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{8.38}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.

Two strategies of using B_k matrices in a preconditioned conjugate gradient algorithm are proposed.

S1 At some iteration r we start with some diagonal matrix

$$B_r = \gamma_r I$$

and then apply formula (8.38) for the next m iterations to obtain B_{r+m} which is used for the next n_r iterations in the preconditioned conjugate gradient algorithm. This strategy corresponds to Buckley–LeNir algorithm discussed in Chap. 4, if we assume that

$$\gamma_r = \frac{y_{r-1}^T y_{r-1}}{s_{r-1}^T s_{r-1}}. \tag{8.39}$$

S2 We use the m vector pairs $\{s_i, y_i\}_{i=k-m}^{k-1}$ that satisfy $s_i^T y_i > 0$, $i = k-m, \ldots, k-1$. Then, starting with some matrix B_k^0 we apply BFGS updates with these vector pairs to B_k^0. We say then that matrix B_k is obtained by a limited memory BFGS updates. If

$$B_k^0 = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} I \tag{8.40}$$

and the matrix B_k is used for next n_r iterations unchanged we have the limited memory BFGS version of the preconditioned conjugate gradient algorithm.

The limited memory BFGS formula based on (8.38) and with B_k^0 evaluated according to (8.39) needs transformations in order to factor B_k as $D_k^T D_k$. To this end suppose that the k vector pairs $\{s_i, y_i\}_{i=k-m}^{k-1}$ satisfy $s_i^T y_i > 0$ for $i = k-m, \ldots, k-1$ and recall compact representations of quasi-Newton matrices stated in Theorem 5.4:

$$B_k = B_k^0 - \begin{bmatrix} B_k^0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_k^0 S_k & L_k \\ L_k & -D_k \end{bmatrix} \begin{bmatrix} S_k^T B_k^0 \\ Y_k \end{bmatrix}, \tag{8.41}$$

with

$$S_k = [s_{k-m} \ \ldots \ s_{k-1}] \tag{8.42}$$

$$Y_k = [y_{k-m} \ \ldots \ y_{k-1}] \tag{8.43}$$

and

$$(L_k)_{i,j} = \begin{cases} s_{k-m-1+i}^T y_{k-m-1+j} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases} \tag{8.44}$$

(cf. (5.57)).

If we assume that $B_k^0 = \gamma_k I$ and introduce matrices $M_k = [\gamma_k S_k \ Y_k]$ and

$$M_k = [\gamma_k S_k \ Y_k] \tag{8.45}$$

$$W_k = \begin{bmatrix} \gamma_k S_k^T S_k & L_k \\ L_k^T & -G_k \end{bmatrix}^{-1} \tag{8.46}$$

then (8.41) can be written as

$$B_k = \gamma_k I - M_k W_k M_k^T. \tag{8.47}$$

In order to transform the matrix B_k to the form $D_k^T D_k$ we do the QR factorization of the matrix M_k^T:

$$M_k^T = R_k Q_k \tag{8.48}$$

where $Q_k \in \mathscr{R}^{n \times n}$ is an orthogonal matrix and $R_k \in \mathscr{R}^{2m \times n}$ has zero elements except the elements constituting the left $2m \times 2m$ submatrix (cf. Appendix B). Taking into account that $Q_k^T Q_k = I$ we can write (8.47) as

$$B_k = Q_k^T \left(\gamma_k I_n - R_k^T W_k R_k \right) Q_k. \tag{8.49}$$

Notice that the matrix $R_k^T W_k R_k$ has zero elements except those lying in the upper left $2m \times 2m$ submatrix (cf. Fig. 8.2). We denote this submatrix by T_k. If we compute the

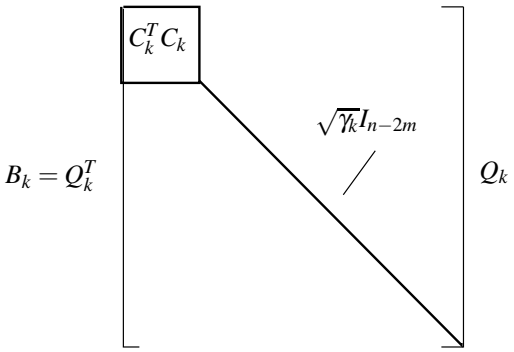

Fig. 8.2 The decomposition of the matrix B_k

Cholesky decomposition of the matrix $\gamma_k I_{2m} - T_k$, $\gamma_k I_{2m} - T_k = C_k^T C_k$ then eventually we come to the relation:

$$B_k = Q_k^T F_k^T F_k Q_k \tag{8.50}$$

with

$$F_k = \begin{bmatrix} C_k & 0 \\ 0 & \sqrt{\gamma_k} I_{n-2m} \end{bmatrix}. \tag{8.51}$$

The desired decomposition of the matrix B_k is thus given by

$$B_k = D_k^T D_k,\ D_k = F_k Q_k$$

where the matrix D_k is nonsingular provided that $s_i^T y_i > 0$ for $i = k-m, \ldots, k-1$. Notice that the matrix Q_k does not have to be stored since it can be easily evaluated from the Householder vectors which have been used in the QR factorization. These vectors can be stored in zero elements of the R_k matrix [87].

Recall the relations (8.5)–(8.6) which now can be written as

$$\begin{aligned} Q_k^T F_k^T \hat{g}_k &= g_k \\ F_k Q_k d_k &= \hat{d}_k. \end{aligned}$$

Solving these equations requires multiplication of vectors in $\mathscr{R}^n$ by the orthogonal matrix Q_k (or Q_k^T), and this can be achieved by the sequence of $2m$ multiplications of the Householder matrices H_k^i, $i = 1, \ldots, 2m$ such that $Q_k = H_k^1 H_k^2 \cdots H_k^{2m}$. The cost of these multiplications is proportional to n. Furthermore, we have to solve the set of n linear equations with the upper triangular matrix F_k, or its transpose. The cost of these operations is also proportional to n since the matrix F_k is of the form (8.51) and we assume that $m \ll n$.

8.5 Conjugate Gradient Algorithm with the BFGS Scaling Matrices

Algorithm 8.1 has to be specified in order to take into account two strategies outlined in the previous section. If we change the scaling matrix at every iteration (it corresponds to $n_r = 1$) then it does not make sense to use the previous direction through the term $\hat{\beta}_k d_{k-1}$ to calculate d_k since it corresponds to the situation when the conjugate gradient algorithm is restarted at every iteration. In this case we have a limited memory quasi-Newton method provided that strategy *S2* is applied.

Another possibility is to update D_k every n_r iterations. Then between the consecutive updates of the matrix D_k we apply conjugate gradient iteration as stated in (8.4). If we use strategy *S1* we have the following algorithm.

Algorithm 8.2. (The preconditioned method of shortest residuals – strategy S1)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\{\hat{\beta}_k\}$, m, n_r, $m < n_r$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute

$$d_1 = -g_1$$

and set $k = 1$, $r = 0$.

2. Find a positive number α_k satisfying (8.10)–(8.11). Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP.
If $k+1 < (r+1)n_r$ and $k+1 > rn_r + m$ go to Step 4.
If $k+1 = (r+1)n_r$ then substitute $B_{k+1} = B^0_{k+1} = \gamma_{k+1} I$ according to (8.39) (with r replaced by k) and increase r by one.
If $k+1 = (r+1)n_r + m$ then compute Q_r, F_r according to (8.47)–(8.51).
Compute d_{k+1} according to

$$B_{k+1} d_{k+1} = -g_{k+1} \tag{8.52}$$

with B_{k+1} obtained from B_k by the formula (8.38). Go to Step 5.
4. Solve

$$Q_r^T F_r^T \hat{g}_{k+1} = g_{k+1} \tag{8.53}$$

$$\hat{d}_{k+1} = -\mathbf{Nr}\{\hat{g}_{k+1}, -\hat{\beta}_{k+1}\hat{d}_k\} \tag{8.54}$$

$$F_r Q_r d_{k+1} = \hat{d}_{k+1} \tag{8.55}$$

5. Increase k by one and go to Step 2.

The global convergence of Algorithm 8.2 is a straightforward conclusion of Theorem 8.2 and Lemma 5.2.

Theorem 8.4. *Suppose that* $\{x_k\}$ *is generated by* Algorithm 8.2 *and*

(i) the condition (8.37) holds,
(ii) β_k *is given by (8.33),*

Then $\{x_k\}$ *converges to the minimizer of* f.

Proof. Because B_k is a symmetric positive definite matrix (since $s_k^T y_k > 0$, $k = rn_r + 1, \ldots, rn_r + m$ which follows from the fact that the modified Wolfe conditions are used to determine α_k), from the SVD decomposition (cf. Appendix B), its spectral norm $\|B_k\|_2$ is given by

$$\|B_k\|_2 = \lambda_1(B_k)$$

where $\lambda_1(B_k)$ is the largest eigenvalue of B_k. Analogously, the spectral norm of B_k^{-1} is stated by

$$\|B_k^{-1}\|_2 = \frac{1}{\lambda_n(B_k)}$$

where $\lambda_n(B_k)$ is the smallest eigenvalue of B_k. Furthermore, we have

$$\begin{aligned}\lambda_n(B_k)\|x\|^2 = \frac{\|x\|^2}{\|B_k^{-1}\|_2} &\leqslant x^T B_k x \leqslant \|B_k\|_2\|x\|^2 \\ &= \lambda_1(B_k)\|x\|^2.\end{aligned}$$

Consider now the sequence $\{B_k\}$. From Lemma 5.2 there exist positive constants d_1 and d_2 such that

$$\begin{aligned}\lambda_1(B_k) &\leqslant d_2, \\ \lambda_n(B_k) &\geqslant d_1\end{aligned}$$

for all k (it can be easily shown that the above estimates hold for B_k^0),
which imply that

$$d_1\|x\|^2 \leqslant x^T D_r^T D_r x \leqslant d_2\|x\|^2$$

for all r and $x \in \mathscr{R}^n$. This together with Theorem 8.2 imply the thesis. □

The proof of Theorem 8.4 establishes also that the matrices D_r generated by Algorithm 8.2 satisfy (8.18).

If strategy *S2* is applied we end up with the following version of a preconditioned conjugate gradient algorithm.

Algorithm 8.3. (The preconditioned method of shortest residuals – strategy S2)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\{\hat{\beta}_k\}$, m, n_r, $m < n_r$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute
$$d_1 = -g_1$$
and set $k = 1$, $r = 0$.
2. Find a positive number α_k satisfying (8.10)–(8.11). Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP.
If $k+1 = (r+1)n_r$ then substitute $B^0_{k+1} = \gamma_{k+1} I_n$ according to (8.40) (with k replaced by $k+1$), build B_{k+1} by using BFGS updating scheme and the pairs $\{s_i, y_i\}_{i=k-m+1}^{k}$, compute Q_{r+1}, F_{r+1} according to (8.47)–(8.51). Evaluate d_{k+1} according to (8.52). Increase r by one and go to Step 2.

Solve (8.53)–(8.55). Increase k by one and go to Step 2.

Algorithm 8.3 posses the same convergence properties as Algorithm 8.2.

8.6 Numerical Experiments

We present numerical results for Algorithm 8.3. In [181] both Algorithm 8.2 and Algorithm 8.3 are compared to L-BFGS-B code on problems from the CUTE collection. These results indicate that these versions of Algorithm 8.1 perform in similar way. Thus in this section we refer only to Algorithm 8.3. Furthermore, it is compared to L-BFGS code since, from the results presented in Sect. 5.5, we know that from two codes based on the limited memory BFGS scheme: L-BFGS-B and L-BFGS, the latter is more efficient.

In the implementation of Algorithm 8.3 we used directional minimization procedure described in [136]–Algorithm 8.3 required several obvious modifications of the procedure presented in Sect. 2.4. L-BFGS code was used with the parameter $m = 5$ and we applied different values of m and n_r as far as Algorithm 8.3 was concerned – in Table 8.1 the runs of Algorithm 8.3 with $m = 3$, $n_r = 12$ and $m = 5$, $n_r = 10$ are reported – in general we can expect that efficiency of Algorithm 8.3 (with respect to the number of function evaluations) increases with higher values of m and lower values of n_r. The results are given for problems from the CUTE collection with dimensions specified in Table 7.4. The stopping criterion was

$$\|g_k\| / \max(1, \|x_k\|) \leqslant 10^{-6}.$$

Results presented in Table 8.1 and in Figs. 8.3–8.7 show that Algorithm 8.3 is competitive to L–BFGS code but L–BFGS wins by some margin. These results are in line with those discussed in [125] where several versions of preconditioned conjugate gradient algorithms, including that of Buckley–LeNir and L–BFGS, are analyzed. One can say that Al–Baali–Fletcher theorem from Chap. 4 on the rate

Table 8.1 Numerical results: comparison of Algorithm 8.3–denoted as PSR in the table (with different values of m and n_r) to L-BFGS code

	PSR $m = 3$ $n_r = 12$		PSR $m = 5$ $n_r = 10$		L-BFGS $m = 5$	
Problem	IF	CPU	IF	CPU	IF	CPU
BROWNAL	27	0.41	27	0.42	15	0.23
BROYDN7D	3784	80.03	3965	95.73	3158	48.92
BRYBND	362	4.73	69	1.02	52	0.35
CHAINWOO	398	4.70	373	5.94	0	0.00
DIXON3DQ	5870	50.61	12207	156.99	10715	30.73
DQDRTIC	31	0.11	29	0.14	22	0.44
DQRTIC	40	0.29	40	0.48	47	0.06
EIGENALS	27870	537.43	25845	524.93	9938	180.51
EXTROSNB	1519	0.03	1364	0.04	1076	0.01
FLETCHBV	5134	65.35	7128	120.39	10102	65.34
FLETCHCR	8060	7.74	7433	9.46	5685	2.24
FMINSURF	1289	25.95	1078	28.50	1192	13.32
GENHUMPS	8675	56.78	8822	72.23	11001	40.29
GENROSE	1714	0.84	1616	1.07	1234	0.24
HILBERTA	82	0.01	23	0.01	31	0.01
LIARWD	43	0.38	33	0.42	28	0.13
MANCINO	22	0.28	22	0.28	15	0.19
MOREBV	64	0.30	58	0.33	45	0.10
NONCVXU2	4103	49.71	4957	79.42	4013	23.93
NONCVXUN	6182	74.94	9851	156.76	12879	76.24
NONDIA	16	0.13	16	0.15	23	0.13
NONDQUAR	756	6.66	510	6.29	558	1.62
POWELLSG	253	2.15	136	1.53	73	0.22
POWER	600	5.20	570	7.26	451	1.14
QUARTC	41	0.61	41	1.00	48	0.13
SCHMVETT	64	0.91	63	1.11	48	0.52
SENSORS	50	0.41	51	0.42	23	0.19
SPARSINE	11599	14.82	8927	15.30	8376	5.36
SPMSRTLS	286	3.40	236	3.59	213	1.35
SROSENBR	25	0.28	22	0.37	20	0.06
TOINTGSS	22	0.19	24	0.22	26	0.14
TQUARTIC	39	0.38	36	0.56	27	0.10
TRIDIA	3855	34.27	3655	47.33	3150	9.00
VAREIGVL	30	0.22	23	0.22	22	0.10
WOODS	454	4.16	227	3.17	121	0.42
Total	93359	1034.41	99477	1343.08	84427	503.76

of convergence of the preconditioned conjugate gradient algorithm finds here its justification.

CPU time of Algorithm 8.3 can be significantly reduced by devising different linear algebra approach (from that outlined in Sect. 4), for example by applying reduced Hessian approach – its use in the context of a preconditioned conjugate gradient algorithm is presented in Chap. 12.

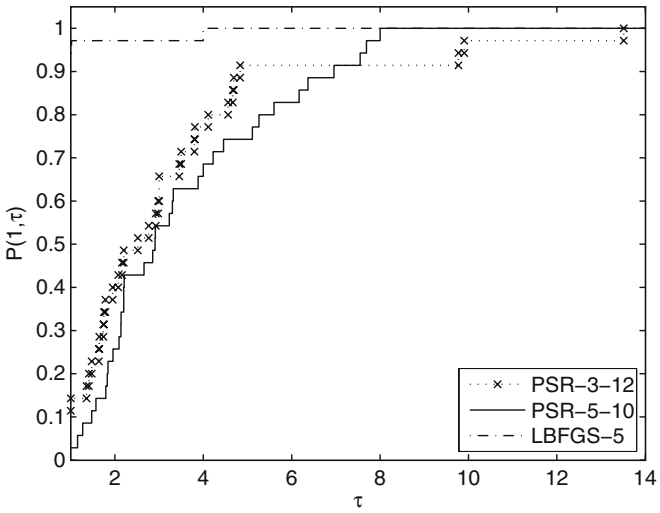

Fig. 8.3 Cumulative distribution ratio of CPU time: comparison of Algorithm 8.3 (denoted as PSR on the figure) to L-BFGS code

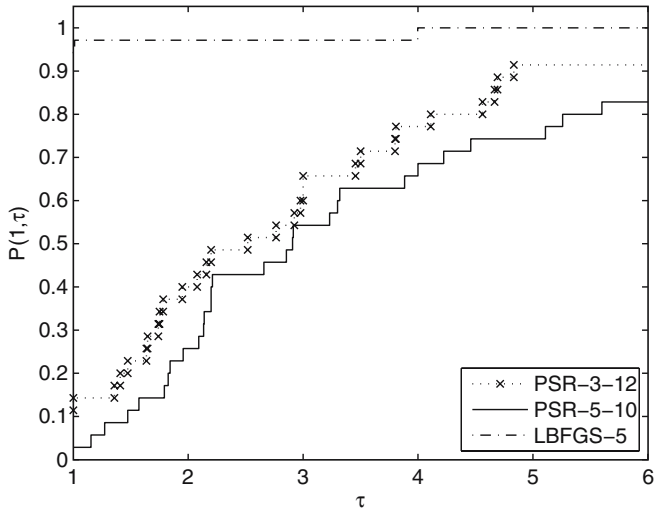

Fig. 8.4 Cumulative distribution ratio of CPU time: comparison of Algorithm 8.3 (denoted as PSR on the figure) to L-BFGS code

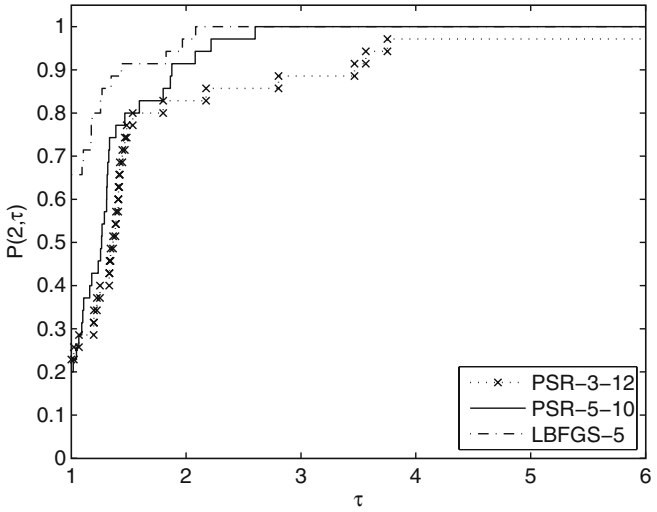

Fig. 8.5 Cumulative distribution ratio of the number of function evaluations: comparison of Algorithm 8.3 (denoted as PSR on the figure) to L-BFGS code

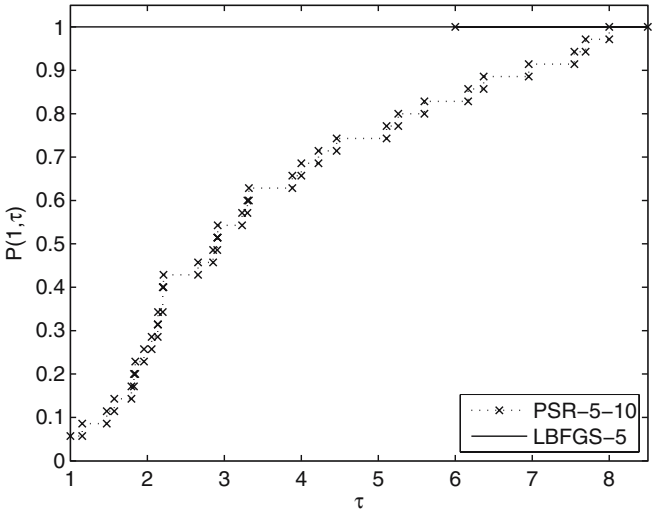

Fig. 8.6 Cumulative distribution ratio of CPU time: comparison of Algorithm 8.3 (denoted as PSR on the figure) to L-BFGS code

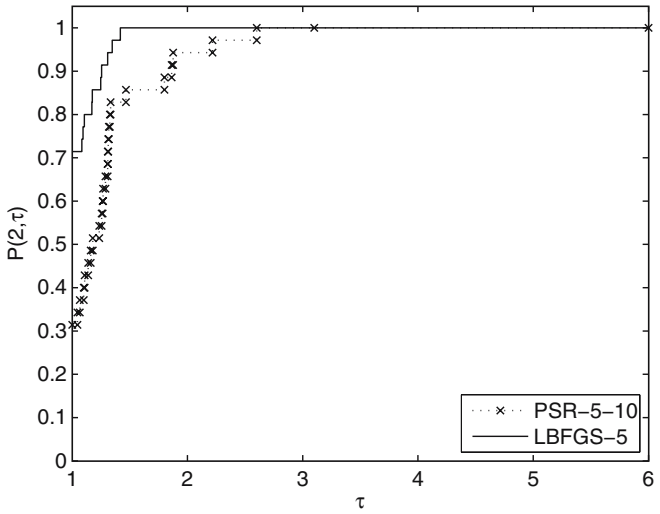

Fig. 8.7 Cumulative distribution ratio of the number of function evaluations: comparison of Algorithm 8.3 (denoted as PSR on the figure) to L-BFGS code

8.7 Notes

The chapter is based on the paper [181] (see also [180]) where a new approach to solving large scale nonlinear problems is presented. The numerical results show that it is a viable technique and competitive with the code L-BFGS. We think that our method gives the flexibility in achieving balance between the computing time and the number of function evaluations. The performance of the proposed method could be improved by using other updating scheme for matrices B_k and by ignoring pairs $\{s_i, y_i\}$ from the sequence $\{s_{k-1}, y_{k-1}\}, \ldots \{s_{k-m}, y_{k-m}\}$ if they are linearly dependent with the others. Notice that our approach is well-suited to this performance enhancement since during QR factorization we can identify these pairs.

The extension of the method to problems with box constraints is presented in Chap. 11.

Chapter 9
Optimization on a Polyhedron

9.1 Introduction

In the chapter our interest focuses on algorithms for problems with constraints. We consider the problem

$$\min_{x\in\Omega} f(x) \tag{9.1}$$

where

$$\Omega = \left\{x \in \mathscr{R}^n :\ c_j^T x \geqslant \delta_j,\ j = 1,\ldots,m\right\}. \tag{9.2}$$

The set Ω is convex and is called the *polyhedron* [187].

For problem (9.1)–(9.2) we provide a family of algorithms which find its stationary points and they accomplish it in such a way that close to a stationary point they perform, on a certain subspace, iterations of an algorithm for unconstrained optimization. If we define the set of active constraints $\mathscr{A}(x)$ as

$$\mathscr{A}(x) = \left\{j = 1,\ldots,m :\ c_j^T x = \delta_j\right\},$$

then the reduced subspace is

$$\left\{x \in \mathscr{R}^n :\ c_j^T x = \delta_j,\ j \in \mathscr{A}(\bar{x})\right\}$$

where $\bar{x}$ is a stationary point for problem (9.1)–(9.2).

In order to define a stationary point for problem (9.1)–(9.2) we need some geometric characterization of the set Ω. The results which follows from now on will concern an arbitrary convex set Ω – when the particular result concerns a polyhedron we will state it explicitly.

Definition 9.1. $\mathscr{N}(x)$ is the normal cone of a convex set Ω at x if

$$\mathscr{N}(x) = \left\{u \in \mathscr{R}^n :\ u^T (y - x) \leqslant 0,\ y \in \Omega\right\}.$$

The *tangent cone* $\mathscr{T}(x)$ can be defined as the dual of the normal cone $\mathscr{N}(x)$ (see

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

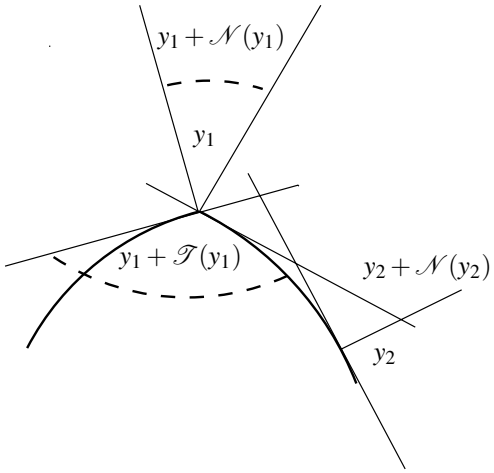

Fig. 9.1 Normal and tangent cones

Fig. 9.1). Thus

$$v \in \mathcal{T}(x) \;\Leftrightarrow\; v^T u \leqslant 0, \;\; \forall u \in \mathcal{N}(x). \tag{9.3}$$

An equivalent definition of the tangent cone is as follows.

Definition 9.2. $\mathcal{T}(x)$ is the tangent cone of a convex set Ω at x if

$$\mathcal{T}(x) = \mathrm{cl}\left(\{\lambda(u-x) : \; \lambda \geqslant 0, u \in \Omega\}\right).$$

Here, $\mathrm{cl}(A)$ denotes the closure of the set A (cf. Appendix A).

Definitions of different types of local solutions are straightforward extensions of the corresponding definitions for the unconstrained case, except that we now restrict consideration to the feasible points in the neighborhood of $\bar{x}$.

Definition 9.3.

(i) $\bar{x}$ is a local solution of problem (9.1)–(9.2) if $\bar{x} \in \Omega$ and there is a neighborhood $\mathcal{N}$ of $\bar{x}$ such that $f(x) \geqslant f(\bar{x})$ for all $x \in \mathcal{N} \bigcap \Omega$.

(ii) $\bar{x}$ is a strict local solution of problem (9.1)–(9.2) if $\bar{x} \in \Omega$ and there is a neighborhood $\mathcal{N}$ of $\bar{x}$ such that $f(x) > f(\bar{x})$ for all $x \in \mathcal{N} \bigcap \Omega$.

(iii) $\bar{x}$ is an isolated local solution of problem (9.1)–(9.2) if $\bar{x} \in \Omega$ and there is a neighborhood $\mathcal{N}$ of $\bar{x}$ such that $\bar{x}$ is the only local solution in $\mathcal{N} \bigcap \Omega$.

The first-order necessary optimality conditions for problem (9.1)–(9.2) are stated in the following theorem.

Theorem 9.1. *Suppose that f is continuously differentiable and $\bar{x}$ is a local solution of problem (9.1)–(9.2). Then the following equivalent conditions hold.*

(i)

$$g(\bar{x})^T (x - \bar{x}) \geqslant 0, \; x \in \Omega. \tag{9.4}$$

(ii)

$$-g(\bar{x}) \in \mathscr{N}(\bar{x}). \tag{9.5}$$

(iii)

$$g(\bar{x}) = \sum_{j \in \mathscr{A}(\bar{x})} \lambda_j c_j \tag{9.6}$$

where $\lambda_j \geqslant 0,\ j \in \mathscr{A}(\bar{x})$.

Proof. *(i).* Suppose that (9.4) does not hold, thus there exists $y \in \Omega$ such that

$$g(\bar{x})^T (y - \bar{x}) < 0. \tag{9.7}$$

Since Ω is convex, we have

$$\bar{x} + \alpha d \in \Omega, \ \ \forall \alpha \in [0,1], \tag{9.8}$$

where $d = y - \bar{x}$.

Since f is continuously differentiable, from the Taylor's expansion theorem, we can write

$$f(\bar{x} + \alpha d) = f(\bar{x}) + \alpha g(\bar{x})^T d + \int_0^\alpha \left[(g(\bar{x} + td) - g(\bar{x}))^T d \right] dt. \tag{9.9}$$

Combining (9.7)–(9.9) we conclude that there exists $\bar{\alpha} > 0$ such that

$$f(\bar{x} + \alpha d) < f(\bar{x})$$

for all $\alpha \in (0, \bar{\alpha}]$ which together with (9.8) contradict our assumption that $\bar{x}$ is a local solution of problem (9.1)–(9.2).

(ii). Since (9.4) holds, from the definition of the normal cone, we obtain (9.5).

(iii). In order to prove (9.6) notice that the normal cone of Ω at a point x is defined by [187]

$$\mathscr{N}(x) = \left\{ v \in \mathscr{R}^n : \ v = -\sum_{j \in \mathscr{A}(x)} \lambda_j c_j, \ \lambda_j \geqslant 0, \ j \in \mathscr{A}(x) \right\}. \tag{9.10}$$

Combining (9.5) and (9.10) leads to (9.6).

Assume now (9.6) is satisfied. Then $-g(\bar{x}) \in \mathscr{N}(\bar{x})$ according to the characterization of $\mathscr{N}(\bar{x})$. Then, from the definition of $\mathscr{N}(\bar{x})$ we have that (9.4) is also true. □

A point $\bar{x}$ which satisfies one the conditions (9.4)–(9.6) is called the *stationary point* for problem (9.1)–(9.2).

We are interested in projection gradient algorithms for problem (9.1)–(9.2). These algorithms follow the pattern in which the most important ingredient is that of determining a new point x_{k+1} according to the rule

$$x_{k+1} = \mathscr{P}_{\Omega}\left[x_k - \alpha_k g_k\right],$$

where $\mathscr{P}_{\Omega}[\cdot]$ is assumed to be the projection operator into a nonempty closed convex set Ω defined by

$$\mathscr{P}_{\Omega}[x] = \arg\min\left\{\|z-x\| : \ z \in \Omega\right\}. \tag{9.11}$$

The dependence of $\mathscr{P}_{\Omega}[\cdot]$ on Ω is usually clear from the context and thus we most often denote the projection mapping by $\mathscr{P}[\cdot]$.

The following lemma gives important characterizations of the projection mapping.

Lemma 9.1. *Suppose that $\mathscr{P}_{\Omega}$ is the projection mapping on a closed convex set Ω. Then,*

(i) condition (9.11) can be equivalently stated as

$$\left(x - \mathscr{P}_{\Omega}[x]\right)^T \left(\mathscr{P}_{\Omega}[x] - y\right) \geqslant 0, \ \ \forall\, y \in \Omega, \tag{9.12}$$

(ii) any point $x \in \mathscr{R}^n$ can be expressed in one and only one way as the sum of two orthogonal vectors: one in $\mathscr{N}(x)$ and the other in $\mathscr{T}(x)$, namely

$$x = \mathscr{P}_{\mathscr{N}(x)}[x] + \mathscr{P}_{\mathscr{T}(x)}[x].$$

Point *(ii)* of Lemma 9.1 states the Moreau decomposition.

Proof. *(i)*. We follow the proofs given in [212]. In order to prove (9.12) consider the identity

$$\begin{aligned}\|x - \mathscr{P}_{\Omega}[x]\|^2 - \|x-y\|^2 &= \|x - \mathscr{P}_{\Omega}[x]\|^2 - \\ &\quad \|x - \mathscr{P}_{\Omega}[x] + \mathscr{P}_{\Omega}[x] - y\|^2 \\ &= -2\left(x - \mathscr{P}_{\Omega}[x]\right)^T \left(\mathscr{P}_{\Omega}[x] - y\right) - \\ &\quad \|\mathscr{P}_{\Omega}[x] - y\|^2. \end{aligned} \tag{9.13}$$

Substituting $\hat{y} = \alpha y + (1-\alpha)\mathscr{P}_{\Omega}[x]$ ($\alpha \in [0,1]$) into (9.13) leads to

$$\begin{aligned}\|x - \mathscr{P}_{\Omega}[x]\|^2 - \|x-\hat{y}\|^2 &= -2\alpha\left(x - \mathscr{P}_{\Omega}[x]\right)^T \left(\mathscr{P}_{\Omega}[x] - y\right) \\ &\quad -\alpha^2 \|\mathscr{P}_{\Omega}[x] - y\|^2.\end{aligned}$$

Therefore, if (9.11) is satisfied we also have

$$2\alpha\left(x - \mathscr{P}_{\Omega}[x]\right)^T \left(\mathscr{P}_{\Omega}[x] - y\right) + \alpha^2 \|\mathscr{P}_{\Omega}[x] - y\|^2 \geqslant 0. \tag{9.14}$$

If we divide both sides of (9.14) by 2α and go with α to zero we obtain (9.12).

(ii). Suppose that the representation is not unique. Thus there exist $x_1,\ \hat{x}_1 \in \mathscr{N}(x)$ and $x_2,\ \hat{x}_2 \in \mathscr{T}(x)$ such that

$$x = x_1 + x_2,\ x_1^T x_2 = 0$$
$$x = \hat{x}_1 + \hat{x}_2,\ \hat{x}_1^T \hat{x}_2 = 0.$$

From these equalities we can write (cf. (9.3))

$$(x - x_1)^T (x_1 - \hat{x}_1) = x_2^T x_1 - x_2^T \hat{x}_1 \geqslant 0$$

for any $\hat{x}_1 \in \mathscr{N}(x)$. This and (9.12) imply that $x_1 = \mathscr{P}_{\mathscr{N}(x)}[x]$, and analogously that $x_2 = \mathscr{P}_{\mathscr{T}(x)}[x]$. Therefore, x_1 and x_2 are uniquely determined since $\mathscr{N}(x)$ and $\mathscr{T}(x)$ are closed sets.

It remains to show that the representation of x as a sum of orthogonal vectors such that one of them belongs to $\mathscr{N}(x)$ and the other to $\mathscr{T}(x)$ exists. From (9.12) we have

$$\left(x - \mathscr{P}_{\mathscr{N}(x)}[x]\right)^T \left(\mathscr{P}_{\mathscr{N}(x)}[x] - y\right) \geqslant 0,\ \ \forall y \in \mathscr{N}(x). \tag{9.15}$$

Take $y \in \mathscr{N}(x)$ then, from *Definition 9.1*, $y + \mathscr{P}_{\mathscr{N}(x)}[x] \in \mathscr{N}(x)$. Therefore, we can substitute $y + \mathscr{P}_{\mathscr{N}(x)}[x]$ in (9.15) instead of y getting

$$\left(x - \mathscr{P}_{\mathscr{N}(x)}[x]\right)^T y \leqslant 0,\ \ \forall y \in \mathscr{N}(x). \tag{9.16}$$

This implies that $x - \mathscr{P}_{\mathscr{N}(x)}[x] \in \mathscr{T}(x)$. Furthermore, setting $y = 0$ in (9.15) and $y = \mathscr{P}_{\mathscr{N}(x)}[x]$ in (9.16) one gets

$$\left(x - \mathscr{P}_{\mathscr{N}(x)}[x]\right)^T \mathscr{P}_{\mathscr{N}(x)}[x] \geqslant 0$$
$$\left(x - \mathscr{P}_{\mathscr{N}(x)}[x]\right)^T \mathscr{P}_{\mathscr{N}(x)}[x] \leqslant 0$$

which is equivalent to $\left(x - \mathscr{P}_{\mathscr{N}(x)}[x]\right)^T \mathscr{P}_{\mathscr{N}(x)}[x] = 0$. Therefore, if we set $x_1 = \mathscr{P}_{\mathscr{N}(x)}[x]$ and $x_2 = x - \mathscr{P}_{\mathscr{N}(x)}[x]$ then $x = x_1 + x_2$, $x_2 \in \mathscr{T}(x)$, $x_1^T x_2 = 0$ and, from the first part of the proof, this is the unique representation. □

We use the above lemma to express any of the conditions (9.4)–(9.6) using the projection mapping $\mathscr{P}$.

Theorem 9.2. *Let $\bar{x} \in \Omega$, then $\bar{x}$ is a stationary point for problem (9.1)–(9.2) if and only if one of the following conditions holds.*

(i)

$$\bar{x} = \mathscr{P}_{\Omega}\left[\bar{x} - \alpha g(\bar{x})\right] \tag{9.17}$$

for all $\alpha \geqslant 0$. Furthermore, (9.17) can be only checked for any particular $\alpha > 0$.

(ii)

$$\mathscr{P}_{\mathscr{T}(\bar{x})}\left[-g(\bar{x})\right] = 0. \tag{9.18}$$

Proof. *(i)*. According to part *(i)* of Lemma 9.1 $\mathscr{P}_{\Omega}$ can be characterized in terms of the scalar product by requiring that

$$(x-\mathscr{P}_{\Omega}[x])^T (y-\mathscr{P}_{\Omega}[x]) \leqslant 0 \tag{9.19}$$

for all $y \in \Omega$. Substitute $x+z$ for x in (9.19) where $x \in \Omega$ and $z \in \mathscr{N}(x)$ to obtain

$$(x+z-\mathscr{P}_{\Omega}[x+z])^T (y-\mathscr{P}_{\Omega}[x+z]) \leqslant 0$$

for all $y \in \Omega$. If we now take x as y then, after obvious transformations we will get

$$\|x-\mathscr{P}_{\Omega}[x+z]\|^2 \leqslant z^T (\mathscr{P}_{\Omega}[x+z]-x). \tag{9.20}$$

However, since $z \in \mathscr{N}(x)$ and $\mathscr{P}_{\Omega}[x+z] \in \Omega$, from the definition of the normal cone, we have

$$z^T (\mathscr{P}_{\Omega}[x+z]-x) \leqslant 0. \tag{9.21}$$

From (9.20)–(9.21) it follows that

$$\mathscr{P}_{\Omega}[x+z] = x \tag{9.22}$$

for all $x \in \Omega$ and $z \in \mathscr{N}(x)$.

If we assume that $\bar{x}$ is a stationary point for problem (9.1)–(9.2), then from (9.5), $-\alpha g(\bar{x}) \in \mathscr{N}(\bar{x})$ for any $\alpha \geqslant 0$, and (9.22) (with $\bar{x}$ as x) imply that (9.17) is valid.

Suppose now that (9.17) holds for some particular $\bar{\alpha} > 0$. However, it implies that $-\bar{\alpha} g(\bar{x}) \in \mathscr{N}(\bar{x})$ and, since $\mathscr{N}(\bar{x})$ is a cone, we must also have $-\alpha g(\bar{x}) \in \mathscr{N}(\bar{x})$ for any $\alpha \geqslant 0$.

Assume now that (9.17) holds with $\alpha = 1$. Then, from (9.19) we have

$$(x-g(\bar{x})-x)^T (y-x) \leqslant 0$$

for all $y \in \Omega$. It implies that (9.4) is true thus $\bar{x}$ is a stationary point.

(ii). Suppose that $\bar{x}$ is a stationary point for problem (9.1)–(9.2). Then, from (9.6) we know that there exist $\lambda_j \geqslant 0$, $j \in \mathscr{A}(\bar{x})$ such that

$$g(\bar{x}) = \sum_{j \in \mathscr{A}(\bar{x})} \lambda_j c_j.$$

From the Moreau decomposition we can write (cf. Fig. 9.2)

$$\mathscr{P}_{\mathscr{N}(\bar{x})}[-g(\bar{x})] + \mathscr{P}_{\mathscr{T}(\bar{x})}[-g(\bar{x})] = -g(\bar{x}),$$

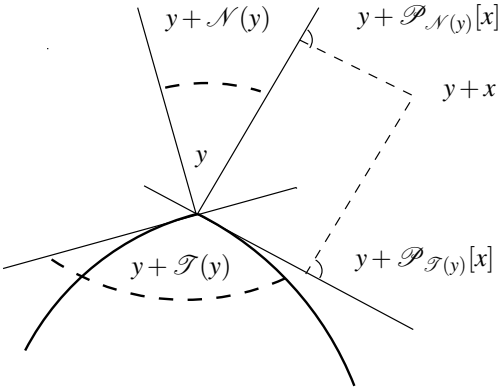

Fig. 9.2 The Moreau decomposition

which enables us to express $\mathscr{P}_{\mathscr{T}(\bar{x})}[-g(\bar{x})]$ in terms of $\mathscr{P}_{\mathscr{N}(\bar{x})}[-g(\bar{x})]$ and $g(\bar{x})$. Notice that $\mathscr{P}_{\mathscr{N}(\bar{x})}[-g(\bar{x})]$ is calculated by solving the problem

$$\min_{v\in\mathscr{N}(\bar{x})} \|v+g(\bar{x})\|. \tag{9.23}$$

According to (9.10) the solution of (9.23) is given by

$$v=-\sum_{j\in\mathscr{A}(\bar{x})}\lambda_j c_j \tag{9.24}$$

where λ_j, $j\in\mathscr{A}(x)$ solve the bound constrained linear least–squares problem

$$\min\left[\|g(x)-\sum_{j\in\mathscr{A}(x)}\lambda_j c_j\|:\ \lambda_j\geqslant 0,\ j\in\mathscr{A}(x)\right] \tag{9.25}$$

From (9.23) and (9.24)–(9.25) we conclude that

$$\mathscr{P}_{\mathscr{T}(x)}[-g(x)]=-g(x)+\sum_{j\in\mathscr{A}(x)}\lambda_j c_j \tag{9.26}$$

with λ_j, $j\in\mathscr{A}(x)$ defined in (9.24). Thus, if $\mathscr{P}_{\mathscr{T}(\bar{x})}[-g(\bar{x})]=0$, then from the characterization (9.26) $\bar{x}$ satisfies (9.6) thus $\bar{x}$ is a stationary point. □

The projection of $-g(x)$ on the set $\mathscr{T}(x)$ is often denoted by $\nabla_\Omega f(x)$ and is called the *projected gradient* (cf. [30]):

$$\nabla_\Omega f(x)=\mathscr{P}_{\mathscr{T}(x)}[-g(x)].$$

Conditions (9.17)–(9.18), guaranteeing that $\bar{x}$ is a stationary point for problem (9.1)–(9.2), are used in optimization algorithms. In general, it is easier to verify that

an algorithm generates a sequence $\{x_k\}$ such that

$$\lim_{k\to\infty} \|\mathscr{P}_{\Omega}[x_k - g_k] - x_k\| = 0, \tag{9.27}$$

then to show that it has the property

$$\lim_{k\to\infty} \mathscr{P}_{\mathscr{T}(x_k)}[-g_k] = 0. \tag{9.28}$$

We show, following [24], that with (9.28) we can associate the following feature of an algorithm: there exists k_0 such that for all $k \geqslant k_0$

$$c_j^T x_k = \delta_j,\ j \in \mathscr{A}(\bar{x}) \tag{9.29}$$

and

$$c_j^T x_k > \delta_j,\ j \notin \mathscr{A}(\bar{x}). \tag{9.30}$$

If an algorithm has the identification property (9.29)–(9.30) then after a finite number of iterations the algorithm performs iterations of the method for the unconstrained problem defined by active constraints at the stationary point.

An algorithm guaranteeing (9.27) has also the property of identifying the active set of constraints at a stationary point but under a nondegeneracy condition. Our definition of a nondegenerate point follows that given in [58].

Definition 9.4. A stationary point $\bar{x}$ of problem (9.1)–(9.2) is nondegenerate if

$$-g(\bar{x}) \in \mathrm{ri}\,(\mathscr{N}(\bar{x}))\,. \tag{9.31}$$

Here, $\mathrm{ri}(S)$ is the *relative interior* of S defined as the interior of S relative to $\mathrm{aff}(S)$ (cf. Appendix A). The *affine hull*–$\mathrm{aff}(S)$ is the smallest affine set which contains S.

Dunn's condition (9.31) can be expressed in terms of Lagrange multipliers associated with active constraints. We recall that

$$\mathscr{N}(x) = \left\{ v \in \mathscr{R}^n :\ v = -\sum_{j\in\mathscr{A}(x)} \lambda_j c_j,\ \lambda_j \geqslant 0,\ j \in \mathscr{A}(x) \right\}.$$

Burke and Moré [23] give the characterization of the polyhedral cone $\mathscr{N}(x)$.

Theorem 9.3. *$v \in \mathrm{ri}\,(\mathscr{N}(x))$ if and only if*

$$v \in \left\{ z \in \mathscr{R}^n :\ z = -\sum_{j\in\mathscr{A}(x)} \lambda_j c_j,\ \lambda_j > 0,\ j \in \mathscr{A}(x) \right\}.$$

Proof. We follow the proof given in [23]. Suppose that

$$v = -\sum_{j\in\mathscr{A}(x)} \lambda_j c_j \tag{9.32}$$

with $\lambda_j > 0$, $j \in \mathscr{A}(x)$. We want to show that $v \in \mathrm{ri}\,(\mathscr{N}(x))$. To this end define by $\mathscr{I}$ the subset of $\mathscr{A}(x)$ for which $\{c_j\}_{j\in\mathscr{I}}$ is the basis of

$$\mathrm{aff}\,(\mathscr{N}(x)) = \mathrm{span}\left\{c_j :\ j \in \mathscr{A}(x)\right\}.$$

If $w \in \mathrm{aff}\,(\mathscr{N}(x))$ and $\|w\| \leqslant \varepsilon$, where $\varepsilon > 0$ is some small number, then

$$w = -\sum_{j\in\mathscr{I}} \mu_j c_j,\ |\mu_j| < \min\left[\lambda_j :\ j \in \mathscr{A}(x)\right].$$

This implies that $v + w \in \mathscr{N}(x)$ thus $v \in \mathrm{ri}\,(\mathscr{N}(x))$.

Assume now that $v \in \mathrm{ri}\,(\mathscr{N}(x))$, we want to show that v can be expressed as in (9.32). Take v_0 such that it is in $\mathrm{span}\left\{-c_j :\ j \in \mathscr{A}(x)\right\}$ with coefficients $\lambda_j^0 > 0$, $j \in \mathscr{A}(x)$. Due to the fact that $v \in \mathrm{ri}\,(\mathscr{N}(x))$ there exists $\alpha > 1$ such that $\alpha v + (1-\alpha)v_0 = w$ and $w \in \mathscr{N}(x)$. This implies that

$$v \in \left(\frac{1}{\alpha}\mathscr{N}(x) + \left(1 - \frac{1}{\alpha}\right)v_0\right),$$

which states that there exist $\lambda_j \geqslant 0$, $j \in \mathscr{A}(x)$ such that

$$v = -\sum_{j\in\mathscr{A}(x)} \left(\frac{1}{\alpha}\lambda_j + \left(1 - \frac{1}{\alpha}\right)\lambda_j^0\right) c_j. \tag{9.33}$$

Since $\lambda_j^0 > 0$, $j \in \mathscr{A}(x)$ (9.33) says that $v \in \mathrm{span}\left\{-c_j :\ j \in \mathscr{A}(x)\right\}$ with coefficients $\lambda_j^v > 0$, $j \in \mathscr{A}(x)$. □

Theorem 9.3 can be used to rephrase the nondegenaracy condition (9.31) by saying that $\bar{x}$ is the nondegenerate point if and only if there exists Lagrange multipliers λ_j, $j \in \mathscr{A}(\bar{x})$ such that

$$g(\bar{x}) = \sum_{j\in\mathscr{A}(\bar{x})} \lambda_j c_j \tag{9.34}$$

and $\lambda_j > 0$ for all $j \in \mathscr{A}(\bar{x})$. Notice that we do not require that the set of the Lagrange multipliers satisfying (9.34) is unique. Therefore, a stationary point is nondegenerate if a strict complementarity condition holds. We do not make any assumption on linear independence of the normals of the active constraints.

If $\{c_j\}_{j\in\mathscr{A}(\bar{x})}$ are linearly independent then we can show that the set of Lagrange multipliers λ_j, $j \in \mathscr{A}(\bar{x})$, for which (9.34) holds, is unique. Indeed, if both sets $\{\lambda_j^1\}_{j\in\mathscr{A}(\bar{x})}$, $\{\lambda_j^2\}_{j\in\mathscr{A}(\bar{x})}$ satisfy (9.34) then

$$\sum_{j\in\mathscr{A}(\bar{x})} \left(\lambda_j^1 - \lambda_j^2\right) c_j = 0$$

and, due to our assumption, $\lambda_j^1 = \lambda_j^2$, $j \in \mathscr{A}(\bar{x})$.

9.2 Exposed Sets

We look for conditions under which algorithms for problem (9.1)–(9.2) identify active constraints at a stationary point after a finite number of iterations. Suppose that an algorithm generates sequence $\{x_k\}$ such that $x_k \to \bar{x}$ and $\bar{x}$ is a stationary point for problem (9.1)–(9.2). If, after a finite number of iterations, active constraints are identified it means that

$$x_k \in \left\{x \in \mathscr{R}^n : \; c_j^T x = \delta_j, \; j \in \mathscr{A}(\bar{x}), \; c_j^T x > \delta_j, \; j \notin \mathscr{A}(\bar{x})\right\} \tag{9.35}$$

for all $k \geqslant k_0$ where k_0 is some finite number.

Our analysis of the identification properties is based on geometry of the set which appears in (9.35). The following definition is crucial in the analysis.

Definition 9.5. A subset $\mathscr{E}(d)$ of a convex set Ω is the *exposed set* of Ω by a vector $d \in \mathscr{R}^n$ if

$$\mathscr{E}(d) = \arg\max \left[x^T d : \; x \in \Omega\right].$$

In literature [187], [204], [24], [22], as far as polyhedral sets are concerned, the terminology of *faces* of a set Ω and the exposed sets are used interchangeably. Therefore, if a face is denoted by $\mathscr{F}$ then $\mathscr{F} = \mathscr{E}(d)$.

It is easy to verify that the set in (9.35) is the exposed set of Ω by $\bar{x}$ therefore condition (9.35) can be equivalently stated as

$$x_k \in \mathscr{E}(\bar{x}), \; \forall k \geqslant k_0.$$

The exposed sets and normal cones are related if we observe that

$$x \in \mathscr{E}(d) \;\Leftrightarrow\; d \in \mathscr{N}(x). \tag{9.36}$$

Indeed, if $x \in \mathscr{E}(d)$ then $d^T(y-x) \leqslant 0$ for all $y \in \Omega$ which says that $d \in \mathscr{N}(x)$. The implication in the other direction can be proved in the same way.

The following lemma gives the characterization of the exposed sets.

Lemma 9.2. *Suppose that Ω is defined by (9.2). Then*

(i) if

$$d = \sum_{j \in \mathscr{I}} \lambda_j c_j, \; \lambda_j \geqslant 0, \; j \in \mathscr{I}, \; \mathscr{I} \subset \{1, \ldots, m\} \tag{9.37}$$

then

$$\mathscr{E}(d) = \left\{x \in \Omega : \; c_j^T x = \delta_j, \; j \in \mathscr{I} \text{ and } \lambda_j > 0\right\},$$

(ii) if $x, y \in \mathscr{E}(d)$ then $(x-y) \perp d$,
(iii) for any x, $y \in \mathrm{ri}(\mathscr{F})$, where $\mathscr{F}$ is a face of Ω, $\mathscr{A}(x) = \mathscr{A}(y)$,
(iv) $\bar{x}$ is a stationary point for problem (9.1)–(9.2) if and only if $\bar{x} \in \mathscr{E}(-g(\bar{x}))$.

Proof. *(i).* Suppose that d is defined by (9.37). Thus we can write

$$d^T x = \sum_{j \in \mathscr{I}} \lambda_j c_j^T x \geqslant \sum_{j \in \mathscr{I}} \lambda_j \delta_j$$

if $x \in \Omega$, $d^T x$ is maximized only when $c_j^T x = \delta_j$ for $\lambda_j > 0$.

(ii). It follows from the fact that $d^T x$ is constant, when $x \in \mathscr{E}(d)$.

(iii). As in [24] take $x \in \mathrm{ri}(\mathscr{F})$ and $f \in \mathscr{F}$. Since x is in the relative interior of $\mathscr{F}$ there exists $\alpha < 0$ such that

$$x_\alpha = x + \alpha(x - y) \in \mathscr{F}.$$

Therefore, for $j \in \mathscr{A}(x)$ we have

$$\delta_j \leqslant c_j^T x_\alpha = (1-\alpha) c_j^T x + \alpha c_j^T y = (1-\alpha)\delta_j + \alpha c_j^T y.$$

This together with $y \in \mathscr{F}$ imply that $j \in \mathscr{A}(y)$. If we now assume $y \in \mathrm{ri}(\mathscr{F})$ and $x \in \mathscr{F}$ we obtain the reverse inclusion $\mathscr{A}(y) \subset \mathscr{A}(x)$.

(iv). Combine (9.5) and (9.36) to get the thesis. □

According to part *(iii)* of Lemma 9.2 we can introduce the following definition.

Definition 9.6. If $\mathscr{F}$ is a face of a polyhedral set Ω, then we define $\mathscr{A}(\mathscr{F})$ as

$$\mathscr{A}(\mathscr{F}) = \mathscr{A}(x)$$

for any $x \in \mathrm{ri}(\mathscr{F})$.

Notice that we cannot use a point y from the boundary of $\mathscr{F}$ since then $\mathscr{A}(x) \subset \mathscr{A}(y)$ and the inclusion is strict for any $x \in \mathrm{ri}(\mathscr{F})$.

Lemma 9.2 enables us to characterize all faces of a polyhedron.

Lemma 9.3. *For a polyhedral set Ω we have*

$$\mathscr{F} = \left\{ x \in \Omega : \; c_j^T x = \delta_j, \; j \in \mathscr{A}(\mathscr{F}) \right\} \tag{9.38}$$

for any face $\mathscr{F}$ of Ω. Moreover, there exist only a finite number of different faces of Ω.

Proof. We can show that $x \in \mathrm{ri}(\mathscr{E}(d))$ if and only if $d \in \mathrm{ri}(\mathscr{N}(x))$. Consider any face $\mathscr{F}$ of Ω. It is also an exposed set thus there exists $d \in \mathscr{R}^n$ such that $\mathscr{F} = \mathscr{E}(d)$. Therefore, $d \in \mathrm{ri}(\mathscr{N}(x))$ if $x \in \mathrm{ri}(\mathscr{F})$. If we recall Theorem 9.3 and part (i) of Lemma 9.2 we will obtain (9.38). Since the number of different subsets of the set $\{1, 2, \ldots, m\}$ is finite the proof is completed. □

Figure 9.3 (based on Fig. 3.1 in [123]) illustrates some of the properties of exposed sets. $\bar{x}$ is a degenerate point since it lies in the relative boundary of the exposed set. Notice that in this case $\mathscr{A}(\bar{x})$ is not a subset of $\mathscr{A}(x)$ for some $x \in \mathscr{E}(-g(\bar{x}))$. It is straightforward to show that if Ω is defined by: $\Omega = \{x \in \mathscr{R}^n : l_i \leqslant x_i \leqslant u_i, \; i = 1, \ldots, n\}$ then

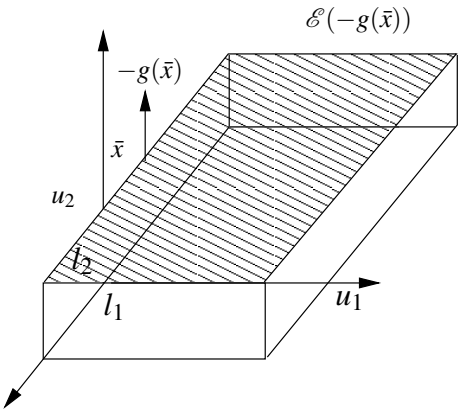

Fig. 9.3 A degenerate stationary point

$$\mathscr{E}(-g(\bar{x})) = \left\{ x \in \Omega : \; x_i = l_i \text{ if } \frac{\partial f}{\partial x_i}(\bar{x}) > 0 \text{ and } x_i = u_i \text{ if } \frac{\partial f}{\partial x_i}(\bar{x}) < 0 \right\}.$$

9.3 The Identification of Active Constraints

According to Lemma 9.2 (9.35) can be stated as

$$x_k \in \mathscr{E}(-g(\bar{x})), \;\; \forall\, k \geqslant k_0. \tag{9.39}$$

The question under which conditions the set of active constraints is identified after a finite number of iterations is crucial for many optimization algorithms which owe their rate of convergence to this fact. In particular, in the next chapter we present a family of conjugate gradient algorithms for problems with constraints defined by a polyhedron. Suppose that after a finite number of iterations the active constraints at a solution are identified. Then, the method performs their iterations on the subspace of active constraints which does not change until a solution is found. If this happens we can expect that behavior of the method will be comparable to efficient conjugate gradient algorithms and not to a steepest descent method.

Notice first that for any convergent sequence $\{x_k\}$ we have

$$\lim_{k\to\infty} x_k = \bar{x} \;\Rightarrow\; \lim_{k\to\infty} \mathscr{A}(x_k) \subset \mathscr{A}(\bar{x}). \tag{9.40}$$

This follows from the following fact.

Lemma 9.4. *The mapping $\Omega \ni x \to \mathscr{A}(x)$ is upper semicontinuous.*

Proof. Consider any convergent sequence $\{x_k\}$ with $\bar{x}$ as its limit point. We have to show that for any index $j \in \{1,2,\ldots,m\}$, if $j \in \mathscr{A}(x_k)$ for infinitely many k, then $j \in \mathscr{A}(\bar{x})$. Indeed,

$$\lim_{k\to\infty} \left[c_j^T x_k = \delta_j\right] = c_j^T \bar{x} = \delta_j.$$

□

Therefore, the identifying property holds if for sufficiently large values of k we have

$$\mathscr{A}(\bar{x}) \subset \mathscr{A}(x_k). \tag{9.41}$$

In order to prove (9.41) we start with the result proved by Robinson [186].

Theorem 9.4. *For a polyhedral set Ω and any $\bar{d} \in \mathscr{R}^n$ there is a neighborhood $\mathscr{B}(\bar{d})$ of $\bar{d}$ such that for any $d \in \mathscr{B}(\bar{d})$*

$$\mathscr{E}(d) \subset \mathscr{E}(\bar{d}).$$

Proof. Suppose that the thesis is not true. It means that there is a sequence $\{d_k\}$ convergent to $\bar{d}$ such that for infinitely many k, $\mathscr{E}(d_k)$ is not a subset of $\mathscr{E}(\bar{d})$. For the simplicity of presentation denote this subsequence by $\{d_k\}$. We know, from Lemma 9.3, that there are only a finite number of faces (which are also exposed sets) of the set Ω. Therefore, we can assume that for some face $\mathscr{F}$ of Ω, $\mathscr{F} = \mathscr{E}(d_{k_l})$ where $\{d_{k_l}\}$ is a subsequence of $\{d_k\}$.

Consider any $z \in \mathscr{F}$. Then $d_{k_l}^T z \geqslant d_{k_l}^T x$ for any $x \in \Omega$. Since $d_{k_l} \to \bar{d}$ we also have $\bar{d}^T z \geqslant \bar{d}^T x$ for any $x \in \Omega$ and thus $\mathscr{F} \subset \mathscr{E}(\bar{d})$ which is the contradiction to our assumption. □

Theorem 9.4 states the property of the mapping $\mathscr{R}^n \ni d \to \mathscr{E}(d)$ similar to that of $\mathscr{A}(\cdot)$ given in (9.40).

Our derivation of (9.41) is in two stages. In the first one we prove (9.39) under a certain condition imposed on the sequence of the projected gradients. Then we show that nondegeneracy of $\bar{x}$ leads to (9.41).

Theorem 9.5. *Suppose that f is continuously differentiable and that $\{x_k\}$ is a sequence which converges to a stationary point $\bar{x}$ for problem (9.1)–(9.2). Then*

$$\lim_{k\to\infty} \mathscr{P}_{\mathscr{T}(x_k)}[-g_k] = 0 \tag{9.42}$$

if and only if there is a $k_0 > 0$ such that

$$x_k \in \mathscr{E}(-g(\bar{x})) \tag{9.43}$$

for all $k \geqslant k_0$.

Proof. Notice that if $\{x_k\}$ and $\{d_k\}$ are sequences such that $x_k \in \Omega$, $d_k \in \mathcal{N}(x_k)$ and $x_k \to \bar{x}$, $d_k \to \bar{d}$ then

$$x_k \in \mathcal{E}(\bar{d}) \tag{9.44}$$

for all k sufficiently large. In order to prove that it is sufficient to recall that $x_k \in \mathcal{E}(d_k)$ and then to apply Theorem 9.4.

Suppose that (9.42) holds. From the Moreau decomposition we have

$$-g_k = \mathcal{P}_{\mathcal{T}(x_k)}[-g_k] + \mathcal{P}_{\mathcal{N}(x_k)}[-g_k].$$

By taking $d_k = \mathcal{P}_{\mathcal{N}(x_k)}[-g_k]$ we obtain

$$-g_k = \mathcal{P}_{\mathcal{T}(x_k)}[-g_k] + d_k$$

with $d_k \in \mathcal{N}(x_k)$. Since $d_k \to -g(\bar{x})$, from (9.42) and (9.44), we have $x_k \in \mathcal{E}(-g(\bar{x}))$ for all k sufficiently large.

Assume now that (9.43) is true for all k sufficiently large. This means that $-g(\bar{x}) \in \mathcal{N}(x_k)$ for these k. Therefore,

$$\|\mathcal{P}_{\mathcal{T}(x_k)}[-g_k]\| \leqslant \|g_k - g(\bar{x})\|$$

due to the fact that $\mathcal{P}_{\mathcal{T}(x_k)}[-g_k]$ is found by solving (9.23) with $\bar{x}$ replaced by x_k. Since $x_k \to \bar{x}$ and g is continuous we have (9.42). □

Unfortunately the above result does not state (9.35). This happens if we additionally assume that a stationary point $\bar{x}$ is nondegenerate.

Theorem 9.6. *Suppose that f is continuously differentiable and that $\{x_k\}$ is a sequence which converges to a nondegenerate stationary point $\bar{x}$ for problem (9.1)–(9.2). Then*

$$\lim_{k\to\infty} \mathcal{P}_{\mathcal{T}(x_k)}[-g_k] = 0$$

if and only if there is a $k_0 > 0$ such that

$$x_k \in \mathcal{F}(\bar{x})$$

for all $k \geqslant k_0$.

Proof. We again recall that for o polyhedral set Ω and $x \in \Omega$ we have that $x \in \mathrm{ri}(\mathcal{E}(d))$ if and only if $d \in \mathrm{ri}(\mathcal{N}(x))$.

Since $\bar{x}$ is a nondegenerate stationary point $\bar{x} \in \mathrm{ri}(\mathcal{E}(-g(\bar{x}))$. Therefore, for sufficiently large values of k, $x_k \in \mathrm{ri}(\mathcal{E}(-g(\bar{x}))$. In order to complete the proof notice that any exposed set is a face of Ω and since $\bar{x} \in \mathrm{ri}(\mathcal{E}(-g(\bar{x})))$ we can write $\mathcal{F} = \mathcal{F}(\bar{x}) = \mathcal{E}(-g(\bar{x}))$. □

According to Lemma 9.2 the face $\mathcal{E}(-g(\bar{x}))$ can be expressed in terms of the positive Lagrange multipliers at $\bar{x}$. Thus the following result holds.

Theorem 9.7. *Suppose that f is continuously differentiable and that $\{x_k\}$ is a sequence which converges to a stationary point $\bar{x}$ for problem (9.1)–(9.2). Assume that $\bar{\lambda}_j$, $j \in \mathscr{A}(\bar{x})$ satisfy*

$$g(\bar{x}) = \sum_{j \in \mathscr{A}(\bar{x})} \bar{\lambda}_j c_j, \ \bar{\lambda}_j \geqslant 0, \ j \in \mathscr{A}(\bar{x}). \tag{9.45}$$

Then

$$\lim_{k \to \infty} \mathscr{P}_{\mathscr{T}(x_k)}[-g_k] = 0$$

if and only if there is a $k_0 > 0$ such that

$$\left\{ i \in \mathscr{A}(\bar{x}) : \ \bar{\lambda}_j > 0 \right\} \subset \mathscr{A}(x_k)$$

for all $k \geqslant k_0$.

Proof. From Lemma 9.2 we know that

$$x \in \mathscr{C}(-g(\bar{x})) \ \Leftrightarrow \ \left\{ i \in \mathscr{A}(\bar{x}) : \ \bar{\lambda}_j > 0 \right\} \subset \mathscr{A}(x)$$

and the thesis follows from Theorem 9.5. □

Eventually we can rephrase Theorem 9.6 in terms of indices of active constraints.

Theorem 9.8. *Suppose that f is continuously differentiable and that $\{x_k\}$ is a sequence which converges to a nondegenerate stationary point $\bar{x}$ for problem (9.1)–(9.2). Then*

$$\lim_{k \to \infty} \mathscr{P}_{\mathscr{T}(x_k)}[-g_k] = 0$$

if and only if there is a $k_0 > 0$ such that

$$\mathscr{A}(x_k) = \mathscr{A}(\bar{x})$$

for all $k \geqslant k_0$.

Proof. If $\bar{x}$ is nondegenerate then for any Lagrange multipliers satisfying (9.45) we have $\bar{\lambda}_j > 0$, $j \in \mathscr{A}(\bar{x})$. Therefore, from Theorem 9.7,

$$\mathscr{A}(\bar{x}) \subset \mathscr{A}(x_k)$$

for large values of k. Lemma 9.4 gives the desired result. □

9.4 Gradient Projection Method

In previous sections we analyzed the identifying properties of the projection operator. In this section we introduce an algorithm which uses the projection and is globally convergent.

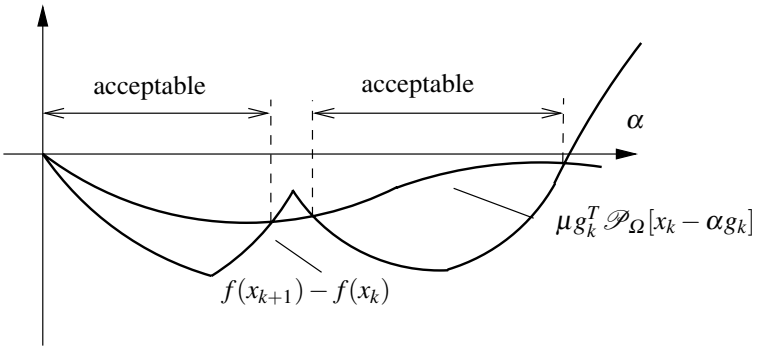

Fig. 9.4 The generalized Armijo line search rule

The crucial step of the algorithm is that of determining the next point x_{k+1}:

$$x_{k+1} = \mathscr{P}_\Omega [x_k - \alpha_k g_k]$$

where Ω is a polyhedron as defined in (9.2). How α_k is calculated has important impact on the efficiency of the iteration process. Goldstein [85] and Levitin and Polyak [122] propose α_k which satisfies

$$\varepsilon \leqslant \alpha_k \leqslant \frac{2}{L}(1-\varepsilon)$$

where $\varepsilon \in (0,1)$ is some predetermined number and L is a Lipschitz constant for the gradient g. There are obvious drawbacks of this approach – ε is not known in advance and L has to be estimated. Therefore, McCormick and Tapia [132] suggest using α_k which is an argument of the minimization problem

$$\min_{\alpha>0} f\left(\mathscr{P}_\Omega [x_k - \alpha g_k]\right). \tag{9.46}$$

The line search rule (9.46) is very expensive to apply, moreover $f(\mathscr{P}_\Omega[\cdot])$ is not a differentiable function, thus the rule (9.46) is rarely used in practice

Bertsekas [8] overcomes the drawbacks of the previous line search rules by proposing the Armijo step-size rule (see Fig. 9.4).

Algorithm 9.1. (The generalized Armijo line search algorithm)

Parameters: $s > 0$, μ, $\beta \in (0,1)$.

1. Set $\alpha = s$.
2. Set

$$x_{k+1} = \mathscr{P}_\Omega [x_k - \alpha g_k].$$

If

$$f(x_{k+1}) \leqslant f(x_k) + \mu g_k^T (x_{k+1} - x_k)$$

substitute α for α_k and STOP.

3. Substitute $\beta\alpha$ for α, go to Step 2.

It is not difficult to show that if x_k is not a stationary point then the above line search algorithm will stop after a finite number of iterations. In order to show that we first establish the following result.

Lemma 9.5. *For every $x,\ y \in \Omega$ and $\alpha \geqslant 0$ we have*

$$\alpha g(x)^T (y - x(\alpha)) \geqslant (x - x(\alpha))^T (y - x(\alpha)) \tag{9.47}$$

where $x(\alpha)$ is defined by $x(\alpha) = \mathscr{P}_\Omega[x - \alpha g(x)]$.

Proof. Since $x(\alpha)$ is determined by the projection we have

$$(x - \alpha g(x) - x(\alpha))^T (y - x(\alpha)) \leqslant 0 \tag{9.48}$$

for any $x,\ y \in \Omega$ and $\alpha \geqslant 0$. Equation (9.47) follows from (9.48). □

The next lemma plays a crucial role in our convergence analysis.

Lemma 9.6. *Suppose that g is a Lipschitz continuous function, i.e. there exists a finite positive number L such that*

$$\|g(y) - g(x)\| \leqslant L\|y - x\|,\ \ \forall\, x, y \in \Omega. \tag{9.49}$$

Then, for any point $x \in \Omega$ and any scalar $\mu \in (0,1)$ the inequalities

$$\alpha [f(x) - f(x(\alpha))] \geqslant \mu \|x - x(\alpha)\|^2 \tag{9.50}$$

$$f(x(\alpha)) - f(x) \leqslant \mu g(x)^T (x(\alpha) - x) \tag{9.51}$$

are satisfied for all α with

$$0 \leqslant \alpha \leqslant \frac{2(1-\mu)}{L}.$$

Proof. Using the Taylor's expansion theorem we can write

$$f(x(\alpha)) = f(x) + g(x)^T (x(\alpha) - x) - \int_0^1 \Big[(g(x - t(x - x(\alpha))) - g(x))^T (x - x(\alpha)) \Big]\, dt.$$

Taking into account (9.47) and (9.49) we transform it to the inequalities

$$\alpha [f(x) - f(x(\alpha))] \geqslant \|x - x(\alpha)\|^2 - \frac{\alpha L}{2} \|x - x(\alpha)\|^2 \tag{9.52}$$

and

$$f(x) - f(x(\alpha)) \geqslant g(x)^T (x - x(\alpha)) - \frac{L}{2}\|x - x(\alpha)\|^2. \tag{9.53}$$

From (9.52) we immediately have (9.50) while (9.53) and (9.47) lead to the relation

$$f(x) - f(x(\alpha)) \geqslant g(x)^T (x - x(\alpha)) - \frac{L\alpha}{2} g(x)^T (x - x(\alpha)).$$

If we impose the condition

$$g(x)^T (x - x(\alpha)) - \frac{L\alpha}{2} g(x)^T (x - x(\alpha)) \geqslant \mu g(x)^T (x - x(\alpha))$$

then, since $g(x)^T (x(\alpha) - x) < 0$, we have (9.51). □

On the basis of Lemma 9.6 we state the gradient projection algorithm.

Algorithm 9.2. (The gradient projection algorithm)

Parameters: $s > 0$, μ, $\beta \in (0,1)$.

1. Choose $x_1 \in \Omega$, set $k = 1$.
2. Find the smallest positive integer number m_k such that

$$f(x(\alpha_k)) - f(x_k) \leqslant \mu g_k^T (x(\alpha_k) - x_k) \tag{9.54}$$

   where $\alpha_k = \beta^{m_k} s$ and $x(\alpha_k) = \mathscr{P}_\Omega [x_k - \alpha_k g_k]$.
   If $x(\alpha_k) = x_k$ then STOP.
3. Substitute $x(\alpha_k)$ for x_{k+1}, increase k by one and go to Step 2.

Lemma 9.6 suggests that in Step 2 we could replace condition (9.54) by

$$f(x(\alpha_k)) - f(x_k) \leqslant -\mu \frac{\|x(\alpha_k) - x_k\|^2}{\alpha_k}$$

without affecting the convergence of the gradient projection algorithm.

Theorem 9.9. *Assume that g satisfies condition (9.49), Ω is a closed convex subset of $\mathscr{R}^n$ and $\{x_k\}$ is a sequence generated by* Algorithm 9.2. *Then every limit point of $\{x_k\}$ is a stationary point for problem (9.1).*

Proof. The sequence $\{f(x_k)\}$ is bounded and monotonically decreasing thus it is convergent. Therefore,

$$\lim_{k\to\infty} [f(x_{k+1}) - f(x_k)] = 0$$

and the line search rule (9.54) gives

$$\lim_{k\to\infty} g_k^T (x_{k+1} - x_k) = 0.$$

From Lemma 9.6 (relation (9.52)) we also have

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x_k\|^2}{\alpha_k} = 0. \tag{9.55}$$

Lemma 9.6 says that for any k

$$\alpha_k \geqslant \min\left[s, \frac{2(1-\mu)\beta}{L}\right]$$

which together with (9.55) guarantee that

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{9.56}$$

Suppose now that $\{x_k\}_{k\in K}$ is a subsequence converging to the point $\bar{x}$. For any x_k, $k \in K$, $y \in \Omega$, from Lemma 9.5, we have

$$\begin{aligned} g_k^T (x_k - y) &= g_k^T (x_{k+1} - y) + g_k^T (x_k - x_{k+1}) \\ &\leqslant \frac{1}{\alpha_k} (x_k - x_{k+1})^T (x_{k+1} - y) + g_k^T (x_k - x_{k+1}) \\ &\leqslant \|x_k - x_{k+1}\| \left\| \frac{x_{k+1} - y}{\alpha_k} + g_k \right\|. \end{aligned}$$

Due to (9.56) and since Ω and $\{x_k\}_{k\in K}$ are bounded we obtain

$$g(\bar{x})^T (\bar{x} - y) \leqslant 0$$

for all $y \in \Omega$ which, by part (i) of Theorem 9.1 means that $\bar{x}$ is a stationary point. □

The gradient projection algorithm has the property of identifying active constraints at a solution in a finite number of iterations. We prove that by showing that the sequence of the projected gradients $\{\nabla_\Omega f(x_k)\}$ converges to zero. The following theorem is due to Calamai and Moré [30].

Theorem 9.10. *Suppose that the assumptions of* Theorem 9.9 *are satisfied. Then*

$$\lim_{k\to\infty} \|\nabla_\Omega f(x_k)\| = 0.$$

Proof. First, we prove that

$$-\|\nabla_\Omega f(x)\| = \min\left[g(x)^T v : \ v \in \mathscr{T}(x), \ \|v\| \leqslant 1\right]. \tag{9.57}$$

We do that in two steps. In the first step we show that

$$-g(x)^T \nabla_\Omega f(x) = \|\nabla_\Omega f(x)\|^2. \tag{9.58}$$

Indeed, since $\mathscr{T}(x)$ is a cone, from (9.19) and the definition of $\nabla_\Omega f(x)$, for any $\lambda \geqslant 1$ we have

$$(\nabla_\Omega f(x) + g(x))^T ((\lambda - 1)\nabla_\Omega f(x)) \geqslant 0. \tag{9.59}$$

If we set $\lambda = 2$ and $\lambda = 0$ in (9.59), we obtain (9.58). In the second step we use (9.58) to establish (9.57). To this end take any $v \in \mathscr{T}(x)$ such that $\|v\| \leqslant \|\nabla_\Omega f(x)\|$. Then, from the definition of $\nabla_\Omega f(x)$, we have

$$\begin{aligned} \|\nabla_\Omega f(x) + g(x)\|^2 &\leqslant \|v + g(x)\|^2 \\ &\leqslant \|g(x)\|^2 + 2g(x)^T v + \|\nabla_\Omega f(x)\|^2. \end{aligned}$$

Evaluating the left-hand side of this inequality and taking into account (9.58) we obtain

$$-\|\nabla_\Omega f(x)\|^2 \leqslant g(x)^T v.$$

Since we have $\|v\| \leqslant \|\nabla_\Omega f(x)\|$, dividing both sides of this inequality by $\|\nabla_\Omega f(x)\|$ leads to (9.57).

From the above characterization of $\nabla_\Omega f(x_k)$ we have: for every $\varepsilon > 0$ there exists a vector $v_k \in \mathscr{T}(x_k)$ with $\|v_k\| \leqslant 1$ and such that

$$\|\nabla_\Omega f(x_k)\| \leqslant -g_k^T v_k + \varepsilon. \tag{9.60}$$

We know, from Lemma 9.5, that for any $z \in \Omega$ the following holds

$$\alpha_k g_k^T (z - x_{k+1}) \leqslant (x_{k+1} - x_k)^T (z - x_{k+1}). \tag{9.61}$$

Since v_{k+1} is a feasible direction $z_{k+1} = x_{k+1} + \tau_{k+1} v_{k+1} \in \Omega$. If we substitute z_{k+1} into (9.61) we obtain the estimate

$$\begin{aligned} \alpha_k g_k^T (z_{k+1} - x_{k+1}) &\leqslant (x_{k+1} - x_k)^T (z_{k+1} - x_{k+1}) \\ &\leqslant \|x_{k+1} - x_k\| \|z_{k+1} - x_{k+1}\|. \end{aligned} \tag{9.62}$$

We can easily deduct from the proof of Theorem 9.9 that

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x_k\|}{\alpha_k} = 0, \tag{9.63}$$

thus dividing both sides of inequality (9.62) by $\alpha_k \tau_{k+1}$ and by taking into account that $\|v_{k+1}\| \leqslant 1$ and $\tau_{k+1} > 0$ we come to the relation

$$\limsup_{k\to\infty} \left[-g_k^T v_{k+1}\right] \leqslant 0. \tag{9.64}$$

From the proof of Theorem 9.9 we also know that $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$ and (9.60), (9.64) imply that

$$\limsup_{k\to\infty} \|\nabla_{\Omega} f(x_{k+1})\| \leqslant \varepsilon. \tag{9.65}$$

$\varepsilon > 0$ is arbitrary thus (9.65) concludes the proof. □

The straightforward application of Theorem 9.8 leads to the conclusion that the gradient projection algorithm has the identifying properties.

Corollary 9.1. *Suppose that the assumptions of* Theorem 9.9 *hold. Furthermore, assume that Ω is a polyhedron. Then, for any subsequence $\{x_k\}_{k\in K}$ which converges to a nondegenerate stationary point $\bar{x}$, there exists a finite number k_0 such that*

$$\mathscr{A}(x_k) = \mathscr{A}(\bar{x})$$

for any $K \ni k \geqslant k_0$.

9.5 On the Stopping Criterion

Algorithm 9.2 stops if

$$x(\alpha_k) = x_k. \tag{9.66}$$

We know from Theorem 9.1 that (9.66) can be substituted by the condition

$$\mathscr{P}_{\Omega}[x_k - g_k] = x_k \tag{9.67}$$

therefore we could state a criterion for a finite iteration convergence as

$$\|x_k - \mathscr{P}_{\Omega}[x_k - \alpha g_k]\| \leqslant \varepsilon$$

for some small positive ε and any positive α provided that we can show that

$$\lim_{k\to\infty} \|x_k - \mathscr{P}_{\Omega}[x_k - \alpha g_k\| = 0. \tag{9.68}$$

It is interesting to note that even though $\|\nabla_{\Omega} f(x_k)\|$ converges to zero (according to Theorem 9.10) we cannot apply

$$\|\nabla_{\Omega} f(x_k)\| \leqslant \varepsilon \tag{9.69}$$

as a finite iteration stopping criterion. This is due to the fact that the mapping $\|\nabla_{\Omega} f(\cdot)\|$ is only lower semicontinuous as shown by Calamai and Moré [30].

Lemma 9.7. *If f is continuously differentiable on Ω then the mapping $\|\nabla_{\Omega} f(\cdot)\|$ is lower semicontinuous on Ω.*

Proof. Consider vector $x_k - z$ where z is an arbitrary element of Ω. Since $x_k \in \Omega$ and Ω is convex $z - x_k$ is a feasible direction at x_k. It implies, from (9.57) that

$$g_k^T (x_k - z) \leqslant \|\nabla_{\Omega} f(x_k)\| \|x_k - z\|,$$

thus

$$g(\tilde{x})^T(\tilde{x}-z) \leqslant \liminf_{k\to\infty} \|\nabla_\Omega f(x_k)\| \|\tilde{x}-z\| \tag{9.70}$$

where we have assumed that $\{x_k\}$ converges to $\tilde{x}$.

For any feasible direction v at $\tilde{x}$ we have that $\tilde{x}+\alpha v \in \Omega$ for some positive α. By setting $z=\tilde{x}+\alpha v$, from (9.70), we obtain

$$g(\tilde{x})^T v \leqslant \liminf_{k\to\infty} \|\nabla_\Omega f(x_k)\|$$

and eventually

$$\|\nabla_\Omega f(\tilde{x})\| \leqslant \liminf_{k\to\infty} \|\nabla_\Omega f(x_k)\| \tag{9.71}$$

because $\mathscr{T}(\tilde{x})$ is a closure of all feasible directions. Equation (9.71) holds for any convergent sequence $\{x_k\}$ thus the proof is completed. □

In Algorithm 9.2 we apply the stopping criterion $x(\alpha_k)=x_k$ which, when relaxed, could take the form

$$\|\mathscr{P}_\Omega[x_k-\alpha_k g_k]-x_k\| \leqslant \varepsilon.$$

This is a valid criterion since for any k we have $\alpha_k \geqslant \alpha_{min}$ with

$$\alpha_{min} = \min\left[s, \frac{2\beta(1-\mu)}{L}\right].$$

Moreover, according to the proof of Theorem 9.9

$$\lim_{k\to\infty} \|\mathscr{P}_\Omega[x_k-\alpha_k g_k]-x_k\| = 0.$$

Since $s \geqslant \alpha_k \geqslant \alpha_{min}$ we can also write

$$\lim_{k\to\infty} \frac{\|\mathscr{P}_\Omega[x_k-\alpha_k g_k]-x_k\|}{\alpha_k} = 0. \tag{9.72}$$

We use (9.72) to prove that (9.68) holds with $\alpha=1$. The proof heavily relies on the following lemma.

Lemma 9.8. *If $\mathscr{P}$ is the projection into Ω then*

(i) $\mathscr{P}$ is a monotone operator, that is,

$$(\mathscr{P}[x]-\mathscr{P}[y])^T(x-y) \geqslant 0 \tag{9.73}$$

for any $x, y \in \mathscr{R}^n$ and the strict inequality holds if $\mathscr{P}[x] \neq \mathscr{P}[y]$.

(ii) The function ϕ_1 defined by

$$\phi_1(\alpha) = \|\mathscr{P}[x+\alpha d]-x\|,\ \alpha>0$$

is isotone (nondecreasing) for all $x \in \mathscr{R}^n$ and $d \in \mathscr{R}^n$.

(iii) the function ϕ_2 defined by

$$\phi_2(\alpha) = \frac{\|\mathscr{P}[x+\alpha d]-x\|}{\alpha},\ \alpha > 0$$

is antitone (nonincreasing) for all $x \in \mathscr{R}^n$ and $d \in \mathscr{R}^n$.

Proof. *(i).* Substituting $\mathscr{P}[y]$ for y in (9.19) gives

$$(x-\mathscr{P}[x])^T(\mathscr{P}[x]-\mathscr{P}[y]) \geqslant 0.$$

Replacing y with x and again using (9.19) results in the inequality

$$(y-\mathscr{P}[y])^T(\mathscr{P}[y]-\mathscr{P}[x]) \geqslant 0$$

and adding both inequalities provides

$$(x-\mathscr{P}[x]-y+\mathscr{P}[y])^T(\mathscr{P}[x]-\mathscr{P}[y]) \geqslant 0.$$

This can be equivalently stated as

$$\|\mathscr{P}[x]-\mathscr{P}[y]\|^2 \leqslant (x-y)^T(\mathscr{P}[x]-\mathscr{P}[y])$$

from which (9.73) follows.

(ii). The proof we give is due to Toint [205]. The proof relies on (9.12) and the monotonicity of $\mathscr{P}$. The rest are algebraic manipulations. We have to show that if $\alpha_2 > \alpha_1$ then $\phi_1(\alpha_2) \geqslant \phi_1(\alpha_1)$. Consider

$$\begin{aligned}\|\mathscr{P}[x+\alpha_2 d]-x\|^2 &= \|\mathscr{P}[x+\alpha_1 d]-x\|^2 + \|\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d]\|^2 + \\ &\quad 2(\mathscr{P}[x+\alpha_1 d]-x)^T(\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d]) \\ &\geqslant \|\mathscr{P}[x+\alpha_1 d]-x\|^2 + \\ &\quad 2(\mathscr{P}[x+\alpha_1 d]-(x+\alpha_1 d))^T(\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d]) \\ &\quad +2\alpha_1 d^T(\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d]).\end{aligned}$$

Since $\mathscr{P}[x+\alpha_2 d] \in \Omega$ (9.12) implies that the first inner product in the last right-hand side is nonnegative. In order to complete the proof of part (ii) we have to show that the other inner product is also nonnegative. To this end observe that

$$\begin{aligned}&\alpha_1 d^T(\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d]) = \\ &\frac{\alpha_1}{\alpha_2-\alpha_1}((x+\alpha_2 d)-(x+\alpha_1 d))^T(\mathscr{P}[x+\alpha_2 d]-\mathscr{P}[x+\alpha_1 d])\end{aligned} \tag{9.74}$$

and since $\mathscr{P}$ is a monotone operator the left-hand side of (9.74) is non-negative.

(iii). We follow the proof given in [30]. We start from the inequality

$$\frac{\|u\|}{\|v\|} \leqslant \frac{u^T (u-v)}{v^T (u-v)} \tag{9.75}$$

which is valid under the assumption that $v^T (u-v) > 0$. To prove (9.75) notice that for any $u \in \mathscr{R}^n$, $v \in \mathscr{R}^n$ we have

$$u^T v \leqslant \|u\| \|v\|. \tag{9.76}$$

After multiplying both sides of inequality (9.76) by $\|u\| + \|v\|$ and then by regrouping the terms we obtain

$$\|v\| \left(\|u\|^2 - u^T v\right) \geqslant \|u\| \left(u^T v - \|v\|^2\right)$$

which under the assumption $v^T (u-v) > 0$ gives (9.75).

Consider now

$$u = \mathscr{P}[x + \alpha d] - x,\ v = \mathscr{P}[x + \beta d] - x$$

with $\alpha > \beta > 0$. From Lemma 9.1 (part *(i)*) we have

$$(x + \alpha d - \mathscr{P}[x + \alpha d])^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d]) \geqslant 0$$

which is equivalent to

$$u^T (u-v) \leqslant \alpha d^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d]). \tag{9.77}$$

In the same way we can show that

$$v^T (u-v) \geqslant \beta d^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d]). \tag{9.78}$$

Furthermore, part *(i)*, since $\alpha > \beta$, gives

$$d^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d]) > 0$$

which means that we can apply the inequality (9.75) that together with (9.77)–(9.78) give

$$\frac{\|\mathscr{P}[x + \alpha d]\|}{\|\mathscr{P}[x + \beta d]\|} \leqslant \frac{\alpha d^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d])}{\beta d^T (\mathscr{P}[x + \alpha d] - \mathscr{P}[x + \beta d])}$$

and part *(iii)* follows. □

Theorem 9.11. *Suppose that the assumptions of* Theorem 9.9 *are satisfied. Then*

$$\lim_{k \to \infty} \|\mathscr{P}_\Omega [x_k - g_k] - x_k\| = 0. \tag{9.79}$$

Proof. Due to the generalized Armijo rule we always have $\alpha_k \leqslant s$. Suppose that $s \leqslant 1$, then using part (iii) of Lemma 9.8 we can write

$$\frac{\|\mathscr{P}_\Omega [x_k - \alpha_k g_k] - x_k\|}{\alpha_k} \geqslant \|\mathscr{P}_\Omega [x_k - g_k] - x_k\|$$

and, due to (9.72), we have the thesis.

Assume now that $s \geqslant 1$. As in the other case we can prove that

$$\frac{\|\mathscr{P}_\Omega [x_k - \alpha_k g_k] - x_k\|}{\alpha_k} \geqslant \frac{\|\mathscr{P}_\Omega [x_k - s g_k] - x_k\|}{s}$$

which, from (9.55), implies that

$$\lim_{k \to \infty} \|\mathscr{P}_\Omega [x_k - s g_k] - x_k\| = 0. \tag{9.80}$$

In order to complete the proof we refer to part (ii) of Lemma 9.8 which says that

$$\|\mathscr{P}_\Omega [x_k - s g_k] - x_k\| \geqslant \|\mathscr{P}_\Omega [x_k - g_k] - x_k\|.$$

and (9.79) follows from (9.80). □

In contrast to

$$\|x(\alpha_k) - x_k\| \leqslant \varepsilon$$

both stopping criteria (9.67) and (9.69) require the calculation of the projected vector. Except the sets of simple structures the evaluation of the projection is an elaborate process. For example, consider the set

$$\Omega = \left\{x \in \mathscr{R}^n : \; C^T x = d\right\}$$

where $C \in \mathscr{R}^{n \times m}$ and $d \in \mathscr{R}^m$. Then,

$$\nabla_\Omega f(x) = -Z \left(Z^T Z\right)^{-1} Z^T g(x)$$

with Z being the matrix whose columns span the null space of C^T (cf. Appendix B). Therefore, in this case, a factorization of C has to be made.

Consider now the set Ω defined by simple constraints

$$\Omega = \{x \in \mathscr{R}^n : \; l \leqslant x \leqslant u\} \tag{9.81}$$

where $l, \; u \in \mathscr{R}^n$ and the inequalities in (9.81) are defined componentwise. In this case we can show that

$$-\left[\nabla_\Omega f(x)\right]_i = \begin{cases} (g(x))_i & \text{if } x_i \in (l_i, u_i) \\ \min\left[(g(x))_i, 0\right] & \text{if } x_i = l_i \\ \max\left[(g(x))_i, 0\right] & \text{if } x_i = u_i \end{cases} \tag{9.82}$$

provided that $l_i < u_i$ with $[\nabla_\Omega f(x)]_i = 0$ in the exceptional case when $l_i = u_i$. Thus, the calculation of the projected gradient is achieved at low cost.

Similarly, the point $x(1)$ can also be found at low expense according to the formula

$$(x(1))_i = \begin{cases} (x)_i - (g(x))_i & \text{if } (x)_i - (g(x))_i \in (l_i, u_i) \\ l_i & \text{if } (x)_i - (g(x))_i \leqslant l_i \\ u_i & \text{if } (x)_i - (g(x))_i \geqslant u_i \end{cases}.$$

Notice that $\|\nabla_\Omega f(\cdot)\|$ defined through (9.82) is a lower semicontinuous function so using it in the stopping criterion can lead to unnecessary many iterations required to achieved a prespecified tolerance. However, if $\{x_k\}$ converges to a nondegenerate stationary point then the set of active constraints is identified after a finite number of iterations and the mapping $\|\nabla_\Omega f(\cdot)\|$ is continuous in the neighborhood of the stationary point.

On the other hand notice that the mapping

$$\nu(x; f) = \|\nabla_\Omega f(x)\|$$

is scale invariant in the sense that $\nu(x; \alpha f) = \alpha \nu(x; f)$. The other stopping criterion mapping

$$\mu(x; f) = \|\mathscr{P}_\Omega [x - g(x)] - x\|$$

does not have this property.

9.6 Notes

Our presentation of numerical methods for optimization problems over a polyhedron is based mainly on [30], [23], [59], [25], [22], [24] and all these papers can be regarded as the extension of the results by Bertsekas [8] which on the other hand continues the work of [122].

Our reference papers extend considerably the material in this chapter. For example [24] analyzes second order sufficient optimality conditions. The following theorem is proved in [24].

Theorem 9.12. *Suppose that f is continuously differentiable on a polyhedron Ω and twice differentiable at the point $\bar{x}$. If $\bar{x}$ is a stationary point of problem (9.1)–(9.2), then*

$$\left\{ w \in \mathscr{T}(\bar{x}) : \; g(\bar{x})^T w = 0 \right\} = \operatorname{cone}\left(\mathscr{E}(-g(\bar{x})) - \bar{x}\right).$$

If

$$w \in \operatorname{cone}\left(\mathscr{E}(-g(\bar{x})) - \bar{x}\right), \; w \neq 0 \Rightarrow w^T \nabla^2 f(\bar{x}) w > 0.$$

then $\bar{x}$ is an isolated stationary point of f.

If Ω is a convex set and

$$g(\bar{x})^T w = 0,\ w \in \mathscr{T}(\bar{x}),\ w \neq 0 \Rightarrow w^T \nabla^2 f(\bar{x}) w > 0$$

then $\bar{x}$ is an isolated stationary point of f.

Here, cone(A) is the cone spanned by the set A, that is, the set of vectors αw for some $\alpha \geqslant 0$ and $w \in A$.

Burke and Moré consider in [23] the relationship between $\nabla_\Omega f$ and the Clarke subdifferential $\partial_C \psi$ of the composite mapping

$$\psi(x) = f(\mathscr{P}_\Omega[x]).$$

First, notice that under the assumption that f is continuously differentiable then ψ is locally Lipschitz on $\mathscr{R}^n$. Secondly, problem (9.1) with Ω as an arbitrary closed convex set can be transformed to the unconstrained problem

$$\min_{x \in \mathscr{R}^n} \psi(x).$$

The main result stated in [23], as far as ψ is concerned, is as follows.

Theorem 9.13. *Suppose that f is continuously differentiable over a nonempty closed convex set Ω. Then*

$$\operatorname{dist}(0, \partial_C \psi(x)) \leqslant \|\nabla_\Omega f(x)\|.$$

Here, $\operatorname{dist}(x,A) = \inf\{\|x - y\| :\ y \in A\}$. Theorem 9.13 says that Algorithm 9.2 generates a sequence $\{x_k\}$ such that

$$\operatorname{dist}(0, \partial_C \psi(x_{k_l})) \to_{k_l \to \infty} 0$$

for any convergent subsequence $\{x_{k_l}\}$ provided that the assumptions of Theorem 9.9 are satisfied.

In [23] the extension of the gradient projection algorithm to a sequential quadratic programming method is analyzed while in [30] the adoption of Algorithm 9.2 to a quadratic problem is considered – the convergence in a finite number of iterations is established.

Chapter 10
Conjugate Gradient Algorithms for Problems with Box Constraints

10.1 Introduction

In the chapter we consider the problem

$$\min_{x \in \mathscr{R}^n} f(x) \tag{10.1}$$

subject to the simple bounds

$$l \leqslant x \leqslant u, \tag{10.2}$$

where we assume that l, u are fixed vectors and the inequalities are taken componentwise. It is the special case of the problem considered in the previous chapter if we notice that the set $\Omega = \{x \in \mathscr{R}^n : \ l \leqslant x \leqslant u\}$ is a polyhedron.

The method that we introduce in this chapter is an extension of the conjugate gradient algorithm presented in Chap. 7. It can also be considered as the extension of the projection gradient algorithm discussed in the previous chapter. The main idea behind the method proposed in this chapter is to use the projection gradient algorithm with respect to variables which are supposed to be active at the next iteration. For the other variables we perform an iterate of an unconstrained optimization algorithm.

If d_k is a direction generated at the kth iteration of the projection algorithm, for example $d_k = -g_k$, then the next point is determined by finding α_k such that for $\mathscr{P}[x_k + \alpha_k d_k]$

$$f(\mathscr{P}[x_k + \alpha_k d_k]) - f(x_k) \leqslant \sigma(x_k) < 0,$$

where $\sigma(\cdot)$ is such that

$$\sigma(x) < 0, \text{ for all noncritical points } x,$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

and

$$\lim_{k\to\infty} \sigma(x_k) = 0, \text{ if } x_k \to \bar{x},$$

where $\bar{x}$ is a critical point of problem (10.1)–(10.2).

10.2 General Convergence Theory

The main part of our analysis concerns directional minimization rules. In Chap. 9 we use only the Armijo line search rule. Since we want to modify the projection gradient algorithm to adopt in it conjugate gradient algorithm iterates the first step we have to do is to introduce new line search rules which would mimic the Wolfe conditions.

To this end and in order to simplify the presentation we consider the problem of minimizing f subject to the constraints

$$x \geqslant 0. \tag{10.3}$$

The more general case can be easily obtained by straightforward generalization of results for the problem with constraints (10.3).

The analog of a conjugate gradient algorithm for problem (10.1)–(10.2) could take the form

$$x_{k+1} = \mathscr{P}[x_k + \alpha_k d_k],$$

where d_k is given by

$$d_k = -\mathbf{Nr}\{g_k, -\beta_k d_{k-1}\}$$

and $\alpha_k > 0$ is a solution to the directional minimization

$$\min_{\alpha>0} f(\mathscr{P}[x_k + \alpha d_k]). \tag{10.4}$$

The rule (10.4) is impractical and the first thing in defining the algorithm is to introduce the proper directional minimization rule. Bertsekas used the generalized Armijo introduced in Sect. 9.4. In the case of our conjugate gradient algorithm that rule could be stated as

$$f(\mathscr{P}[x_k + \beta^{m_k} d_k]) - f(x_k) \leqslant -\mu d_k^T (\mathscr{P}[x_k + \beta^{m_k} d_k] - x_k), \tag{10.5}$$

where $\beta \in (0,1)$ and m_k is the smallest integer number from the set $\{0,1,\ldots\}$ which guarantees (10.5).

The scheme (10.5) for the directional minimization would not result in an efficient algorithm because it is well known that some approximation of the exact minimization (10.4) is desired for the fast convergence of a conjugate gradient algorithm.

In order to define a new directional minimization rule we look at the function $\phi_k(\alpha) = f(\mathscr{P}[x_k + \alpha d_k])$ as a nondifferentiable function. $\phi(\alpha)$ is a composition of two functions: the first one is Lipschitzian and the second one continuously differentiable. In order to show that consider a piecewise linear *arc* of the form

$$x_k(\alpha) = x_k + d_k(\alpha), \quad \text{where } (d_k(\alpha))_i = \begin{cases} \alpha(d_k)_i & \text{if } \alpha \leqslant \alpha_k^i, \\ \alpha_k^i(d_k)_i & \text{if } \alpha > \alpha_k^i, \end{cases} \tag{10.6}$$

and the breakpoints $\{\alpha_k^i\}_1^n$ are calculated as follows

$$\alpha_k^i = -\frac{(x_k)_i}{(d_k)_i}, \quad i = 1, \ldots, n \tag{10.7}$$

(assuming that if $(d_k)_i \geqslant 0$ then $\alpha_k^i = \infty$). The points α_k^i, when ordered, define *bent* points of $x_k(\alpha)$: $0 \leqslant t_k^1 < t_k^2 < \cdots < t_k^m$. The function $\phi_k(\alpha) = f(\mathscr{P}[x_k + \alpha d_k]) = f(x_k + d_k(\alpha))$ is differentiable at all $\alpha \geqslant 0$ but $\{t_k^i\}_1^m$. If we denote by $f_{k,i}^{\alpha}(\beta)$ the function $\beta \to f((x_k(\alpha))_1, \ldots, (x_k(\beta))_i, \ldots, (x_k(\alpha))_n)/(d_k)_i$, then we can write

$$\nabla f_{k,i}^{\alpha}(\alpha) = \begin{cases} (g(x_k + d_k(\alpha)))_i & \text{if } \alpha < \alpha_k^i \\ 0 & \text{if } \alpha > \alpha_k^i \end{cases}, \quad i = 1, \ldots, n. \tag{10.8}$$

At the breakpoint α_k^i $f_{k,i}^{\alpha}(\cdot)$ is not differentiable and the Clarke's generalized gradient (cf. Appendix A) is expressed by the formula

$$\partial f_{k,i}^{\alpha_k^i}(\alpha_k^i) = \operatorname{co}\left(\left\{\left(g(x_k + d_k(\alpha_k^i))\right)_i, 0\right\}\right).$$

In this case we have freedom in choosing some vectors which could emulate the role of gradients in the directional minimization. We apply the following convention

$$(g_k(\alpha))_i = \begin{cases} (g(x_k + d_k(\alpha)))_i & \text{if } \alpha < \alpha_k^i \\ 0 & \text{if } \alpha \geqslant \alpha_k^i \end{cases}, \quad i = 1, \ldots, n. \tag{10.9}$$

Notice that according to the used so far notation $g_k = g_k(0)$.

Therefore, an *ad hoc* directional minimization could be as follows.

Ad hoc rule: find a positive number α_k such that

$$f(\mathscr{P}[x_k + \alpha_k d_k]) - f(x_k) \leqslant -\mu d_k^T (\mathscr{P}[x_k + \alpha_k d_k] - x_k), \tag{10.10}$$

$$g_k(\alpha_k)^T d_k \geqslant -\eta \|d_k\|^2, \quad 0 < \mu < \eta < 1. \tag{10.11}$$

Obviously if constraints (10.2) are not present then ad hoc rule reduces to the rule that is discussed in Chap. 2: find a positive number α_k such that

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant -\mu \alpha_k \|d_k\|^2,$$
$$g(x_k + \alpha_k d_k)^T d_k \geqslant -\eta \|d_k\|^2, \quad 0 < \mu < \eta < 1.$$

The directional minimization rules (10.10)–(10.11) have one major drawback – the function $\phi_k(\alpha)$ can have bent points close to zero. In general $\phi_k(\alpha)$ is nondifferentiable at a finite number of points but tractability of analytical properties of $\phi_k(\alpha)$ for α close to zero is crucial in deriving globally convergent algorithms.

To overcome the drawback of ad hoc rule we make two modifications. First of all, if $(x_k)_i$ is close to 0 and $(g_k)_i > 0$ then we set $(d_k)_i = -(g_k)_i$. We know from Lemma 9.6 that if $d_k = -g_k$ and x_k is not a critical point then for sufficiently small values of α_k condition (10.10) is satisfied. If $(x_k)_i$ is close to 0, $(g_k)_i \leqslant 0$ and $(x_{k+1})_i = 0$ then we must have $(d_k)_i \leqslant 0$ which means that $(g_k)_i(d_k)_i \geqslant 0$ does not contribute to the "descent property" of d_k. Thus we can expect that other components of d_k will guarantee sufficient decrease of f. This approach was exploited in [10]. That method of determining d_k has another advantage. If the strict complementarity condition is satisfied at a solution $\bar{x}$, i.e. $(g(\bar{x}))_i > 0$ for all i such that $\bar{x}_i = 0$, then these constraints are likely to be identified in a finite number of iterations if, instead of using $(d_k)_i$ (whose sign does not have to be related to the sign of $(g_k)_i$), we use $-(g_k)_i$.

To implement these modifications we define the set of indices $I_k^+ \subset \{1,\ldots,n\}$

$$I_k^+ = \{i:\ (x_k)_i \leqslant \varepsilon_k \quad \text{and} \quad (g_k)_i > 0\}, \tag{10.12}$$

where $\{\varepsilon_k\}$ is such that $\varepsilon_k > 0$ and

$$\lim_{k\in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0 \Leftrightarrow \lim_{k\in K} \varepsilon_k = 0. \tag{10.13}$$

for any subsequence $\{x_k\}_{k\in K}$.

In [10] it is proposed to use $\{\varepsilon_k\}$ defined in the following way

$$w_k = x_k - \mathscr{P}[x_k - Dg_k],\ \ \varepsilon_k = \min(\varepsilon, \|w_k\|),\ \varepsilon > 0, \tag{10.14}$$

where D is a diagonal positive definite matrix. One can easily show that condition (10.13) is satisfied for those ε_k.

The sets I_k^+ are used to modify the direction finding subproblem. Instead of solving problem (1.5) we find a new direction according to the rule

$$d_k = -\mathbf{Nr}\{g_k, -\beta_k d_{k-1}^+\}. \tag{10.15}$$

Here, d_{k-1}^+ is defined by

$$(d_{k-1}^+)_i = \begin{cases} (d_{k-1})_i & \text{if } i \notin I_k^+, \\ -(g_k)_i/\beta_k & \text{if } i \in I_k^+. \end{cases} \tag{10.16}$$

Let us notice that according to (10.15) and (10.16) we have

$$(d_k)_i = -(g_k)_i \quad \text{for } i \in I_k^+. \tag{10.17}$$

The second modification takes into account the fact that in order to find α_k we can minimize $f(\mathscr{P}[x_k + \alpha_k d_k])$ on the subintervals $[t_k^l, t_k^{l+1})$, $l = 1,\ldots,m-1$. To this

end we introduce, for a given α, the set of all indices such that $i \notin I_k^+$, or $i \in I_k^+$ and $\alpha_k^i > \alpha$:

$$I_k(\alpha) = \{i:\ i \notin I_k^+\} \bigcup \{i:\ i \in I_k^+ \text{ and } \alpha_k^i > \alpha\}. \tag{10.18}$$

Having defined $I_k(\alpha)$ we can partition d_k into two vectors in such a way that one of them, $d_k^-(\alpha)$, is as follows

$$d_k^-(\alpha) = \{(d_k)_i\}_{i \in I_k(\alpha)}.$$

Eventually we can state our new directional minimization rule which we call the *generalized Wolfe rule*: find a positive number α_k such that

$$\begin{aligned} f(\mathscr{P}[x_k + \alpha_k d_k]) - f(x_k) &\leqslant -\mu d_k^T (\mathscr{P}[x_k + \alpha_k d_k] - x_k) \\ g_k(\alpha_k)^T d_k &\geqslant -\eta \|d_k^-(\alpha_k)\|^2, \end{aligned} \tag{10.19}$$

where $0 < \mu < \eta < 1$.

Let us notice that condition (10.19) uses the vector $d_k^-(\alpha)$ instead of the vector which is defined by d_k and all indices which are not active at $x_k + d_k(\alpha)$ (in other words we do not set to zero components of d_k which become active at α and which indices do not belong to I_k^+). This gives us less tight directional minimization rule.

We can state our algorithm as follows.

Algorithm 10.1. (The conjugate gradient algorithm for problems with box constraints)

Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu,\ \varepsilon > 0,\ \{\beta_k\}$, diagonal positive definite matrix D.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute

$$d_1 = -g_1$$

and set $k = 1$.

2. Find a positive number α_k such that

$$f(\mathscr{P}[x_k + \alpha_k d_k]) - f(x_k) \leqslant -\mu d_k^T (\mathscr{P}[x_k + \alpha_k d_k] - x_k) \tag{10.20}$$

$$g_k(\alpha_k)^T d_k \geqslant -\eta \|d_k^-(\alpha_k)\|^2. \tag{10.21}$$

3. Compute

$$w_{k+1} = x_{k+1} - \mathscr{P}[x_{k+1} - D g_{k+1}]. \tag{10.22}$$

If $\|w_{k+1}\| = 0$ then STOP, otherwise calculate

$$\varepsilon_{k+1} = \min(\varepsilon, \|w_{k+1}\|) \tag{10.23}$$

$$d_{k+1} = -\mathbf{Nr}\{g_{k+1}, -\beta_{k+1} d_k^+\}. \tag{10.24}$$

4. Substitute $\mathscr{P}[x_k + \alpha_k d_k]$ for x_{k+1}, increase k by one, go to Step 2.

The rule (10.14) for ε_k can be modified without affecting global convergence properties of Algorithm 10.1. For example we can apply

$$\varepsilon_k = \min(\varepsilon_1, \varepsilon_2 \|w_k\|), \tag{10.25}$$

where $\varepsilon_1, \varepsilon_2 > 0$. We can also use the scalar ε_k defined in the following way

$$\varepsilon_k = \min(\varepsilon_1, \varepsilon_2 \|w_k\|_\infty), \tag{10.26}$$

where $\|\cdot\|_\infty$ is the Chebyshev norm. If $\varepsilon_2 \geqslant 1$ then Theorem 10.3 is valid for both (10.25) and (10.26).

We can prove the following lemma.

Lemma 10.1. *Assume that x_k is a noncritical point and $d_k \neq 0$ is calculated in* Step 3 *of* Algorithm 10.1. *Then there exists a positive α_k such that (10.20)–(10.21) are satisfied, or*

$$\lim_{\alpha \to \infty} f(\mathscr{P}[x_k + \alpha d_k]) = -\infty. \tag{10.27}$$

Proof. First suppose that zero is a bent point. From (7.14) we have

$$g_k^T d_k \leqslant -\|d_k\|^2.$$

Let $(x_k)_i = 0$. If $i \in I_k^+$ then

$$(g_k)_i (d_k)_i = -(d_k)_i^2,$$

by (10.17), and i determines the zero bent point. Then for the modified vector $\hat{d}_k$ obtained by setting $(d_k)_i = 0$ we have

$$g_k^T \hat{d}_k \leqslant -\|\hat{d}_k\|^2. \tag{10.28}$$

If $i \notin I_k^+$ then $(g_k)_i \leqslant 0$, by (10.12), so that $(d_k)_i < 0$ implies

$$(g_k)_i (d_k)_i \geqslant 0$$

and the modified vector $\hat{d}_k$ will also satisfy (10.28). Therefore, $\hat{d}_k$ is a direction of descent.

Suppose now that t_k^1 is the first nonzero bent point along the direction d_k (so assume that $\hat{d}_k = d_k$). It means that on the subinterval $[0, t_k^1)$ $f(\mathscr{P}[x_k + \alpha d_k])$ is a differentiable function. Therefore, we can apply the procedure described in Lemma 2.4 to find $\alpha_k \in [0, t_k^1)$ (if it exists) which satisfies (10.20)–(10.21).

If conditions (10.20)–(10.21) are not satisfied on the subinterval $[0, t_k^1)$ the directional minimization is continued on the next subinterval $[t_k^1, t_k^2)$ and only these variables vary which have not hit constraints. On the subinterval the directional

minimization rules can be stated as follows

$$f(\mathscr{P}[x_k+\alpha d_k])-f(x_k)\leqslant -\mu\left[\sum_{i\notin T_k^1}\alpha(d_k)_i^2+\sum_{i\in T_k^1}t_k^1(d_k)_i^2\right]$$
$$g_k(\alpha)^T d_k\geqslant -\eta\|d_k^-(\alpha)\|^2,\ \alpha\in[t_k^1,t_k^2),$$

where T_k^1 is a set of indices of constraints which have become active at $\alpha=t_k^1$.

We also have

$$\sum_{i\notin T_k^1}\left(g(x_k+t_k^1 d_k)\right)_i(d_k)_i<-\eta\|d_k^-(t_k^1)\|^2$$

because otherwise we would stop at t_k^1. Then, either the point α_k lies in the current subinterval, or we move to the next subinterval $[t_k^2,t_k^3)$ and repeat the same procedure. If α_k is not located in any of the subintervals $[t_k^i,t_k^{i+1})$, $[t_k^m,\infty)$ then (10.27) holds. □

A procedure which finds α_k satisfying (10.20)–(10.21) in a finite number of operations can be easily constructed following procedures described in Lemma 2.4.

The procedure described in Lemma 2.4 generates a sequence of points $\{\alpha^l\}_1^\infty$ constructed in the following way.

If

$$f(\mathscr{P}[x_k+\alpha^l d_k])-f(x_k)\leqslant -(\mu+\varepsilon)d_k^T\left(\mathscr{P}[x_k+\alpha^l d_k]-x_k\right)$$

($\eta>\mu+\varepsilon$, $\varepsilon>0$) we set $\alpha_\mu^{l+1}=\alpha^l$, $\alpha_N^{l+1}=\alpha_N^l$ and $\alpha_\mu^{l+1}=\alpha_\mu^l$, $\alpha_N^{l+1}=\alpha^l$ otherwise ($\alpha_\mu^0=0$, $\alpha_N^0=\infty$). Next, we substitute α^{l+1} by $(\alpha_\mu^l+\alpha_N^l)/2$ if $\alpha_N^{l+1}<\infty$, or by $2\alpha^l$ if $\alpha_N^{l+1}=\infty$.

We can see from the proof of Lemma 2.4 that in order to justify our claim that this procedure finds α_k fulfilling (10.20)–(10.21) in a finite number of iterations we have to analyze carefully the case when

$$\lim_{l\to\infty}\alpha_\mu^l=\lim_{l\to\infty}\alpha_N^l=\hat{\alpha}$$

and $\hat{\alpha}$ is a bent point.

Therefore, let us assume that $\hat{\alpha}$ is a bent point. We have

$$f(\mathscr{P}[x_k+\alpha_N^l d_k])-f(x_k)>-(\mu+\varepsilon)d_k^T\left(\mathscr{P}[x_k+\alpha_N^l d_k]-x_k\right)\quad(10.29)$$
$$f(\mathscr{P}[x_k+\alpha_\mu^l d_k])-f(x_k)\leqslant -(\mu+\varepsilon)d_k^T\left(\mathscr{P}[x_k+\alpha_\mu^l d_k]-x_k\right).\quad(10.30)$$

If infinitely often

$$g_k(\alpha_N^l)^T d_k\geqslant -\eta\|d_k^-(\alpha_N^l)\|^2\quad(10.31)$$

then, because $f(\mathscr{P}[x_k+\alpha d_k])$ is continuous, $\varepsilon>0$, after a finite number of iterations both (10.20) and (10.21) will be satisfied.

To prove (10.31) we consider the inequality

$$f(\mathscr{P}[x_k+\hat{\alpha}d_k])-f(x_k)\leqslant-(\mu+\varepsilon)d_k^T\,(\mathscr{P}[x_k+\hat{\alpha}d_k]-x_k)\,, \tag{10.32}$$

which follows from (10.30). Equations (10.32) and (10.29) imply that

$$\begin{aligned} f(\mathscr{P}[x_k+\alpha_N^l d_k])-f(\mathscr{P}[x_k+\hat{\alpha}d_k]) &> -(\mu+\varepsilon)d_k^T\Big(\mathscr{P}[x_k+\alpha_N^l d_k] \\ &\quad -\mathscr{P}[x_k+\hat{\alpha}d_k]\Big) \\ &= -(\mu+\varepsilon)(\alpha_N^l-\hat{\alpha})\|\hat{d}_k\|^2 \end{aligned} \tag{10.33}$$

for sufficiently large l. Here $\hat{d}_k$ is the vector d_k modified in such a way that if $(x_k(\hat{\alpha}))_i=0$ (see (10.6)) then $(\hat{d}_k)_i=0$. On the other hand, since f is continuously differentiable on the right of $\hat{\alpha}$ (for sufficiently large l), we have

$$f(\mathscr{P}[x_k+\alpha_N^l d_k])-f(\mathscr{P}[x_k+\hat{\alpha}d_k])=(\alpha_N^l-\hat{\alpha})g_k(\alpha_N^l)^T d_k+o(\alpha_N^l-\hat{\alpha}).$$

(Here, $o(\alpha)$ is a higher order term with respect to α). Thus, from (10.33),

$$g_k(\alpha_N^l)^T d_k+\frac{o(\alpha_N^l-\hat{\alpha})}{\alpha_N^l-\hat{\alpha}}>-(\mu+\varepsilon)\|\hat{d}_k\|^2>-\eta\,\|d_k^-(\alpha_N^l)\|^2,$$

for sufficiently large l, where the last inequality follows from the fact that $\|\hat{d}_k\|\leqslant\|d_k^-(\alpha)\|$ due to the definition of $\hat{d}_k$ and $d_k^-(\alpha)$. Therefore (10.31) must happen.

To investigate the convergence of Algorithm 10.1 we begin by providing the following crucial lemma.

Lemma 10.2. *If g is locally Lipschitz continuous with constant L, the direction d_k is determined by (10.22) and the step-size coefficient α_k satisfies (10.20)–(10.21), then for any bounded subsequence $\{x_k\}_{k\in K}$*

$$\lim_{k\in K}\|x_k-\mathscr{P}[x_k+d_k]\|=0. \tag{10.34}$$

Proof. Because $\{f(x_k)\}_{k\in K}$ is non-increasing and bounded from below (because $\{x_k\}_{k\in K}$ is bounded), it has to be convergent. From (10.20) and the theorem on three sequences we have

$$\begin{aligned} \lim_{k\in K}[f(x_{k+1})-f(x_k)]&=0 \\ \lim_{k\in K}\mu\,d_k^T\,(\mathscr{P}[x_k+\alpha_k d_k]-x_k)&=0\,. \end{aligned} \tag{10.35}$$

Because $\{d_k\}_{k\in K}$ is bounded, namely

$$\|d_k\|\leqslant\|g_k\|,\quad\forall k\in K,$$

by (10.15), the sequence $\{x_k - \mathscr{P}[x_k + d_k]\}_{k\in K}$ is also bounded. Assume that (10.34) does not hold. It means that there exists a subsequence of $\{x_k - \mathscr{P}[x_k + d_k]\}_{k\in K}$ which is convergent and its limit does not satisfy (10.34). Without the loss of generality we can assume that the sequence $\{x_k - \mathscr{P}[x_k + d_k]\}_{k\in K}$ itself does not converge to zero. Furthermore, because $\{x_k\}_{k\in K}$, $\{d_k\}_{k\in K}$ are bounded, we can also extract their convergent subsequences. We also assume the following (again without the loss of generality):

$$\lim_{k\in K} x_k = \bar{x}$$
$$\lim_{k\in K} d_k = \bar{d}.$$

Thus a contradiction to (10.34) implies that there exists at least one j such that

$$(\bar{x})_j > 0 \quad \text{and} \quad (\bar{d})_j \neq 0, \tag{10.36}$$

or

$$(\bar{x})_j = 0 \quad \text{and} \quad (\bar{d})_j > 0. \tag{10.37}$$

For any index i we have

$$(d_k)_i(\mathscr{P}[x_k + \alpha_k d_k] - x_k)_i = \begin{cases} \alpha_k (d_k)_i^2 & \text{if } \alpha_k < \alpha_k^i \\ -(x_k)_i (d_k)_i & \text{if } \alpha_k \geqslant \alpha_k^i \end{cases} \geqslant 0 \tag{10.38}$$

since $(d_k)_i \leqslant 0$ if $\alpha_k \geqslant \alpha_k^i$.

Therefore, from (10.35), for each $i \in \{1,\ldots,n\}$ we have

$$\lim_{k\in K} (d_k)_i(\mathscr{P}[x_k + \alpha_k d_k] - x_k)_i = 0. \tag{10.39}$$

Let j satisfies (10.36)–(10.37). If $\alpha_k < \alpha_k^j$ for infinitely many $k \in K_1 \subset K$ then, because $(d_k)_j \neq 0$ for sufficiently large $k \in K_1$ by (10.36)–(10.37),

$$\lim_{k\in K_1} \alpha_k = 0$$

from the first part of (10.38).

If $\alpha_k^j \leqslant \alpha_k$ for infinitely many $k \in K_2 \subset K$ then the second part of (10.38) gives

$$\lim_{k\in K_2} (x_k)_j = 0,$$

since $(d_k)_j \neq 0$ for sufficiently large $k \in K_2$ by (10.36)–(10.37). Therefore, by (10.37), $(d_k)_j > 0$ for sufficiently large $k \in K_2$. However, this, (10.7) imply that $\alpha_k^j = -\infty$ for sufficiently large $k \in K_2$ which contradicts the definition of K_2. This guarantees, that there exists an index $\bar{k} \in K$, such that $k \in K$ and $k \geqslant \bar{k}$ imply $k \in K_1$.

Thus

$$\lim_{k\in K}\alpha_k = 0. \tag{10.40}$$

In order to continue the proof we consider two cases:

Case (1):

$$\lim_{k\in K}\|x_k - \mathscr{P}[x_k - g_k]\| \neq 0, \tag{10.41}$$

Case (2):

$$\lim_{k\in K}\|x_k - \mathscr{P}[x_k - g_k]\| = 0. \tag{10.42}$$

Moreover, the analysis will concern only those indices $i \in \{1,\ldots,n\}$ for which

$$\lim_{k\in K}(d_k)_i = (\bar{d})_i \neq 0. \tag{10.43}$$

As we will show later this assumption does not invalidate the proof.

Case (1). First we show that

$$(g_k)_i\,(d_k)_i \geqslant 0, \;\; \forall i \notin I_k^+ \text{ such that } \alpha_k^i \leqslant \alpha_k. \tag{10.44}$$

In order to prove (10.44), due to (10.12), we have to show that $(x_k)_j \leqslant \varepsilon_k$ for all j such that $j \notin I_k^+$, $\alpha_k^j \leqslant \alpha_k$ since then $(g_k)_j \leqslant 0$ and $(d_k)_j \leqslant 0$. Because (10.41) holds we also have

$$\lim_{k\in K}\min(\varepsilon, \|x_k - D\mathscr{P}[x_k - g_k]\|) = \lim_{k\in k}\varepsilon_k = \bar{\varepsilon} > 0, \tag{10.45}$$

and due to our restriction (10.43), from (10.39), we have

$$\lim_{k\in K}(x_k)_j = 0 \tag{10.46}$$

for all j such that $\alpha_k^j \leqslant \alpha_k$. Equations (10.45) and (10.46) imply that $(x_k)_j \leqslant \varepsilon_k$ for all these j which proves (10.44).

α_k has to satisfy the relation

$$\begin{aligned} g_k(\alpha_k)^T d_k &= \sum_{i:\ \alpha_k^i > \alpha_k} (g(x_k + d_k(\alpha_k)))_i\,(d_k)_i \\ &\geqslant -\eta\|d_k^-(\alpha_k)\|^2. \end{aligned}$$

Due to our assumption we have

$$\|d_k\|\|d_k(\alpha_k)\|L + \sum_{i:\ \alpha_k^i > \alpha_k} (g_k)_i\,(d_k)_i \geqslant -\eta\|d_k^-(\alpha_k)\|^2,$$

and, from (10.18) and (10.44), we can write

$$\|d_k(\alpha_k)\|\|d_k\|L+\sum_{i\in I_k(\alpha_k)}(g_k)_i(d_k)_i\geqslant-\eta\|d_k^-(\alpha_k)\|^2. \tag{10.47}$$

We can show that

$$\|d_k(\alpha_k)\|\leqslant\alpha_k\|d_k\|,$$

and this together with

$$\sum_{i\in I_k(\alpha_k)}(g_k)_i(d_k)_i\leqslant-\|d_k^-(\alpha_k)\|^2$$

(which follows from (10.18), the definition of $d_k^-(\alpha_k)$ and (7.14)), (10.47) imply that

$$(1-\eta)\|d_k^-(\alpha_k)\|^2\leqslant\alpha_k\|d_k\|^2L. \tag{10.48}$$

Because of (10.40), in order to prove the lemma we have to show that

$$\lim_{k\in K}\|d_k^-(\alpha_k)\|\neq 0. \tag{10.49}$$

Indeed, if (10.49) does not hold then there exists at least one j which satisfies (10.36)–(10.37) and does not belong to $I_k(\alpha_k)$ for sufficiently large $k\in K$. It means that $j\in I_k^+$ and $\alpha_k^j\leqslant\alpha_k$. It follows, from (10.38), (10.39), that $\lim_{k\in K}(x_k)_j=0$ which contradicts (10.37) because $j\in I_k^+$ and we must have $(d_k)_j=-(g_k)_j<0$ for sufficiently large $k\in K$.

Case (2). Contrary to Case (1), we cannot guarantee that $(x_k)_j\leqslant\varepsilon_k$ for all j such that $j\notin I_k^+$, $\alpha_k^j\leqslant\alpha_k$ and thus that (10.44) holds.

If we assume that (10.42) is true and $\lim_{k\in K}(x_k)_i=(\bar{x})_i>0$ then $(g(\bar{x}))_i=0$. If $\lim_{k\in K}(x_k)_i=(\bar{x})_i=0$ and $(g_k)_i(d_k)_i<0$ for infinitely many $k\in K$, then from (10.37) and (10.42) (which implies $(g(\bar{x}))_i\geqslant 0$) we also have $(g(\bar{x}))_i=0$.

It means that instead of (10.47) we can use the inequality

$$\|d_k(\alpha_k)\|\|d_k\|L+\sum_{i\in I_k(\alpha_k)}(g_k)_i(d_k)_i-P_k\geqslant-\eta\|d_k^-(\alpha_k)\|^2,$$

where $\{P_k\}$ is such that $\lim_{k\in K}P_k=0$.

For large $k\in K$ we come to the contradiction

$$(1-\eta)\|d_k^-(\alpha_k)\|^2\leqslant\alpha_k\|d_k\|^2L-P_k. \tag{10.50}$$

To complete the proof notice that (10.43) was needed to show that (10.44) is valid in Case (1). If (10.43) is not satisfied for some i then instead of (10.48) we will obtain (10.50) where $P_k\to_{k\in K}0$. □

The convergence analysis and the statement of the Polak–Ribière version of Algorithm 10.1 requires partitioning of variables $(x)_i$ into two subvectors in such a way that d_k is composed of two vectors d_k^1 and d_k^2 where the first vector d_k^1 is represented by

$$d_k^1 = \{(d_k)_i\}_{i \notin I_k^+}.$$

We assume that after the variables reordering we can write

$$\begin{aligned} g_k &= (g_k^1, g_k^2), \\ d_{k-1}^+ &= ((d_{k-1}^+)^1, (d_{k-1}^+)^2) = (d_{k-1}^1, (d_{k-1}^+)^2). \end{aligned} \tag{10.51}$$

The equality (10.51) suggests that d_{k-1}^+ is partitioned according to I_{k-1}^+, however when we refer to partitioning we mean that it is done on the basis of I_k^+.

Lemma 10.2 refers to the condition (10.34) which is not equivalent to the condition

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0$$

which would state the global convergence of Algorithm 10.1 (cf. Theorem 9.2).

The global convergence analysis of Algorithm 10.1 follows that of Algorithm 7.1 for unconstrained problems. It means that we pay special attention in the analysis to two situations which can hamper the convergence: (1) vectors d_{k-1}^+ are not properly scaled – $\beta_k d_{k-1}^+$ tends to zero while $\|g_k\|$ never reaches it; (2) the angle between vectors $-g_k^1$ and d_{k-1}^1 tends to π.

Theorem 10.1. *Suppose that g is locally Lipschitz continuous with constant L. Moreover, assume that for any convergent subsequence $\{x_k\}_{k \in K}$ whose limit is not a critical point*

(i)

$$\liminf_{k \in K} (\beta_k \|d_{k-1}^1\|) \geqslant \nu_1 \liminf_{k \in K} \|g_k^1\|, \tag{10.52}$$

where ν_1 is some positive constant,

(ii) There exists a number ν_2 such that $\nu_2 \in (0,1)$,

$$\left(g_k^1\right)^T d_{k-1}^1 \leqslant \nu_2 \|g_k^1\| \|d_{k-1}^1\|, \quad \text{whenever} \quad \lambda_k \in (0,1). \tag{10.53}$$

Then $\lim_{k \to \infty} f(x_k) = -\infty$, or every accumulation point of the sequence $\{x_k\}_1^\infty$ generated by Algorithm 10.1 *is a critical point.*

Proof. Assume that

$$\hat{x} \neq \mathscr{P}[\hat{x} - g(\hat{x})], \tag{10.54}$$

where $\hat{x}$ is the accumulation point of $\{x_k\}_{k\in K}$. If (10.54) holds then two cases can occur:

(1) $(\hat{x})_i = 0$ and $(g(\hat{x}))_i < 0$ for some $i \in \{1,\ldots,n\}$
(2) $(\hat{x})_i \neq 0$ and $(g(\hat{x}))_i \neq 0$ for some $i \in \{1,\ldots,n\}$

If the case (1) occurs then there exists k_1 such that $i \notin I_k^+$ for $k \geqslant k_1$.

In the second case, if there exists k_2 such that for $k \geqslant k_2$, $i \in I_k^+$, then $(g_k)_i > 0$, $(d_k)_i = -(g_k)_i$ and from Lemma 10.2 we get $(g(\hat{x}))_i = 0$ which contradicts the fact that the second case is considered.

Therefore, if (10.54) is satisfied then there exists k_3 such that for all $k \geqslant k_3$ there exists $i \notin I_k^+$ for which either (1) or (2) holds. It means that

$$\liminf_{k\in K} \|g_k^1\| > 0.$$

Now we will show that

$$\lim_{k\in K} \|d_k^1\| = 0.$$

We have

$$(\mathscr{P}[x_k + d_k])_i - (x_k)_i = (d_k)_i, \tag{10.55}$$

or

$$(\mathscr{P}[x_k + d_k])_i - (x_k)_i = -(x_k)_i \tag{10.56}$$

which can occur if $(d_k)_i < 0$. Therefore, from Lemma 10.2, either

$$\lim_{k\in K} (d_k)_i = 0, \tag{10.57}$$

or

$$\lim_{k\in K} (x_k)_i = 0. \tag{10.58}$$

Let us denote by I^0 all indices for which (10.57) does not hold and which do not belong to I_k^+. From (10.58) and the fact that

$$\lim_{k\in K} \varepsilon_k \geqslant \hat{\varepsilon} > 0$$

(because $\hat{x}$ is not a critical point) we have $(g_k)_i \leqslant 0$, $\forall i \in I^0$ and, from (10.56), $(d_k)_i < 0$, $\forall i \in I^0$. Therefore,

$$0 \leqslant \sum_{i\in I^0} (g_k)_i \leqslant -\|d_k\|^2 - \sum_{i\notin I^0} (g_k)_i < 0$$

for sufficiently large $k \in K$. This is not possible thus (10.58) cannot happen.

The rest of the proof can be carried out in a similar way as the proof of cases (a), (b) and (c) in the proof of Theorem 7.1 and therefore it is omitted. □

As in the case of Algorithm 7.1 condition (10.53) is not necessary for the global convergence provided that the difference between x_{k+1} and x_k tends to zero.

Lemma 10.3. *Suppose that g is locally Lipschitz continuous with constant L. Assume that* $\{x_k\}$ *is generated by* Algorithm 10.1 *and* $\{\beta_k\}$ *satisfies* (10.52) *for any convergent subsequence whose limit is not a critical point. Then either*

$$\lim_{k\to\infty} f(x_k) = -\infty,$$

or for every convergent subsequence $\{x_k\}_{k\in K}$ *such that*

$$\lim_{k\in K} \|x_{k+1} - x_k\| = 0 \tag{10.59}$$

we have

$$\lim_{k\in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0. \tag{10.60}$$

Proof. First of all we assume that there exists $M > -\infty$ such that

$$f(x_k) \geqslant M, \quad \forall k,$$

otherwise we would have the thesis.

Assume that (10.60) does not hold, therefore

$$\lim_{k\in K} \|x_k - \mathscr{P}[x_k - g_k]\| \neq 0. \tag{10.61}$$

Then, as in the proof of Theorem 10.1, we can show that there exists $\alpha_1 > 0$ such that

$$\liminf_{k\in K} \|g_k^1\| = \alpha_1. \tag{10.62}$$

and that the following holds

$$\lim_{k\in K} \|d_k^1\| = 0. \tag{10.63}$$

Let us introduce the following notation

$$I^1 = \{i : \lim_{k\in K} (g_k)_i \neq 0]\}$$
$$I_k^1 = I^1 \bigcap \{i : i \notin I_k^+\}.$$

Denote by

$$\min_{i\in I^1} |(g(\hat{x}))_i| = \alpha_2 > 0. \tag{10.64}$$

The first step in the proof is to show that for infinitely many $k \in K$

$$I_k^1 = I_{k+1}^1. \tag{10.65}$$

Assume that this is not true, thus for infinitely many $k \in K$ we have, for some i, either

$$(x_k)_i > \varepsilon_k, \text{ or } (x_k)_i \leqslant \varepsilon_k \quad \text{and} \quad (g_k)_i \leqslant 0, \tag{10.66}$$

and

$$(x_{k+1})_i \leqslant \varepsilon_{k+1} \quad \text{and} \quad (g_{k+1})_i > 0, \tag{10.67}$$

or

$$(x_k)_i \leqslant \varepsilon_k \quad \text{and} \quad (g_k)_i > 0, \tag{10.68}$$

and

$$(x_{k+1})_i > \varepsilon_{k+1}, \quad \text{or} \quad (x_{k+1})_i \leqslant \varepsilon_{k+1} \quad \text{and} \quad (g_{k+1})_i \leqslant 0. \tag{10.69}$$

Both cases (10.66)–(10.67), (10.68)–(10.69) can be analyzed in the same way, therefore we will only consider (10.66)–(10.67). We assume that there exists $i \in I^1$ such that

$$(x_k)_i > \varepsilon_k, \qquad (g_k)_i > 0, \tag{10.70}$$

$$(x_{k+1})_i \leqslant \varepsilon_{k+1}, \quad (g_{k+1})_i > 0, \tag{10.71}$$

for sufficiently large $k \in K$ (because (10.59) holds and g is continuous).

Due to our assumption (10.59), (10.61) and the definition of $\{\varepsilon_k\}$ we have

$$\lim_{k\in K} \varepsilon_k = \lim_{k\in K} \varepsilon_{k+1} = \lim_{k\in K} \min\left[\varepsilon, \|x_k - \mathscr{P}[x_k - Dg_k]\|\right] = \hat{\varepsilon} > 0. \tag{10.72}$$

Furthermore, we have

$$\begin{aligned}(x_k)_i - (\mathscr{P}[x_k - Dg_k])_i &= \begin{cases} (x_k)_i & \text{if } (x_k)_i - d_i (g_k)_i \leqslant 0 \\ d_i (g_k)_i & \text{if } (x_k)_i - d_i (g_k)_i > 0 \end{cases} \\ &= \min\left[(x_k)_i, d_i (g_k)_i\right], \end{aligned} \tag{10.73}$$

(where $d_i > 0$ is the ith diagonal element of D), and from (10.59), (10.70)–(10.73), (10.64)

$$\begin{aligned}\lim_{k\in K}(x_k)_i = \lim_{k\in K}(x_{k+1})_i &= \lim_{k\in K} \varepsilon_k = \hat{\varepsilon}, \\ \lim_{k\in K} \min\left[(x_k)_i, d_i (g_k)_i\right] &= \min\left[(\hat{x})_i, d_i (g(\hat{x}))_i\right] \\ &\geqslant \min\left[\hat{\varepsilon}, d_i \alpha_2\right] > 0,\end{aligned}$$

due to the fact that $i \in I^1$. But this is impossible because for infinitely many $k \in K$ $i \in I^+_{k+1}$ ((10.71)) and thus according to (10.59), (10.73) (with $D = I$) and Lemma 10.2

$$\lim_{k\in K} \min\left[(x_k)_i, (g_k)_i\right] = 0.$$

In order to conclude the proof let us assume that condition (10.53) of Theorem 10.1, for the subsequence $\{x_{k+1}\}_{k\in K}$, does not hold. Thus, for every $\nu \in (0,1)$ we have $\left(g^1_{k+1}\right)^T d^1_k > \nu \|g^1_{k+1}\| \|d^1_k\|$ (a contradiction to condition (10.53)), provided that $0 < \lambda_{k+1} < 1$ and $k \in K$ are sufficiently large.

Since $\|x_{k+1} - x_k\| \to_{k\in K} 0$ and (10.62)–(10.65) hold, we have the relation

$$\begin{aligned} 0 < \alpha_1 \nu &\leqslant \liminf_{k\in K} \left\langle g^1_{k+1}, \frac{d^1_k}{\|d^1_k\|} \right\rangle \\ &\leqslant \lim_{k\in K} (-\|d^1_k\|) = 0, \end{aligned} \tag{10.74}$$

where the last inequality follows from

$$(g^1_k)^T d^1_k \leqslant -\|d^1_k\|^2$$

which holds because of (7.14) and the fact that $(g_k)_i (d_k)_i = -(d_k)^2_i$ for $i \in I^+_k$.

Equation (10.74) is impossible, thus (10.61) cannot happen. □

10.3 The Globally Convergent Polak–Ribière Version of Algorithm 10.1

In this section we examine Algorithm 10.1 with the sequence $\{\beta_k\}$ defined by

$$\beta_k = \frac{\left\|g^1((x^1_k, x^2_{k-1}))\right\|^2}{\left|\left(g^1((x^1_k, x^2_{k-1})) - g^1(x_{k-1})\right)^T g^1((x^1_k, x^2_{k-1}))\right|}, \tag{10.75}$$

where, as usual, by x^1 we mean a vector defined by indices not belonging to I^+_k and the vector $g^1((x^1_k, x^2_{k-1}))$ is as follows

$$g^1((x^1_k, x^2_{k-1})) = \{\left(g((x^1_k, x^2_{k-1}))\right)_i\}_{i\notin I^+_k}.$$

In Chap. 7 we proved that $\{\beta_k\}$ constructed in this way guarantees that directions generated by Algorithm 10.1 are equivalent to those generated by the Polak–Ribière algorithm, if directional minimization is exact and $I^+_k = \emptyset \ \forall k$.

We can prove the theorem.

Theorem 10.2. *Suppose that g is locally Lipschitz continuous with constant L. Then* Algorithm 10.1 *gives*

$$\lim_{k\to\infty} f(x_k) = -\infty,$$

or for any convergent subsequence $\{x_k\}_{k\in K}$

$$\lim_{k\in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0 \tag{10.76}$$

provided that:

(i) β_k *is given by* (10.75),
(ii) *there exists* $M < \infty$ *such that* $\alpha_k \leqslant M$, $\forall k$

Proof. We have for formula (10.75)

$$\begin{aligned}\beta_k \left\|d_{k-1}^1\right\| &= \frac{\left\|g^1((x_k^1, x_{k-1}^2))\right\|^2 \left\|d_{k-1}^1\right\|}{\left|\left(g^1((x_k^1, x_{k-1}^2)) - g^1(x_{k-1})\right)^T g^1((x_k^1, x_{k-1}^2))\right|} \\ &\geqslant \frac{\left\|g^1((x_k^1, x_{k-1}^2))\right\|^2 \left\|d_{k-1}^1\right\|}{L\left\|g^1((x_k^1, x_{k-1}^2))\right\| \left\|x_k^1 - x_{k-1}^1\right\|}. \end{aligned} \tag{10.77}$$

Because we have

$$\left|(\mathscr{P}[x_{k-1} + \alpha_{k-1} d_{k-1}] - x_{k-1})_i\right| \leqslant \alpha_{k-1} \left|(d_{k-1})_i\right|, \quad \forall i \in \{1, \ldots, n\},$$

thus

$$\begin{aligned}\left\|x_k^1 - x_{k-1}^1\right\| &= \left\|(\mathscr{P}[x_{k-1} + \alpha_{k-1} d_{k-1}])^1 - x_{k-1}^1\right\| \\ &\leqslant \alpha_{k-1} \left\|d_{k-1}^1\right\| \leqslant M \left\|d_{k-1}^1\right\|,\end{aligned}$$

and eventually

$$\beta_k \left\|d_{k-1}^1\right\| \geqslant \frac{\left\|g^1((x_k^1, x_{k-1}^2))\right\|}{LM}. \tag{10.78}$$

If $f(x_k) \geqslant \tilde{L} > -\infty$ then, from Lemma 10.2, we have

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k + d_k]\| = 0$$

and this is equivalent to

$$\lim_{k \in K} \min\left[(x_k)_i, -(d_k)_i\right] = 0, \quad \forall i \in \{1, \ldots, n\}. \tag{10.79}$$

Moreover, we have

$$(\mathscr{P}[x_k + \alpha_k d_k])_i - (x_k)_i = \begin{cases} -(x_k)_i & \text{if } (x_k + \alpha_k d_k)_i \leqslant 0 \\ \alpha_k (d_k)_i & \text{if } (x_k + \alpha_k d_k)_i > 0 \end{cases}.$$

Therefore,

$$\lim_{k \in K} (x_k - \mathscr{P}[x_k + \alpha_k d_k])_i = \lim_{k \in K} \min\left[(x_k)_i, -\alpha_k (d_k)_i\right]$$

and, because (10.79) and *(2)* hold, we have

$$\lim_{k \in K} (x_k - \mathscr{P}[x_k + \alpha_k d_k])_i = 0, \quad \forall i \in \{1, \ldots, n\}. \tag{10.80}$$

Lemma 10.3 states that only violation of condition (10.52) can destroy our convergence result (10.76). If we assume that $\{x_{k-1}\}_{k\in K}$ is bounded then using Lemma 10.2 and the assumption *(ii)* we can show that

$$\lim_{k\in K}\|x_k - x_{k-1}\| = 0$$

in the same way we have shown (10.80). This implies

$$\liminf_{k\in K}\left\|g^1((x_k^1, x_{k-1}^2))\right\| = \liminf_{k\in K}\|g_k^1\|$$

and that together with (10.78) assures us that condition (10.52) is fulfilled.

Therefore, in order to complete the proof we have to show that $\{x_{k-1}\}_{k\in K}$ is bounded. Indeed for any component i of x_{k-1} such that $\lim_{k\in K}(x_{k-1})_i = \infty$ we have $\lim_{k\in K}\alpha_{k-1}(d_{k-1})_i = -\infty$ because $\{(x_k)_i\}_{k\in K}$ is bounded. This implies that $\lim_{k\in K}(d_{k-1})_i(x_k - x_{k-1})_i = \infty$, due to the assumption *(2)*. But this is not possible because according to (10.20) we would have $\lim_{k\in K} f(x_k) = -\infty$. Thus $\{x_{k-1}\}_{k\in K}$ is bounded. □

The numerical results presented in Sect. 5 have been obtained by Algorithm 10.1 with

$$\beta_k = \frac{\|g_k^1\|^2}{\left|\left(g_k^1 - g_{k-1}^1\right)^T g_k^1\right|}, \tag{10.81}$$

which can be evaluated by using only one gradient calculated at every iteration. For the sequence $\{\beta_k\}$ constructed in that way we cannot guarantee global convergence of our algorithm. However, if at every iteration the condition

$$\beta_k \left\|d_{k-1}^1\right\| \geqslant \nu_1 \left\|g_k^1\right\|, \tag{10.82}$$

where ν_1 is some small positive number, is satisfied then Algorithm 10.1 with β_k defined by (10.81) is globally convergent. Thus, at every iteration we have checked (10.82) and if it has been violated we have set $d_k = -g_k$.

The next result applies to problems which satisfy sufficient optimality conditions together with the strict complementarity condition. We show the important property that Algorithm 10.1 determines the set of bounds that are active at the solution in a finite number of iterations.

Let $\bar{x}$ be a point which satisfies necessary optimality conditions for problem (10.1), (10.2). Assume also that f is twice continuously differentiable and there exist $m_2 > m_1 > 0$ such that

$$m_1\|z\|^2 \leqslant z^T\nabla^2 f(\bar{x})z \leqslant m_2\|z\|^2 \tag{10.83}$$

for all z satisfying

$$z \neq 0, \;\; (z)_i = 0, \quad \forall i \in \mathscr{A}(\bar{x}), \tag{10.84}$$

where

$$\mathscr{A}(x) = \{i : \ (x)_i = l_i, \ \ or \ (x)_i = u_i\}. \tag{10.85}$$

Then, at the point $\bar{x}$ *sufficient optimality conditions* for problem (10.1), (10.2) are satisfied (cf. e.g. Theorem 1.24 in [9], Theorem 12.6 in [146]).

Definition 10.1. Let $\bar{x}$ be a point at which necessary optimality conditions for problem (10.1), (10.2) are fulfilled. We say that strict complementarity condition holds at $\bar{x}$ if

$$(g(\bar{x}))_i > 0, \qquad \forall\, i \in \mathscr{A}(\bar{x}) \bigcap \{i : (\bar{x})_i = l_i\} \tag{10.86}$$

$$(g(\bar{x}))_i < 0, \qquad \forall\, i \in \mathscr{A}(\bar{x}) \bigcap \{i : (\bar{x})_i = u_i\}. \tag{10.87}$$

Theorem 10.3. *Let $\bar{x}$ be a point at which sufficient optimality conditions together with strict complementarity condition are satisfied. Assume that $\{x_k\}$ is generated by* Algorithm 10.1*, the assumptions of* Theorem 10.2 *hold and ε_k is calculated according to (10.14). Then there exists $\delta > 0$ such that if*

$$\|x_{\bar{k}} - \bar{x}\| \leqslant \delta$$

for some $\bar{k}$, then $\{x_k\}$ converges to $\bar{x}$ and

$$\mathscr{A}(\bar{x}) = \mathscr{A}(x_k) = I_k^+, \quad \forall k \geqslant \bar{k} + 1.$$

Proof. The proof is given in the case of constraints (10.3). From the strict complementarity condition (10.86) we know that there exists $\delta_1 > 0$ such that

$$(g_k)_i > 0, \ (\mathscr{P}[x_k - Dg_k])_i = 0 \quad \text{for all} \quad i \in \mathscr{A}(\bar{x})$$

and all x_k satisfying $\|x_k - \bar{x}\| \leqslant \delta_1$. Thus, from the definition of ε_k and the fact that $\bar{x}$ is a critical point, there exists $0 < \delta_2 \leqslant \delta_1$ such that

$$(x_k)_i \leqslant \varepsilon_k, \quad \forall i \in \mathscr{A}(\bar{x}) \tag{10.88}$$

and

$$(x_k)_i > \varepsilon_k, \quad \forall i \notin \mathscr{A}(\bar{x}) \tag{10.89}$$

for all x_k satisfying $\|x_k - \bar{x}\| \leqslant \delta_2$.

By taking into account $(g(\bar{x}))_i > 0$ for $i \in \mathscr{A}(\bar{x})$ and (10.88)–(10.89) one can show that

$$I_k^+ = \mathscr{A}(\bar{x}), \tag{10.90}$$

if $\|x_k - \bar{x}\| \leqslant \delta_2$.

Furthermore, there exist $\bar{\varepsilon} > 0$, $0 < \delta_3 \leqslant \delta_2$ for which

$$(x_k)_i > \bar{\varepsilon}, \quad \forall i \notin \mathscr{A}(\bar{x}) \tag{10.91}$$

if $\|x_k - \bar{x}\| \leqslant \delta_3$.

From (10.15)–(10.16) and the definition of d_k^1, the following relation holds

$$\begin{aligned} \|d_k^1\| &= \|d_k\| - \|g_k^2\| \\ &\leqslant \|g_k\| - \|g_k^2\| \\ &= \|g_k^1\| \end{aligned}$$

(where the inequality follows from (10.15)). Thus, from (10.90)–(10.91) and the fact that $\bar{x}$ is a critical point we know that for any $\hat{\varepsilon} > 0$ there exists $0 < \hat{\delta} \leqslant \delta_3$ such that

$$|(d_k)_i| \leqslant \hat{\varepsilon}, \quad \forall i \notin \mathscr{A}(\bar{x}) \tag{10.92}$$

if $\|x_k - \bar{x}\| \leqslant \hat{\delta}$. In particular, there exists $0 < \delta_4 \leqslant \delta_3$ such that

$$|(d_k)_i| \leqslant \bar{\varepsilon}/M, \quad \forall i \notin \mathscr{A}(\bar{x}) \tag{10.93}$$

if $\|x_k - \bar{x}\| \leqslant \delta_4$. Here, M is as specified in *(2)* of Theorem 10.2.

Now we will show that there exists $0 < \delta_5 \leqslant \delta_4$ such that

$$\alpha_k^i \leqslant \alpha_k, \quad \forall i \in I_k^+ \tag{10.94}$$

for x_k satisfying $\|x_k - \bar{x}\| \leqslant \delta_5$.

Assume the contrary. From the directional minimization rule we have the inequality

$$\sum_{i:\ \alpha_k^i > \alpha_k} (g(x_k + d_k(\alpha_k)))_i (d_k)_i \geqslant -\eta \|d_k^-(\alpha_k)\|^2.$$

This implies that

$$L_1(x_k) + L_2(x_k) \geqslant -\eta (R_1(x_k) + R_2(x_k)), \tag{10.95}$$

where

$$\begin{aligned} L_1(x_k) &= -\sum_{i \in I_k^+:\ \alpha_k^i > \alpha_k} (g_{k+1})_i (g_k)_i \\ L_2(x_k) &= \sum_{i \notin I_k^+:\ \alpha_k^i > \alpha_k} (g_{k+1})_i (d_k)_i \\ R_1(x_k) &= \sum_{i \in I_k^+:\ \alpha_k^i > \alpha_k} (g_k)_i^2 \\ R_2(x_k) &= \sum_{i \notin I_k^+} (d_k)_i^2. \end{aligned}$$

Equations (10.88) and (10.90) and the strict complementarity condition lead to

$$(x_{k+1})_i \leqslant \max\left[(x_k)_i - \alpha_k (g_k)_i, 0\right] \leqslant (x_k)_i \leqslant \varepsilon_k, \quad \forall i \in I_k^+ \tag{10.96}$$

provided that $\|x_k - \bar{x}\| \leqslant \delta_2$. Thus, since $\alpha_k \leqslant M$, $\forall k$ and (10.90), (10.92), (10.96) hold, for any $\tilde{\varepsilon} > 0$ there exists $\tilde{\delta} > 0$ such that

$$\|x_{k+1} - x_k\| \leqslant \tilde{\varepsilon}, \tag{10.97}$$

$$R_1(x_k) \leqslant -L_1(x_k) + \tilde{\varepsilon}, \tag{10.98}$$

if $\|x_k - \bar{x}\| \leqslant \tilde{\delta}$.

Furthermore, from (10.90), (10.92) we know that

$$|L_2(x_k)| \leqslant \tilde{\varepsilon} \quad \text{and} \quad R_2(x_k) \leqslant \tilde{\varepsilon} \tag{10.99}$$

if $\tilde{\delta}$ is sufficiently small.

Equations (10.95), (10.98) and (10.99) give a contradiction. It is sufficient to take

$$\tilde{\varepsilon} = 0.5(1-\eta)/(2+\eta) \min_{i \in \mathscr{A}(\bar{x})} \left[(g(\bar{x}))_i^2\right] > 0$$

(which is positive due to the strict complementarity condition). Then, we have

$$\tilde{\varepsilon} - R_1(x_k) + \tilde{\varepsilon} \geqslant -\eta R_1(x_k) - \eta\tilde{\varepsilon}.$$

Due to the contradiction to (10.94), for any $0 < \delta \leqslant \tilde{\delta}$ there exists x_k such that $\|x_k - \bar{x}\| \leqslant \delta$, $R_1(x_k)$ is positive and

$$\begin{aligned} 0 < (1-\eta)R_1(x_k) &\leqslant (2+\eta)\tilde{\varepsilon} \leqslant 0.5(1-\eta) \min_{i \in \mathscr{A}(\bar{x})} \left[(g(\bar{x}))_i^2\right] \\ &\leqslant 0.7(1-\eta) \min_{i \in \mathscr{A}(\bar{x})} \left[(g(\bar{x}))_i^2\right]. \end{aligned}$$

Because of (10.90) this is not possible, thus (10.94) is true.

Due to $(g(\bar{x}))_i > 0$, $\forall i \in \mathscr{A}(\bar{x})$, $\alpha_k \leqslant M$, $\forall k$, from (10.90), (10.91), (10.93) and (10.94) there exists $0 < \delta_6 \leqslant \delta_5$ such that

$$\mathscr{A}(\bar{x}) = \mathscr{A}(x_{k+1}) \quad \text{if} \quad \|x_k - \bar{x}\| \leqslant \delta_6 \tag{10.100}$$

and, due to (10.97), also

$$\|x_{k+1} - \bar{x}\| \leqslant \delta_4 \quad \text{if} \quad \|x_k - \bar{x}\| \leqslant \delta_6. \tag{10.101}$$

Thus, from (10.90) and (10.100)–(10.101) we have

$$\mathscr{A}(\bar{x}) = I_{k+1}^+ = \mathscr{A}(x_{k+1}) \quad \text{if} \quad \|x_k - \bar{x}\| \leqslant \delta_6.$$

Eventually, if $\|x_k - \bar{x}\| \leqslant \delta_6$ we have $\|x_{k+1} - \bar{x}\| \leqslant \delta_4$, $\mathscr{A}(\bar{x}) = \mathscr{A}(x_{k+1})$ and $(k+1)$th iteration of algorithm 10.1 reduces to the iteration of Algorithm 7.1 applied to unconstrained optimization problem defined by active constraints at $\bar{x}$.

Now, because $\|d_k\| \leqslant \|g_k\|$ and $\alpha_k \leqslant M$, we can recall Theorem 2.4. According to it there exists the neighborhood of $\bar{x}$–$\mathscr{N} \subset \mathscr{B}(\bar{x}, \delta_6) = \{x \in \mathscr{R}^n : \|x - \bar{x}\| \leqslant \delta_6\}$ such that, if $x_{k+1} \in \mathscr{N}$ and $\mathscr{A}(\bar{x}) = \mathscr{A}(x_{k+1})$ then $x_{k+2} \in \mathscr{N}$ and, from (10.100), $\mathscr{A}(\bar{x}) = \mathscr{A}(x_{k+2})$. That means that if for some $\bar{k} \in K$

$$x_{\bar{k}} \in \mathscr{N}, \quad \mathscr{A}(x_{\bar{k}}) = \mathscr{A}(\bar{x}),$$

then

$$x_k \in \mathscr{N}, \quad \mathscr{A}(x_k) = \mathscr{A}(\bar{x}), \quad \forall k \geqslant \bar{k} \quad \text{and} \quad \lim_{k \to \infty} x_k = \bar{x}$$

(from Theorem 2.4).

From the above, in order to complete the proof we have to show that there exists $\bar{\delta} > 0$ such that if $\|x_k - \bar{x}\| \leqslant \bar{\delta}$, then $x_{k+1} \in \mathscr{N}$ and $\mathscr{A}(x_{k+1}) = \mathscr{A}(\bar{x})$. According to the arguments we used to prove (10.100)–(10.101) for any $\tilde{\delta} > 0$ there exists $\bar{\delta} > 0$ such that if $\|x_k - \bar{x}\| \leqslant \bar{\delta}$ then $\|x_{k+1} - \bar{x}\| \leqslant \tilde{\delta}$, $\mathscr{A}(x_{k+1}) = \mathscr{A}(\bar{x})$. If we choose $\tilde{\delta}$ small enough we have $\mathscr{B}(\bar{x}, \tilde{\delta}) \subset \mathscr{N}$ as required. □

Notice that if the assumptions of Theorem 10.3 are satisfied then after a finite number of iterations we have $x_k^2 = x_{k-1}^2$. If $\{\beta_k\}$ is defined by (10.81) and $\nu_1 < L$ where L is the Lipschitz constant for g, then it is easy to show (cf. (10.77)) that after a finite number of iterations (10.82) is satisfied.

Consider now the quadratic case, i.e.

$$f(x) = 1/2 x^T Q x - b^T x, \tag{10.102}$$

where Q is a positive definite matrix. We also assume that the stationary point satisfies sufficient optimality conditions. If Algorithm 10.1 is applied to problem (10.1)–(10.2) with f defined by (10.102) then after a finite number of iterations active constraints at a solution are identified and our problem is reduced to a problem of unconstrained optimization. The question is whether Algorithm 10.1 performs then the iterates of a conjugate gradient algorithm.

To this end assume that an exact directional minimization is applied. It means that at every iteration we look for α_k which is a solution to the problem

$$f(\mathscr{P}[x_k + \alpha_k d_k]) = \min_{\alpha > 0} f(\mathscr{P}[x_k + \alpha d_k]). \tag{10.103}$$

It is not difficult to see that if rule (10.103) is applied then conditions (10.20)–(10.21) are satisfied. If function f is defined by (10.102) then finding α_k is equivalent to calculating the generalized Cauchy point as described in the next chapter and in [35]. After a finite number of iterations, l, active constraints at a solution are identified, and (10.103) reduces to a standard exact directional minimization of a smooth function.

Let us introduce the following notation

$$d^1 = d_l, \quad g^2 = g_{l+1}, \quad d^2 = d_{l+1}, \quad g^3 = g_{l+2}, \quad d^3 = d_{l+2}, \ldots,$$
$$d^{k+1} = d_{l+k}, \quad g^{k+1} = g_{l+k}, \ldots.$$

We can show that the directions $\underline{d}^k = d^k/\gamma_k^2$ (where $\gamma_k^2 > 0$ is defined as in (7.48)) satisfy the relations

$$\underline{d}^2 = -g^2 + \frac{(g^1)^T Q\underline{d}^1}{(\underline{d}^1)^T Q\underline{d}^1}\underline{d}^1,$$
$$\underline{d}^k = -g^k + \sum_{j=1}^{k-1} \frac{(g^k)^T Q\underline{d}^j}{(\underline{d}^j)^T Q\underline{d}^j}\underline{d}^j, \quad k = 2, \ldots, n$$

(cf. Theorem 1.6).

Moreover, we can show that

$$\operatorname{span}\{d^1, d^1, \ldots, d^k\} = \operatorname{span}\{d^1, g^1, \ldots, g^k\}, \quad k = 1, \ldots, n$$

and that

$$\left(g^k\right)^T d^j = 0, \quad j = 1, \ldots, k-1$$

(cf. Theorem 1.7). From that we obtain

$$\left(g^k\right)^T g^j = 0, \quad j = 1, \ldots, k-1.$$

Therefore, we can show that

$$\underline{d}^k = -g^k + \beta^k \underline{d}^{k-1} + \frac{(g^k)^T Q\underline{d}^1}{(\underline{d}^1)^T Q\underline{d}^1}\underline{d}^1,$$

where

$$\beta^k = \frac{(g^k)^T (g^k - g^{k-1})}{\|g^{k-1}\|^2} = \frac{\|g^k\|^2}{\|g^{k-1}\|^2}, \ k = 1, \ldots, n.$$

If

$$\left(g^k\right)^T Qd^1 = 0, \quad k = 2, \ldots, n, \tag{10.104}$$

then from the iteration the active constraints at a solution have been identified, Algorithm 10.1 behaves as a conjugate gradient algorithm applied to problem (10.1)–(10.2) with the variables $(x_k)_i$, $i \in \mathscr{A}(\bar{x})$ fixed at zero. Therefore, under (10.104), our approach to problem (10.1)–(10.2) is fully justified.

If (10.104) is satisfied then we can show that

$$\left(g^2\right)^T g^1 = 0. \tag{10.105}$$

Conditions (10.104) and (10.105) are the measure of conjugacy of the directions generated by Algorithm 10.1. Because Algorithm 10.1 is applied to nonquadratic functions and directional minimization is not exact we are interested in situations when these conditions are only approximately satisfied. This requirement is fulfilled because when

$$\left(g_k^1 - g_{k-1}^1\right)^T g_k^1 \approx 0,$$

or, when

$$\left(g_k^1\right)^T g_{k-1}^1 \approx \|g_k^1\|^2,$$

then, as it is easily seen, Algorithm 10.1 is restarted

$$d_k \approx -g_k.$$

10.4 Convergence Analysis of the Fletcher–Reeves Version of Algorithm 10.1

We can expect from Chap. 7 that convergence results for the Fletcher–Reeves version of Algorithm 10.1 are not so strong due to the fact that its unconstrained counterpart is characterized by weaker convergence properties than that of the Polak–Ribière version.

The following lemma is needed to prove the main result of the section. The lemma requires new notation. By I_k^- we denote

$$I_k^- = \{i : \ (x_k + \alpha_k d_k)_i = 0\}.$$

Based on this notation we can define vectors $d_k^{1,1}$, $x_k^{1,1}$, $d_k^{1,2}$,…,$d_k^{2,2}$ where, for example, $d_k^{1,1}$ consists of all these components of d_k which have indices belonging to I_k^- and not to I_k^+, and $x_k^{1,2}$ consists from all these components defined by indices from the set $I_k^+ \bigcap I_k^-$.

Lemma 10.4. *Suppose that:*

(i) g is locally Lipschitz continuous with constant L.
(ii) The level set $\mathscr{M} = \{x \in R^n : \ f(x) \leqslant f(x_1)\}$ *is bounded.*
(iii) There exists $0 < M < \infty$ *such*

$$\alpha_k \leqslant M \quad \textit{for all } k.$$

(iv)

$$\beta_k = 1 \quad for\ all\ k,$$

and

$$\liminf_{k\to\infty} \|\mathscr{P}[x_k - g_k] - x_k\| > 0. \tag{10.106}$$

Then,

$$\sum_{k=1}^{\infty} \left\| x_k^{2,1} \right\|^2 < \infty, \tag{10.107}$$

$$\sum_{k=1}^{\infty} \left\| d_k^{2,2} \right\|^2 < \infty, \tag{10.108}$$

$$\sum_{k=1}^{\infty} \left\| x_k^{1,1} \right\|^2 < \infty, \tag{10.109}$$

$$\sum_{k=1}^{\infty} \left\| d_k^{1,2} \right\|^2 < \infty. \tag{10.110}$$

Proof. From (10.19) we have

$$\begin{aligned} [f(x_k) - f(x_{k+1})]/\mu &\geqslant d_k^T (\mathscr{P}[x_k + \alpha_k d_k] - x_k) \\ &\geqslant \sum_{i \in I_k^+ \bigcap I_k^-} [-(d_k)_i (x_k)_i] \\ &\quad + \alpha_k \sum_{i \in I_k^+, i \notin I_k^-} (d_k)_i^2 \\ &\quad + \sum_{i \notin I_k^+, i \in I_k^-} [-(x_k)_i (d_k)_i] \\ &\quad + \alpha_k \sum_{i \notin I_k^+, i \notin I_k^-} (d_k)_i^2. \end{aligned} \tag{10.111}$$

Now, we will consider separately all components of the right-side of (10.111). First of all, from the proof of Lemma 10.2 and (10.106) we know that if (10.106) is satisfied then there exists $m > 0$ such that

$$\alpha_k \geqslant m$$

for all k.

Moreover, if $i \in I_k^-$ we must have

$$-(d_k)_i \geqslant \frac{(x_k)_i}{\alpha_k}$$

which implies that

$$-(x_k)_i (d_k)_i \geqslant \frac{(x_k)_i^2}{\alpha_k} \geqslant \frac{(x_k)_i^2}{M}.$$

Therefore, we have

$$[f(x_k) - f(x_{k+1})]/\mu \geqslant m \sum_{i \notin I_k^-} (d_k)_i^2 + \frac{1}{M} \sum_{i \in I_k^-} (x_k)_i^2 .$$

Since $f(x_k)$ is bounded due to the fact that $\{x_k\}$ is bounded, we have (10.107)–(10.110). □

We will also need the following lemma.

Lemma 10.5. *Suppose that the assumptions* (i)–(iv) *of* Lemma 10.4 *are satisfied. Then,*

$$\sum_{k=1}^{\infty} \left\| d_k^1 \right\|^2 < \infty, \tag{10.112}$$

$$\|x_{k+1} - x_k\| \left\| d_k^1 \right\| \leqslant N Q_k^2, \tag{10.113}$$

where $N < \infty$ and Q_k^2 are such that

$$\sum_{k=1}^{\infty} Q_k^2 < \infty.$$

Proof. We have

$$\begin{aligned} \|\mathscr{P}[x_k + \alpha_k d_k] - x_k\| &= \left(\sum_{i \in I_k^+, i \notin I_k^-} \alpha_k^2 (d_k)_i^2 + \sum_{i \in I_k^+, i \in I_k^-} (x_k)_i^2 \right. \\ &\quad \left. + \sum_{i \notin I_k^+, i \notin I_k^-} \alpha_k^2 (d_k)_i^2 + \sum_{i \notin I_k^+, i \in I_k^-} (x_k)_i^2 \right)^{\frac{1}{2}} \\ &\leqslant \left(\sum_{i \in I_k^+, i \notin I_k^-} (d_k)_i^2 + \sum_{i \in I_k^+, i \in I_k^-} (x_k)_i^2 \right. \\ &\quad \left. + \sum_{i \notin I_k^+, i \notin I_k^-} (d_k)_i^2 + \sum_{i \notin I_k^+, i \in I_k^-} (x_k)_i^2 \right)^{\frac{1}{2}} M. \end{aligned} \tag{10.114}$$

In order to prove (10.112) consider our directional minimization rule

$$-\eta \left\| d_k^{1,1} \right\|^2 - \eta \left\| d_k^{1,2} \right\|^2 - \eta \left\| d_k^{2,2} \right\|^2 \leqslant \left(g_{k+1}^{1,2} \right)^T d_k^{1,2} + \left(g_{k+1}^{2,2} \right)^T d_k^{2,2}. \tag{10.115}$$

Since we have

$$\left(g_{k+1}^{1,2} \right)^T d_k^{1,2} + \left(g_{k+1}^{2,2} \right)^T d_k^{2,2} = \left(g_k^{1,2} \right)^T d_k^{1,2} + \left(g_k^{2,2} \right)^T d_k^{2,2} + \left(g_{k+1}^{1,2} - g_k^{1,2} \right)^T d_k^{1,2} + \left(g_{k+1}^{2,2} - g_k^{2,2} \right)^T d_k^{2,2} \tag{10.116}$$

thus

$$-\eta \left\| d_k^{1,1} \right\|^2 - \eta \left\| d_k^{1,2} \right\|^2 - \eta \left\| d_k^{2,2} \right\|^2 \leqslant - \left\| d_k^{1,1} \right\|^2 - \left\| d_k^{1,2} \right\|^2 + \left| \left(g_{k+1}^{1,2} - g_k^{1,2} \right)^T d_k^{1,2} \right| + \left| \left(g_{k+1}^{2,2} - g_k^{2,2} \right)^T d_k^{2,2} \right|. \tag{10.117}$$

It is easy to show that

$$\left| \left(g_{k+1}^{1,2} - g_k^{1,2} \right)^T d_k^{1,2} \right| \leqslant LMQ_k^2 , \tag{10.118}$$

$$\left| \left(g_{k+1}^{2,2} - g_k^{2,2} \right)^T d_k^{2,2} \right| \leqslant LMQ_k^2 , \tag{10.119}$$

where

$$Q_k = \left(\sum_{i \in I_k^+, i \notin I_k^-} (d_k)_i^2 + \sum_{i \in I_k^+, i \in I_k^-} (x_k)_i^2 + \sum_{i \notin I_k^+, i \notin I_k^-} (d_k)_i^2 + \sum_{i \notin I_k^+, i \in I_k^-} (x_k)_i^2 \right)^{\frac{1}{2}}. \tag{10.120}$$

Eventually we come to the relation

$$(1-\eta) \left\| d_k^{1,1} \right\|^2 \leqslant -(1-\eta) \left\| d_k^{1,2} \right\|^2 + \eta \left\| d_k^{2,2} \right\|^2 + 2LMQ_k^2 \tag{10.121}$$

which, together with (10.107)–(10.110), imply that

$$\sum_{k=1}^{\infty} \left\| d_k^{1,1} \right\|^2 < \infty. \tag{10.122}$$

Equation (10.112) follows from (10.122) and (10.110).

Moreover, (10.121) implies

$$\left\|d_k^1\right\| = \left(\sum_{i \notin I_k^+} (d_k)_i^2\right)^{\frac{1}{2}} \leqslant NQ_k. \tag{10.123}$$

Combining (10.114)–(10.123) we come to the conclusion

$$\|x_{k+1} - x_k\| \left\|d_k^1\right\| \leqslant NQ_k^2$$

which together with the previous lemma give the thesis. □

Theorem 10.4. *Suppose that the assumptions* (i)–(iv) *of* Lemma 10.4 *are satisfied and one of the following conditions holds:*

(1) There exists $c < \infty$ such that

$$\left(g_{k+1}^1\right)^T d_k^1 \leqslant c \left\|d_k^1\right\|^2$$

for all k,

(2)

$$\limsup_{k\to\infty} \frac{\left\|d_k^{2,2}\right\|^2 + \left\|x_k^{2,1}\right\|^2}{\left\|d_k^1\right\|^2} = S < \infty.$$

Then,

$$\liminf_{k\to\infty} \|x_k - \mathscr{P}[x_k - g_k]\| = 0.$$

Proof. Assume that

$$\liminf_{k\to\infty} \|x_k - \mathscr{P}[x_k - g_k]\| \neq 0. \tag{10.124}$$

If (10.124) holds then two cases can occur:

(a) $\liminf_{k\to\infty} (x_k)_i = 0$ and $\liminf_{k\to\infty} (g_k)_i < 0$ for some $i \in \{1,\ldots,n\}$.
(b) $\liminf_{k\to\infty} (x_k)_i \neq 0$ and $\liminf_{k\to\infty} (g_k)_i \neq 0$ for some $i \in \{1,\ldots,n\}$.

If the case (a) occurs then there exists subsequence $\{x_k\}_{k\in K}$ and k_1 such that $i \notin I_k^+$ for $K \ni k \geqslant k_1$.

In the second case, if there exists subsequence $\{x_k\}_{k\in K_1}$ and k_2 such that for $k \geqslant k_2$, $i \in I_k^+$, then $(g_k)_i > 0$, $(d_k)_i = -(g_k)_i$ and from Lemma 10.4 we get $\liminf_{k\to\infty} (g_k)_i = 0$ which contradicts the fact that the second case is considered.

Therefore, if (10.124) is satisfied then there exists k_3 such that for all $k \geqslant k_3$ there exists $i \notin I_k^+$ for which either (a) or (b) holds. It means that

$$\liminf_{k\to\infty} \|g_k^1\| = \gamma > 0.$$

As in the proof of Theorem 7.6 we can show that

$$\frac{1}{\left\|d_k^1\right\|^2} = \frac{1}{\left\|d_{k-1}^1\right\|^2}(1+\tau_k), \tag{10.125}$$

where

$$\tau_k = \frac{\|d_{k-1}\|^2 + 2\left(g_k^1\right)^T d_{k-1}^1 + \left(\left(g_k^1\right)^T d_{k-1}^1\right)^2 / \left\|d_{k-1}^1\right\|^2}{\left\|g_k^1\right\|^2 - \left(\left(g_k^1\right)^T d_{k-1}^1\right)^2 / \left\|d_{k-1}^1\right\|^2}$$

if $0 < \lambda_k < 1$ and, as in the proof of Theorem 7.6, this is the most interesting case to consider.

From (10.121) we also have

$$\prod_{k=1}^{\infty}(1+\tau_k) = \infty \Rightarrow \sum_{k=1}^{\infty}\tau_k = \infty. \tag{10.126}$$

(1). We have

$$\tau_k \leqslant \frac{1+2c+c^2}{\gamma}\left\|d_k^1\right\|^2$$

for some $\gamma > 0$. Therefore, from (10.121)

$$\sum_{k=1}^{\infty}\tau_k < \infty$$

which is a contradiction with (10.126).

(2). On the basis of the previous lemma we can prove

$$\begin{aligned}\left|\left(g_k^1\right)^T d_{k-1}^1\right| &= \left|\left(g_k^1 - g_{k-1}^1\right)^T d_{k-1}^1\right| + \left|\left(g_{k-1}^1\right)^T d_{k-1}^1\right| \\ &\leqslant L\|\mathscr{P}[x_k + \alpha_k d_k] - x_k\|\left\|d_k^1\right\| + \left\|d_k^1\right\|^2 \\ &\leqslant NQ_k^2 + \left\|d_k^1\right\|^2.\end{aligned}$$

Now we give the appropriate bound for the sequence $\{\tau_k\}$:

$$\begin{aligned}\tau_k \leqslant\ & \frac{\left\|d_{k-1}^1\right\|^2}{\gamma} + \frac{2}{\gamma}\left[LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|^2\right] \\ & + \frac{1}{\gamma}\frac{\left[LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|\right]^2}{\left\|d_{k-1}^1\right\|^2}\end{aligned}$$

$$
\begin{aligned}
&= \frac{\left\|d_{k-1}^1\right\|^2}{\gamma} + \frac{2}{\gamma}\left[LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|^2\right] \\
&\quad + \frac{1}{\gamma}\left[\frac{L^2N^2Q_{k-1}^4}{\left\|d_{k-1}^1\right\|^2} + 2LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|^2\right] \\
&\leqslant \frac{\left\|d_{k-1}^1\right\|^2}{\gamma} + \frac{2}{\gamma}\left[LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|^2\right] \\
&\quad + \frac{1}{\gamma}\left[L^2N^2\left\{R_{k-1}^2 + Q_{k-1}^2\right\} + LNQ_{k-1}^2 + \left\|d_{k-1}^1\right\|^2\right].
\end{aligned}
$$

In the above expression R_{k-1}^2 follows from the considerations:

$$
\frac{Q_k^2}{\left\|d_k^1\right\|^2} = \frac{\sum_{i\in I_k^+, i\notin I_k^-}(d_k)_i^2 + \sum_{i\in I_k^+, i\in I_k^-}(x_k)_i^2}{\sum_{i\notin I_k^+}(d_k)_i^2} + 1
$$

and as a result

$$
\frac{Q_k^2}{\left\|d_k^1\right\|^2} \leqslant SR_k^2 + Q_k^2,
$$

where

$$
R_k^2 = \sum_{i\in I_k^+, i\notin I_k^-}(d_k)_i^2 + \sum_{i\in I_k^+, i\in I_k^-}(x_k)_i^2.
$$

Eventually, referring to Lemma 10.5 we have proved that

$$
\sum_{k=1}^{\infty}\tau_k < \infty. \tag{10.127}
$$

This contradicts with (10.125) if we notice that

$$
\sum_{k=1}^{\infty}\left\|d_k^1\right\|^2 < \infty.
$$

The cases $\lambda_k = 1$ and $\lambda_k = 0$ can be considered in the same way as in the proof of Theorem 7.1. □

Eventually we can prove the following theorem.

Theorem 10.5. *Let $\bar{x}$ be a point at which sufficient optimality conditions together with strict complementarity condition are satisfied. Assume that $\{x_k\}$ is generated by* Algorithm 10.1, *the assumptions of* Theorem 10.4 *hold and ε_k is calculated according to (10.14). Then there exists $\delta > 0$ such that if*

$$
\|x_{\bar{k}} - \bar{x}\| \leqslant \delta
$$

for some $\bar{k}$, then $\{x_k\}$ converges to $\bar{x}$ and

$$\mathscr{A}(\bar{x}) = \mathscr{A}(x_k) = I_k^+, \ \ \forall k \geqslant \bar{k} + 1.$$

This theorem can be proved as Theorem 10.3.

10.5 Numerical Experiments

Algorithm 10.1 has been implemented in FORTRAN using *double precision* accuracy. We then compared our algorithm to the limited memory BFGS algorithm [26]. We used the directional minimization described in Sect. 2 supplemented by a quadratic interpolation search. We used the following strategy – if the step from the previous iteration, for the normalized direction, did not satisfy conditions (10.20), (10.21) we performed a quadratic interpolation search based on the directional derivative and a value of the function calculated at this step. Then we checked again conditions (10.20), (10.21), if they were not satisfied, we applied the algorithm described just after the proof of Lemma 8.14. We used the following values of parameters: $\mu = 0.0001, \ \eta = 0.9$. We applied the following parameters for ε_k defined in (10.14): $\varepsilon = 0.001$ and D a diagonal matrix with elements equal to 0.001. A value of ε is not crucial, but the entries of the matrix D should be small enough to avoid the situation when too many components of d_k are only defined by components of g_k. We used $\{\beta_k\}$ defined by (10.81) and we set ν_1 in restarting criterion (10.82) to 10^{-6}. The code L-BFGS-B was run with $m = 5$ of the updates saved.

The results of our computations are shown in the tables below. The first results are grouped in Tables 10.1–10.3, which together with Table 10.4, are taken from the paper [175]. In that way we show that the superiority of the Polak–Ribière version of Algorithm 10.1 over L-BFGS-B code (in terms of CPU time) is becoming more evident when the problems dimensions are getting larger.

Our stopping criterion was

$$|(\mathscr{P}[x_k - g_k])_i - (x_k)_i| \leqslant 10^{-5}, \quad \forall i \in \{1, \ldots, n\}.$$

Results for JNLBRNGB problem from the CUTE collection are not presented for $n = 5625, n = 10000, n = 15625$ because both algorithms failed for these problems. CPU time in brackets, in Table 10.3, is the total CPU time spent on the evaluation of functions and their gradients.

We also tested Algorithm 10.1 with the ε_k defined by (10.26) with the parameters: $\varepsilon_1 = 0.001$, $\varepsilon_2 = 1.00$ and the matrix D a diagonal matrix with the entries equal to 0.001. The results for this version of our method are reported in [175] and they are very similar to those stated in Tables 10.1–10.3. Finally, in Table 10.4, we show the results for the version of the algorithm when $d_k^1 = -g_k^1$ at every iteration (we call this version the steepest descent: stp. descent). In this case the algorithm is the projection algorithm (Algorithm 9.4) except the directional minimization rule. The

Table 10.1 Numerical results for problems from the CUTE collection with $n \leqslant 2000$: Algorithm 10.1 against L-BFGS-B

		$\varepsilon_k = \min(0.001, \|w_k\|)$			m = 5	
Problem	n	LIT	IF	CPU	CPU (L-BFGS-B)	IF (L-BFGS-B)
BDEXP	1000	4	6	0.13	1.66	15
JNLBRNGA	1024	140	165	5.31	7.60	92
JNLBRNGB	1024	893	1093	33.50	34.70	424
LINVERSE	999	533	673	25.64	28.81	291
LINVERSE	1999	100	114	9.53	62.46	323
MCCORMCK	1000	20	39	1.09	0.81	15
NONSCOMP	1000	42	56	1.17	3.66	45
OBSTCLAE	1024	134	153	5.12	8.40	90
OBSTCLAL	1024	53	67	2.11	3.85	40
OBSTCLBL	1024	63	80	2.58	4.23	50
OBSTCLBM	1024	43	55	1.97	2.72	33
OBSTCLBU	1024	56	69	2.22	4.62	44
PROBPENL	500	1	3	0.03	0.05	3
TORSION1	1024	74	90	2.89	3.70	43
TORSION2	1024	72	87	2.77	5.51	61
TORSION3	1024	37	47	1.59	1.45	23
TORSION4	1024	44	63	1.85	3.77	49
TORSION6	1024	26	42	1.27	2.04	30
Total CPU				100.77	180.04	

results for problems with $n < 2000$ (results for problems with $n > 2000$ were even more favorable for the conjugate gradient version of the algorithm) clearly show the effectiveness of our approach.

The comparison of the Polak–Ribière and Fletcher–Reeves versions of Algorithm 10.1 is given in Table 10.5. The reported results were obtained on a different computer, so recorded CPU times do not match those presented in the previous tables.

The efficiency profiles are presented in Figs. 10.1–10.10. The first conclusion we can draw from these figures is that the Polak–Ribière version of Algorithm 10.1 is superior to the efficient code L-BFGS-B if we take into account the CPU time. However, with respect to the criterion of the number of function evaluations it is less efficient. Rather surprising is that Algorithm 10.1 for large problems shown in Table 10.3 is comparable to L-BFGS-B in both criteria.

The steepest version of Algorithm 10.1 is not competitive to its conjugate gradient counterpart. This suggests that not only efficient handling of box constraints contributes to the efficiency of Algorithm 10.1. Apparently, the way iterates are executed on the subspace of active constraints is also important.

Fletcher–Reeves version of Algorithm 10.1 is significantly inferior to the Polak–Ribière version of Algorithm 10.1 which is in line with the previous observation and the well-known fact that Fletcher–Reeves version of a conjugate gradient algorithm is not efficient as its Polak–Ribière counterpart.

Table 10.2 Numerical results for problems from the CUTE collection with $2000 < n \leqslant 10000$: Algorithm 10.1 against L-BFGS-B

		$\varepsilon_k = \min(0.001, \|w_k\|)$			m = 5	
Problem	n	LIT	IF	CPU	CPU (L-BFGS-B)	IF (L-BFGS-B)
BDEXP	5000	3	5	0.64	6.30	15
JNLBRNGA	5625	266	318	67.67	118.70	188
JNLBRNGA	10000	395	437	187.36	222.90	236
MCCORMCK	5000	30	56	9.02	6.95	15
MCCORMCK	10000	16	37	11.35	13.78	15
NONSCOMP	5000	45	61	6.58	22.82	51
NONSCOMP	10000	42	62	14.61	46.38	45
OBSTCLAE	5625	210	237	51.81	137.9	258
OBSTCLAE	10000	344	393	160.13	532.80	526
OBSTCLAL	5625	119	133	29.39	42.67	90
OBSTCLAL	10000	158	180	73.42	107.40	124
OBSTCLBL	5625	153	175	38.05	62.46	120
OBSTCLBL	10000	189	215	87.15	216.10	212
OBSTCLBM	5625	124	144	31.35	41.88	83
OBSTCLBM	10000	116	137	55.12	109.90	111
OBSTCLBU	5625	162	189	40.92	46.18	90
OBSTCLBU	10000	192	228	91.48	130.80	135
TORSION1	5625	211	251	53.51	62.70	125
TORSION1	10000	240	283	114.45	166.70	171
TORSION2	5476	139	165	34.91	88.07	168
TORSION2	10000	141	156	63.89	328.50	281
TORSION3	5476	153	170	37.08	20.72	51
TORSION3	10000	152	170	70.29	66.07	84
TORSION4	5476	143	170	37.41	86.94	182
TORSION4	10000	232	273	110.38	272.80	296
TORSION6	5476	67	86	17.86	54.50	123
TORSION6	10000	98	131	51.78	187.80	223
Total CPU				1547.64	3200.72	

10.6 Notes

The chapter presents the extension of the projection gradient algorithm by applying the two-metric approach introduced by Bertsekas [10]. The essential idea behind the Berteskas proposition is to use the steepest descent direction to identify the active constraints and at the same time to perform the more efficient iterate on the subspace of free variables. We show that conjugate gradient iterates can be applied while in [10] the Newton iterates are considered. In the next chapter we introduce preconditioned conjugate gradient iterates to the two-metric projection method.

In [76] Gafni and Bertsekas show the application of the two-metric projection methods to problems with general convex constraints. They consider the problem of minimizing the function f over a convex subset Ω of $\mathscr{R}^n$. Such a set can be characterized by

Table 10.3 Numerical results for problems from the CUTE collection with $n > 10000$: Algorithm 10.1 against L-BFGS-B

		$\varepsilon_k = \min(0.001, \|w_k\|)$			m=5	
Problem	n	LIT	IF	CPU	CPU (L-BFGS-B)	IF (L-BFGS-B)
JNLBRNGA	15625	340	403	266.28	469.10	332
OBSTCLAE	15625	290	340	225.60	1035.00	660
OBSTCLAL	15625	202	233	157.31	207.10	156
OBSTCLBL	15625	154	184	122.26	430.40	272
OBSTCLBM	15625	160	192	126.32	230.70	146
OBSTCLBU	15625	298	340	227.72	298.90	194
TORSION1	14884	303	355	223.17	320.40	224
TORSION2	14884	328	376	241.34	842.00	521
TORSION3	14884	187	207	136.03	89.17	76
TORSION4	14884	210	243	156.35	578.70	417
TORSION6	14884	145	177	110.50	507.00	362
Total CPU				1992.88	5008.47	

Table 10.4 Numerical results for problems from the CUTE collection with $n \leqslant 2000$: Algorithm 10.1 against the steepest descent version of Algorithm 10.1. 900* means that the specified maximum number of iterations, 900, has been exceeded

		$\varepsilon_k = \min(0.001, \|w_k\|)$			$\varepsilon_k = \min(0.001, \|w_k\|)$	
Problem	n	LIT	IF	CPU	CPU (stp. descent)	IF (stp. descent)
BDEXP	1000	4	6	0.13	0.12	6
JNLBRNGA	1024	140	165	5.31	40.58	900*
JNLBRNGB	1024	893	1093	33.50	32.29	900*
LINVERSE	999	533	673	25.64	5.4	128
LINVERSE	1999	100	114	9.53	130.48	900*
MCCORMCK	1000	20	39	1.09	1.45	45
NONSCOMP	1000	42	56	1.17	1.65	75
OBSTCLAE	1024	134	153	5.12	17.64	481
OBSTCLAL	1024	53	67	2.11	10.06	288
OBSTCLBL	1024	63	80	2.58	11.0	324
OBSTCLBM	1024	43	55	1.97	8.36	246
OBSTCLBU	1024	56	69	2.22	11.41	335
PROBPENL	500	1	3	0.03	0.05	3
TORSION1	1024	74	90	2.89	22.47	635
TORSION2	1024	72	87	2.77	32.25	935
TORSION3	1024	37	47	1.59	5.93	166
TORSION4	1024	44	63	1.85	10.92	305
TORSION6	1024	26	42	1.27	3.64	100
Total CPU				100.77	345.7	

Table 10.5 Numerical results for the Polak–Ribière version of Algorithm 10.1 (denoted as SRPRB) and Fletcher–Reeves version of Algorithm 10.1 (denoted as SRFRB)

		$\varepsilon_k = \min(0.001\|w_k\|)$		$\varepsilon_k = \min(0.001\|w_k\|)$	
Problem	n	LIT/IF (SRPRB)	CPU	CPU	LIT/IF (SRFRB)
BDEXP	5000	3/5	0.09	0.09	3/5
JNLBRNGA	5625	266/318	9.88	17.44	253/764
JNLBRNGA	10000	395/437	24.42	30.78	306/578
JNLBRNGA	15625	340/406	37.33	129.73	655/1668
MCCORMCK	5000	30/56	1.32	0.99	21/42
MCCORMCK	10000	16/37	1.75	1.72	16/37
NONSCOMP	5000	45/61	0.91	0.92	44/60
NONSCOMP	10000	42/62	2.07	2.13	45/63
OBSTCLAE	5625	210/237	6.79	8.25	172/331
OBSTCLAE	10000	344/393	21.05	16.69	193/345
OBSTCLAE	15625	290/340	31.29	32.55	257/376
OBSTCLAL	5625	119/133	3.81	4.02	96/154
OBSTCLAL	10000	158/180	9.71	9.98	101/214
OBSTCLBL	5625	153/175	4.98	3.51	89/130
OBSTCLBL	10000	189/215	11.51	22.10	149/512
OBSTCLAL	15625	202/233	21.68	47.18	208/611
OBSTCLBM	5625	124/144	4.05	4.44	91/179
OBSTCLBM	10000	116/137	7.31	7.43	111/144
OBSTCLBM	15625	160/192	17.69	38.39	152/506
OBSTCLBU	5625	162/189	5.31	4.36	92/174
OBSTCLBU	10000	192/228	11.98	19.81	129/458
OBSTCLBU	15625	298/340	35.59	67.57	248/895
TORSION1	5476	211/251	6.85	9.34	148/409
TORSION1	10000	240/283	15.20	25.96	220/575
TORSION1	14884	303/355	31.09	27.73	201/351
TORSION2	5476	139/165	4.54	9.55	165/361
TORSION2	10000	141/156	8.31	14.76	200/293
TORSION2	14884	328/376	33.24	36.06	252/464
TORSION3	5476	153/170	4.79	2.53	67/95
TORSION3	10000	152/170	9.34	7.43	84/156
TORSION3	14884	187/207	18.36	13.39	96/168
TORSION4	5476	143/177	4.75	4.34	96/176
TORSION4	10000	232/273	14.66	20.72	198/450
TORSION4	14884	210/243	21.21	30.54	264/371
TORSION6	5476	67/86	2.31	3.13	67/125
TORSION6	10000	98/131	6.81	11.71	84/266
TORSION6	14884	145/177	15.39	13.34	80/173
Total		6757/7922	484.25	740.54	5791/13202

$$\Omega = \left\{ x \in \mathscr{R}^n : \; a_i^T x \geqslant b_i, \;\; i \in I \right\}, \tag{10.128}$$

where I is possibly an infinite index set. For given $x \in \Omega$ we define $\mathscr{A}(x)$ as the set of indices corresponding to active constraints at x:

$$\mathscr{A}(x) = \left\{ i \in I : \;\; a_i^T x = b_i \right\} \tag{10.129}$$

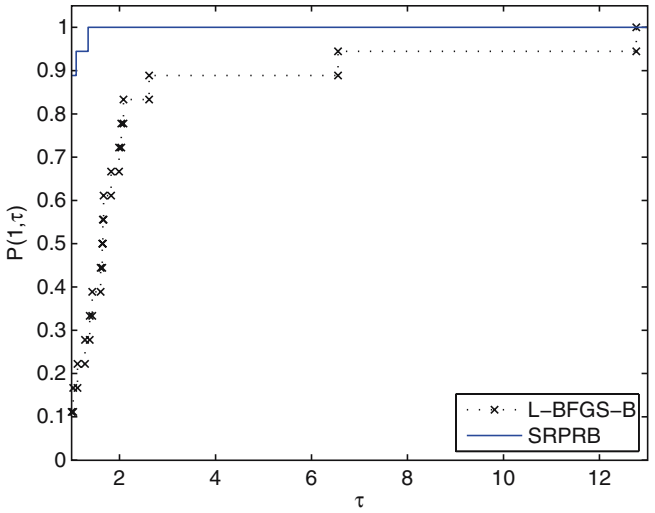

Fig. 10.1 Cumulative distribution ratio of CPU time: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.1

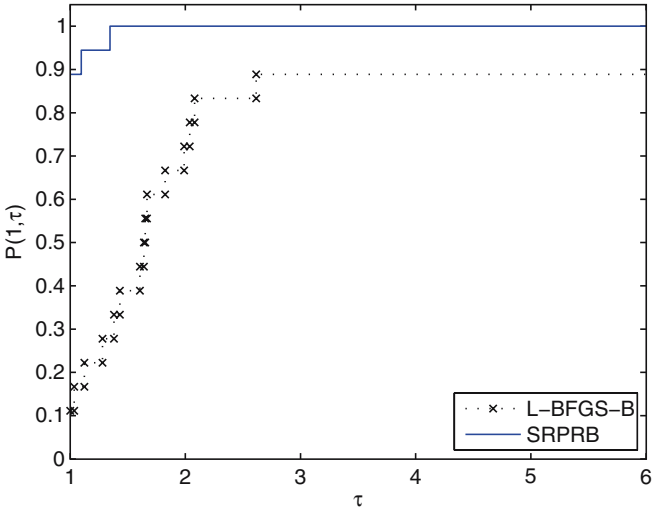

Fig. 10.2 Cumulative distribution ratio of CPU time: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.1

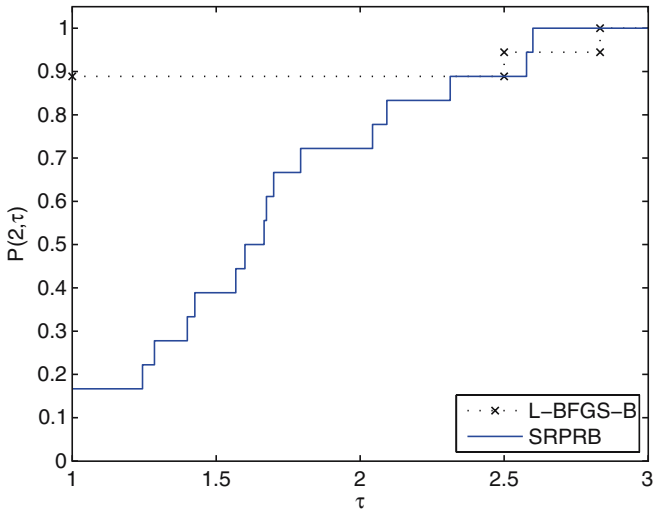

Fig. 10.3 Cumulative distribution ratio of the number of function evaluations: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.1

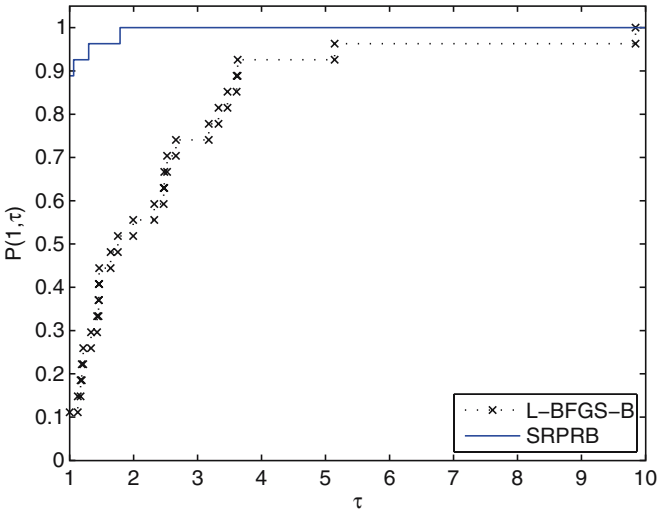

Fig. 10.4 Cumulative distribution ratio of CPU time: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.2

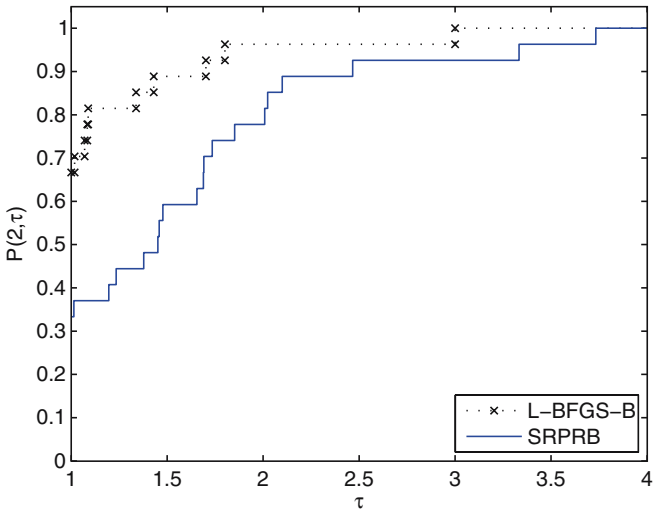

Fig. 10.5 Cumulative distribution ratio of the number of function evaluations: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.2

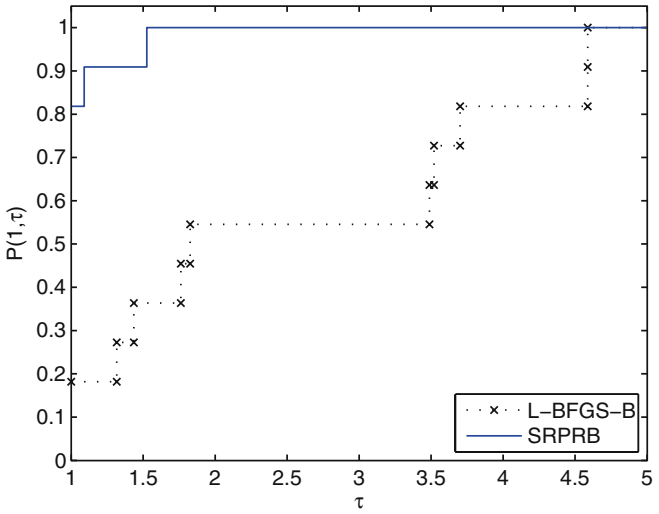

Fig. 10.6 Cumulative distribution ratio of CPU time: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.3

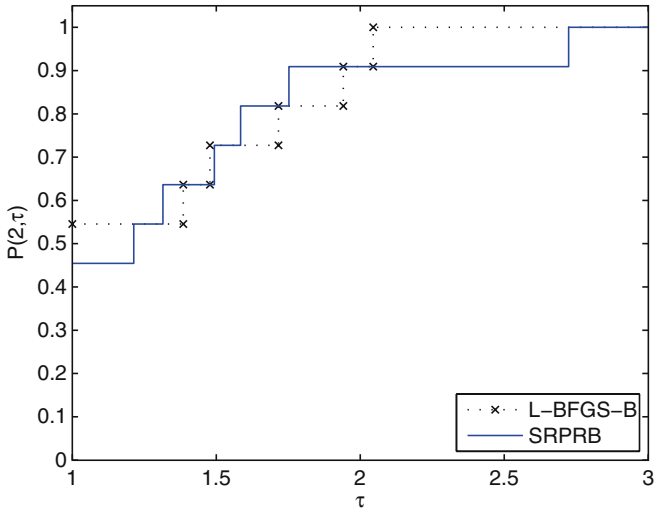

Fig. 10.7 Cumulative distribution ratio of the number of function evaluations: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against L-BFGS-B on problems from Table 10.3

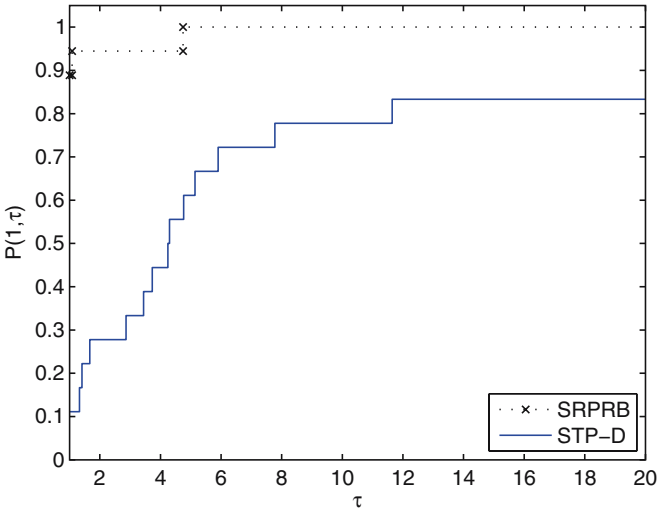

Fig. 10.8 Cumulative distribution ratio of CPU time: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against its steepest descent version on problems from Table 10.4

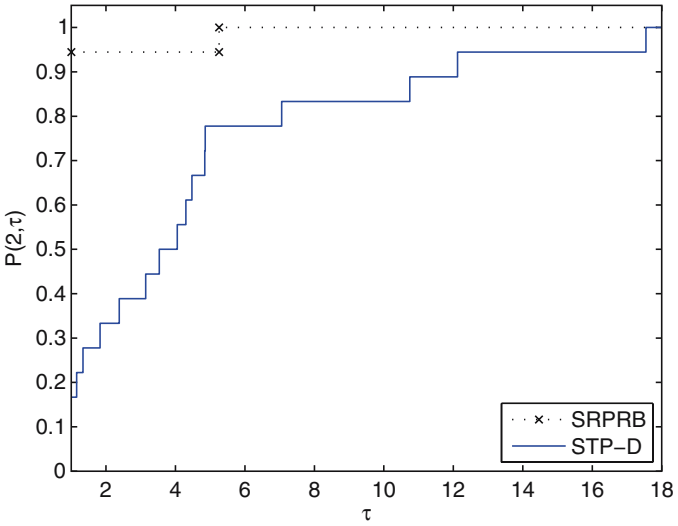

Fig. 10.9 Cumulative distribution ratio of the number of function evaluations: comparison of the Polak–Ribière version of Algorithm 10.1 (denoted on figure as SRPRB) against its steepest descent version on problems from Table 10.4

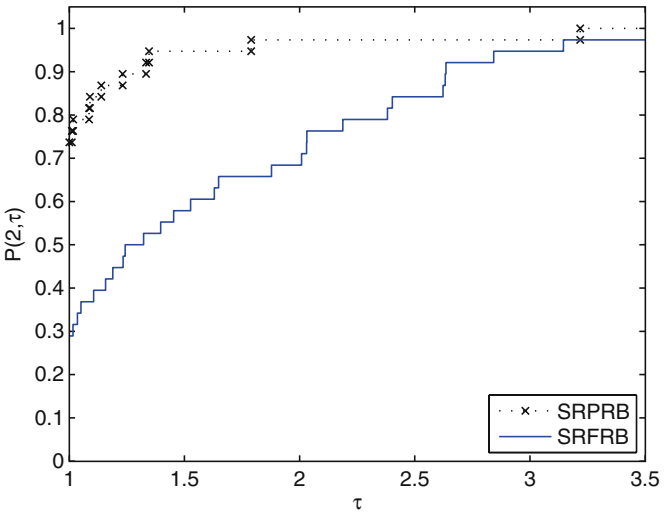

Fig. 10.10 Cumulative distribution ratio of the number of function evaluations: comparison of the Polak–Ribière and Fletcher–Reeves versions of Algorithm 10.1 (denoted on figure as SRPRB and SRFRB respectively) on problems from Table 10.5

and the set of indices for ε-active constraints

$$\mathscr{A}_\varepsilon(x) = \left\{ i \in I : \;\; b_i \leqslant a_i^T x \leqslant b_i + \varepsilon \|a_i\| \right\},$$

where $\varepsilon > 0$.

We choose $J(x)$ a subset of I such that $\mathscr{A}_\varepsilon(x) \subset J(x)$ and we use it to construct the cone $C(x)$:

$$C(x) = \left\{ z \in \mathscr{R}^n : \;\; z^T a_i \geqslant 0, \;\; \forall i \in \mathscr{A}_\varepsilon(x)) \right\}.$$

Notice that if we choose $J(x) = \mathscr{A}_\varepsilon(x)$ then for a finite set I and small values of ε we have $J(x) = \mathscr{A}(x)$ and $C(x)$ is the cone of feasible directions at x. In general, $C(x)$ is the subset of the set of feasible directions at x which means that for any $\Delta x \in C(x)$, with $\|\Delta x\| \leqslant \varepsilon$ vector $x + \Delta x$ belongs to Ω.

The dual cone to $C(x)$ denoted by $C^\perp(x)$ is defined by

$$C^\perp(x) = \left\{ z \in \mathscr{R}^n : \;\; z^T y \leqslant 0, \;\; \forall y \in C(x) \right\}$$

and in the case of $J(x) = \mathscr{A}(x)$ is generated by the vectors a_i corresponding to the active constraints at x.

The two-metric projection approach by Gafni and Bertsekas is based on the orthogonal decomposition (cf. the Moreau decomposition) of $-g(x)$ with respect to the cones $C(x)$ and $C^\perp(x)$. Thus, we project $-g(x)$ on the set $C(x)$ getting $d(x)$:

$$d(x) = \mathscr{P}_{C(x)}[-g(x)].$$

Then, the other part of $-g(x)$ decomposition is given by

$$d^\perp(x) = \mathscr{P}_{C^\perp(x)}[-g(x)]$$

and

$$-g(x) = d(x) + d^\perp(x).$$

Next we construct the subspace $\Gamma(x)$ spanned by the closed cone

$$C_+(x) = C(x) \bigcap \left\{ z : \; z^T d^\perp(x) = 0 \right\},$$

i.e. is given by

$$\Gamma(x) = \operatorname{span} C_+(x).$$

Vector $d(x)$ belongs to $\Gamma(x)$ since $d(x) \in C(x)$ and $d(x)$ is orthogonal to $d^\perp(x)$. If we now define a mapping $D(x) : \Gamma(x) \to \Gamma(x)$ then instead of $d(x)$ we will use $\tilde{d}(x)$, the projection of $D(x)d(x)$ on the cone $C_+(x)$, i.e.

$$\tilde{d}(x) = \mathscr{P}_{C_+(x)}[D(x)d(x)],$$

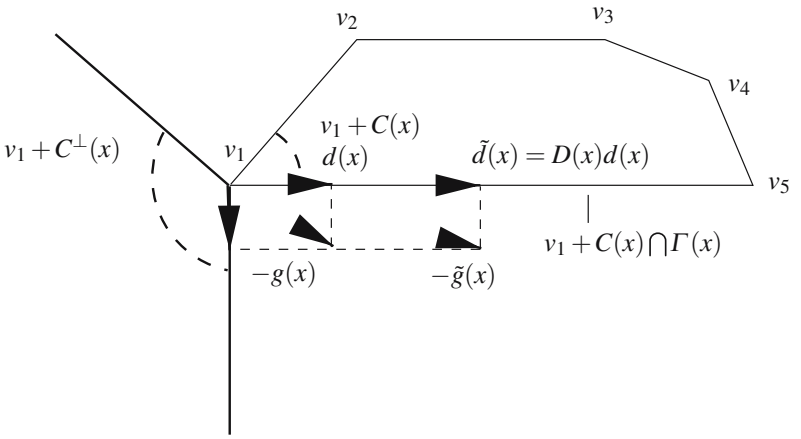

Fig. 10.11 The construction of $\tilde{g}(x)$

and, as the search direction, $\tilde{g}(x)$ is given by

$$\tilde{g}(x) = -\left(d^{\perp}(x) + \tilde{d}(x)\right)$$

instead of $-g(x)$. The construction of $\tilde{g}(x)$ is illustrated in Fig. 10.11.

Once $\tilde{g}(x_k)$ is determined the next point x_{k+1} is set to

$$x_{k+1} = \mathscr{P}_{\Omega}\left[x_k + \alpha_k \tilde{g}(x_k)\right], \tag{10.130}$$

where α_k is found by using some line search rule. Gafni and Bertsekas call that scheme the two-metric projection method since the projection in (10.130) is defined with respect to the Euclidean norm. The second norm is related to the vector $\tilde{g}(x_k)$ which is regarded as the Frechet derivative of f evaluated with respect to the norm

$$\|x\|_k = \sqrt{x^T B_k x},$$

where B_k is some positive matrix representing the mapping between g_k and $\tilde{g}(x_k)$. Thus, we can write

$$f(x) = f(x_k) + \langle \tilde{g}(x_k), x - x_k \rangle_k + o\left(\|x - x_k\|_k\right)$$

with $\langle x, y \rangle_k = x^T B_k y$.

As a particular case of (10.128) consider the set

$$\Omega = \{x \in \mathscr{R}^n : \; x \geqslant 0\}.$$

Then,

$$J(x) = \mathscr{A}_{\varepsilon}(x) = \{i : \; 0 \leqslant (x)_i \leqslant \varepsilon\}$$

and the cones $C(x)$ and $C^{\perp}(x)$ are given by

$$C(x) = \{z \in \mathscr{R}^n : \ (z)_i \geqslant 0, \ \ \forall J(x)\} \ ,$$
$$C^{\perp}(x) = \{z \in \mathscr{R}^n : \ (z)_i \leqslant 0, \ \ \forall i \in J(x), \ (z)_i = 0, \ \ \forall i \notin J(x)\} \ .$$

and, if we introduce the subset of $J(x)$

$$\tilde{J}(x) = \{i \in J(x) : \ (g(x))_i > 0\} ,$$

then the components of $-g_k$ can be evaluated in the following way

$$(d(x))_i = \begin{cases} -(g(x))_i & \text{if } i \notin \tilde{J}(x) \\ 0 & \text{if } i \in \tilde{J}(x) \end{cases} ,$$
$$(d^{\perp}(x))_i = \begin{cases} 0 & \text{if } i \notin \tilde{J}(x) \\ -(g(x))_i & \text{if } i \in \tilde{J}(x) \end{cases} .$$

Notice that if $\tilde{J}(x)$ is empty then $d(x) = -g(x)$ and $\tilde{g}(x) = Dg(x)$ for some positive matrix $D \in \mathscr{R}^{n \times n}$. In the other case we can rearrange indices of vector x so that the first p indices belong to $\tilde{J}(x)$ and $g(x)$ can be stated as

$$g(x) = \begin{bmatrix} g^1(x) \\ g^2(x) \end{bmatrix}$$

with $g^1 \in \mathscr{R}^p$ and $g^2 \in \mathscr{R}^{n-p}$. Then, with $D(x) \in \mathscr{R}^{p \times p}$,

$$\tilde{q}(x) = -\begin{bmatrix} \mathscr{P}_{C(x)}[D(x)g^1(x)] \\ g^2(x) \end{bmatrix},$$

where $\mathscr{P}_{C(x)}[D(x)g^1(x)]$ is obtained by setting to zero components of $D(x)g^1(x)$ whose indices belong to $J(x)$ and which are negative (notice that $\tilde{J}(x)$ is the subset of $J(x)$).

Chapter 11
Preconditioned Conjugate Gradient Algorithms for Problems with Box Constraints

11.1 Introduction

In this chapter we continue dealing with the problem:

$$\min_{x \in \mathscr{R}^n} f(x) \tag{11.1}$$

subject to the constraints

$$l \leqslant x \leqslant u. \tag{11.2}$$

In the previous chapter we show that the method of shortest residuals which uses the projection operator to cope with box constraints is competitive to the benchmark code L-BFGS-B in terms of CPU time (cf. Figs. 10.1, 10.2, 10.4, 10.6). For larger problems it is almost as efficient as L-BFGS-B program also in terms of the number of function evaluations (cf. Fig. 10.7). Encouraged by the efficiency of the method discussed in the previous chapter we attempt to further increase its efficiency by applying preconditioning matrix D along the lines outlined in Chap. 8.

The aim of the chapter is also to introduce to the algorithm behind the L-BFGS-B code which handles differently box constraints.

11.2 Active Constraints Identification by Solving Quadratic Problem

The L-BFGS-B code is based on the algorithm which has three major steps at an iteration:

1. The generalized Cauchy point evaluation – the step is required to identify active constraints for the next iteration.

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

2. The subspace minimization – having identified active constraints the quadratic approximation to the objective function f, evaluated at the current point x_k, is minimized on the subspace of free variables.
3. On the basis of the two points obtained in the previous steps the direction of descent for f at x_k is obtained along which f is approximately minimized.

The algorithm refers to the quadratic approximation of f evaluated at the point x_k:

$$m_k(x) = f(x_k) + g_k^T (x - x_k) + \frac{1}{2} (x - x_k)^T B_k (x - x_k) \tag{11.3}$$

in which B_k is the limited memory BFGS approximation to the Hessian matrix. In order to find a direction of descent for f at x_k one could solve the quadratic problem

$$\min_{l \leqslant x \leqslant u} m_k(x) \tag{11.4}$$

to get $\hat{x}_{k+1}$ and then substitute $\hat{x}_{k+1} - x_k$ for d_k. However, this approach would not be efficient since solving (11.4) resorts to an iterative procedure with the cost of $B_k v$ ($v \in \mathscr{R}^n$) evaluation (cf. Algorithm 5.5) multiplied by the number of iterations performed by the procedure. Such a procedure proposed in [138] (see also [137]) performs iterations of a conjugate gradient algorithm to explore the face of the feasible region defined by the current iterate and uses the gradient projection method to move to a different face.

The algorithm presented in [26] also applies the gradient projection method to determine the active set of constraints at each iteration. However, the gradient projection iteration is used on the quadratic approximation of f evaluated at the current point instead of on the function f itself. This approach significantly simplifies the directional minimization performed at the iteration of the gradient projection method and, as the convergence results derived in [35] for the related trust region algorithm suggest, can be sufficient to identify the active set of constraints at a solution in a finite number of iterations.

Suppose that the gradient projection method is applied to the function m_k thus the directional minimization can be stated as follows

$$\min_{t>0} m_k \left(\mathscr{P}[x_k - t g_k] \right). \tag{11.5}$$

(Notice that g_k is also the gradient of m_k evaluated at x_k.) In order to find an approximate solution to the problem (11.6) the generalized Cauchy point is found, i.e. a point x_k^C defined through $\bar{t}_k$:

$$x_k^C = \mathscr{P}[x_k - \bar{t}_k g_k], \tag{11.6}$$

where $\bar{t}_k$ is the first local minimizer of problem (11.5). The justification for using the generalized Cauchy point in the scheme proposed in [26] (see also [36], [37]) probably lies in the convergence analysis carried out in [35] where a globally convergent trust region algorithm for bound constrained problems is discussed. The use of the

Cauchy point in trust regions methods guarantees their global convergence [146]. When these methods are modified to cope with problems with simple constraints then the generalized Cauchy point (which is the Cauchy point evaluated in the presence of the constraints) not only is needed for global convergence but also to identify the set of active constraints at a solution in a finite number of iterations.

The calculation of the generalized Cauchy point reduces to one-dimensional minimization of a piecewise quadratic function. Suppose that $t_k^1, \ldots, t_k^m$ are bent points for the direction $d_k = -g_k$ as defined through (10.6), i.e. we first evaluate the points $\hat{t}_k^i$, $i = 1, \ldots, n$:

$$\hat{t}_k^i = \begin{cases} (x_k - u)_i / (g_k)_i & \text{if } (g_k)_i < 0, \\ (x_k - l)_i / (g_k)_i & \text{if } (g_k)_i > 0, \\ \infty & \text{otherwise,} \end{cases} \tag{11.7}$$

and then we order them to obtain $0 = t_k^0 \leqslant t_k^1 < t_k^2 < \cdots < t_k^m$.

The bent points define the line segments of the piecewise linear search path so the jth segment connects the point $x_k^j = x_k + d_k(t_k^j)$ and $x_k^{j+1} = x_k + d_k(t_k^{j+1})$ where

$$d_k(t) = \mathscr{P}[x_k - t g_k] - x_k, \; t > 0. \tag{11.8}$$

For each segment we also define the search direction for the jth segment – d_k^j according to the formula

$$\left(d_k^j\right)_i = \begin{cases} -(g_k)_i & \text{if } \hat{t}_k^i > t_k^j, \\ 0 & \text{otherwise.} \end{cases} \tag{11.9}$$

Using this notation we can write the quadratic (11.3) on the line segment $[x_k^j, x_k^{j+1})$ as the function of $\Delta t \in [0, t_k^{j+1} - t_k^j)$. If we assume that

$$z_k^j = x_k^j - x_k \tag{11.10}$$

then for $x \in [x_k^j, x_k^{j+1})$

$$\begin{aligned} m_k(x) &= f(x_k) + g_k^T (x - x_k) + \frac{1}{2} (x - x_k)^T B_k (x - x_k) \\ &= f(x_k) + g_k^T \left(z_k^j + \Delta t d_k^j\right) \\ &\quad + \frac{1}{2} \left(z_k^j + \Delta t d_k^j\right)^T B_k \left(z_k^j + \Delta t d_k^j\right) \end{aligned} \tag{11.11}$$

and, since $x = x_k^j + \Delta t d_k^j$, the quadratic m_k is in fact a quadratic of Δt:

$$\begin{aligned} m_k^j(\Delta t) &= m_k(x_k^j + \Delta t d_k^j) \\ &= \alpha_k^j + \beta_k^j \Delta t + \frac{1}{2} \gamma_k^j \Delta t^2 \end{aligned} \tag{11.12}$$

with

$$\alpha_k^j = f(x_k) + g_k^T z_k^j + \frac{1}{2}\left(z_k^j\right)^T B_k z_k^j \tag{11.13}$$

$$\beta_k^j = g_k^T d_k^j + \left(d_k^j\right)^T B_k z_k^j \tag{11.14}$$

$$\gamma_k^j = \left(d_k^j\right)^T B_k d_k^j. \tag{11.15}$$

The local minimizer $\Delta\bar{t}_k^j$ of m_k^j on $[t_k^j, t_k^{j+1})$, if it exists, can be determined by taking into account the first and second order derivatives of m_k^j, i.e.

$$\Delta\bar{t}_k^j = -\frac{\beta_k^j}{\gamma_k^j}. \tag{11.16}$$

If $\Delta\bar{t}_k^j$ is not located in the interval $[t_k^j, t_k^{j+1})$ then it lies in the proceeding intervals provided that $\beta_k^j \geqslant 0$, or in the next intervals under the assumption $\beta_k^j < 0$.

The searching for x_k^C starts with $j = 0$. If the local minimizer of m_k^0 is not in the interval $[t_k^0, t_k^1]$ we increase j by one, β_k^j and γ_k^j are updated and the next interval is examined. We proceed in that way until we find $\bar{j}$ such that $\Delta\bar{t}_k^j \in [t_k^j, t_k^{j+1})$ in which case we assume $x_k^C = x_k^j + \Delta\bar{t}_k^j d_k^j$, or $x_k^C = x_k^{m-1} + \left(t_k^m - t_k^{m-1}\right) d_k^{m-1}$. The cost of the generalized Cauchy point evaluation is estimated in [26] as $(2m+2)n + O(m^2)n_{int} + n\log n$ floating point operations where m is the number of pairs used in the limited memory BFGS matrix, n_{int} is the number of intervals explored by the algorithm and $n\log n$ is the cost of sorting break points to define the bent points.

Having the generalized Cauchy point x_k^C we determine the set of active constraints $\mathscr{A}(x_k^C)$ and proceed to the next step of an iteration, that of the subspace minimization. To this end we define the set of free variables:

$$\mathscr{F}\left(x_k^C\right) = \{1, \ldots, n\} \backslash \mathscr{A}\left(x_k^C\right) \tag{11.17}$$

and formulate the quadratic programming problem

$$\min_{x \in \mathscr{R}^n} m_k(x) \tag{11.18}$$

subject to the constraints

$$(x)_i = \left(x_k^C\right)_i, \quad \forall i \in \mathscr{A}\left(x_k^C\right) \tag{11.19}$$

$$(l)_i \leqslant (x)_i \leqslant (u)_i, \quad \forall i \in \mathscr{F}\left(x_k^C\right). \tag{11.20}$$

The problem (11.18)–(11.20) is not solved exactly, instead we look for the point $\hat{x}_{k+1}$ for which $m_k(\bar{x}_{k+1}) \leqslant m_k(x_k^C)$ and this can be easily achieved by setting $\bar{x}_{k+1} = x_k^C$. However, there exists a better strategy for the approximate solution of the problem then resorting to the generalized Cauchy point.

We solve approximately the quadratic problem (11.18)–(11.20) in two stages. First, we solve exactly the problem in which constraints (11.20) are ignored. The problem (11.18)–(11.19) can be stated as

$$\min_{d^F \in \mathscr{R}^{n_F}} \left[m_k^F(d^F) = m_k\left(x_k^C + Z_k d^F\right)\right], \tag{11.21}$$

where $n_F = |\mathscr{F}(x_k^C)|$ and $Z_k \in \mathscr{R}^{n \times n_F}$ is a matrix whose columns are unit vectors corresponding to the free variables.

If $\bar{d}_k^F$ is the solution to the problem (11.21), in the second stage we set

$$\bar{x}_{k+1} = x_k^C + \hat{t}_k \bar{d}_k^F \tag{11.22}$$

with

$$\hat{t}_k = \max\left\{t : t \leqslant 1,\ (l)_i - \left(x_k^C\right)_i \leqslant t\left(\bar{d}_k^F\right)_i \leqslant (u)_i - \left(x_k^C\right)_i, \right. \\ \left. i \in \mathscr{F}\left(x_k^C\right)\right\}. \tag{11.23}$$

Since $\bar{x}_{k+1}$ is on a path from x_k^C to the minimizer of (11.18)–(11.19) along which m_k decreases, the direction $d_k = \bar{x}_{k+1} - x_k^C$ is a direction of descent of f at x_k:

$$\begin{aligned} f(x_k) &= m_k(x_k) > m_k\left(x_k^C\right) \\ &\geqslant m_k(\bar{x}_{k+1}) = f(x_k) + g_k^T d_k + \frac{1}{2} d_k^T B_k d_k \end{aligned} \tag{11.24}$$

since we assume that B_k is positive definite thus $g_k^T d_k < 0$ (provided that $d_k \neq 0$).

There are two reasons for using two-stage direction finding procedure. First, we solve a problem with only equality constraints and its solution, as we show below, can be accomplished with the acceptable number of floating point operations. Furthermore, the outcome of the procedure is the point $\bar{x}_{k+1}$ for which $m_k(\bar{x}_{k+1}) \leqslant m_k(x_k^C)$ and this, by analogy to trust region methods, can guarantee global convergence of the discussed procedure.

The problem (11.21) can be reformulated as (see [26] for details):

$$\min_{d^F \in \mathscr{R}^{n_F}} \left[m_k^F\left(d^F\right) = r_k^T d^F + \frac{1}{2}\left(d^F\right)^T B_k^F d^F + \delta_k \right], \tag{11.25}$$

where

$$r_k = Z_k^T\left(g_k + B_k\left(x_k^C - x_k\right)\right) \tag{11.26}$$

$$B_k^F = Z_k^T B_k Z_k \tag{11.27}$$

and δ_k is some constant.

In order to solve (11.25) it is sufficient to invert the matrix B_k^F. If B_k is the limited memory BFGS matrix stated in its compact representation form (cf. (5.50))

$$B_k = \gamma_k I - W_k M_k W_k^T$$

then

$$B_k^F = \gamma_k I - Z_k^T W_k M_k W_k^T Z_k$$

(since $Z_k^T Z_k = I$) and, by applying the Sherman–Morrison–Woodbury formula (cf. (B.17)), the solution to problem (11.25) can be stated as

$$d_k^F = -\frac{1}{\gamma_k} r_k - \frac{1}{\gamma_k^2} Z_k^T W_k \left(I - \frac{1}{\gamma_k} M_k W_k^T Z_k Z_k^T W_k \right)^{-1} M_k W_k^T Z_k r_k. \tag{11.28}$$

The cost of d_k^F evaluation can be estimated (cf. [26]) as $2m^2 n_F + 6mn_F + 4n_F + O(m^3)$ floating point operations.

The third step of an iteration focuses on the approximate minimization of f along the direction d_k. The strong Wolfe conditions are used as criteria for stopping the step-length selection procedure:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k \tag{11.29}$$

$$\left| g(x_k + \alpha_k d_k)^T d_k \right| \leqslant \eta \left| g_k^T d_k \right| \tag{11.30}$$

with $0 < \mu < \eta < 1$. If the step-length procedure generates points which are not greater than one, then $x_{k+1} = x_k + \alpha_k d_k$ is a feasible point. In the other case some safeguards are needed to keep the points in the feasible region and a procedure such as the Moré–Thuente algorithm cannot guarantee finding α_k in a finite number of iterations (at least theoretically).

The algorithm proposed in [26] has one feature which makes the algorithm similar to the projected method introduced in [10]: at an iteration the set of supposedly active constraints at the next iteration is identified and at the same time the function is minimized with respect to the assumed free variables. However, in the method by Bertsekas the identification is based on the value of function f while in the other method on the value of m_k.

The outlined algorithm, which is the basis for the benchmark code L-BFGS-B, does not have proven global convergence properties.

11.3 The Preconditioned Shortest Residuals Algorithm: Outline

According to the Bertsekas approach [10] which we follow in Algorithm 10.1 the scaling matrix should be applied only to variables with indices corresponding to the set I_k^+. Therefore, we choose the scaling matrix as

$$D_k = \begin{bmatrix} D_k^+ & 0 \\ 0 & I_{n_k} \end{bmatrix}. \tag{11.31}$$

Here, D_k^+ is a nonsingular matrix of dimension $n - n_k$ and I_{n_k} is the identity matrix of dimension $n_k = |I_k^+|$ (cf. (10.12)). As in Chap. 10, for the simplicity of presentation we assume that $x \geqslant 0$ are the only constraints.

We use the matrix D_k to transform x variables to $\hat{x}$ variables. As a result we end up with the equations

$$D_k^T \hat{g}_k = g_k \tag{11.32}$$
$$D_k d_k = \hat{d}_k. \tag{11.33}$$

Notice that we have

$$(\hat{d}_k)_i = (d_k)_i, \quad i \in I_k^+ \tag{11.34}$$
$$(\hat{g}_k)_i = (g_k)_i, \quad i \in I_k^+ \tag{11.35}$$

and $\hat{d}_k$ is defined by

$$\hat{d}_k = -\mathbf{Nr}\left\{\hat{g}_k, -\hat{\beta}_k \left(\hat{d}_{k-1}^c\right)^+\right\}, \tag{11.36}$$

where

$$\hat{d}_{k-1}^c = D_k d_{k-1},$$
$$\left(\left(\hat{d}_{k-1}^c\right)^+\right)_i = \begin{cases} \left(\hat{d}_{k-1}^c\right)_i & \text{if } i \notin I^+ \\ -(\hat{g}_k)_i/\hat{\beta}_k & \text{if } i \in I_k^+ \end{cases}$$

(cf. (8.9) and (10.16)).

The rule (11.36) can also be stated as

$$\hat{d}_k = -\lambda_k \hat{g}_k + (1-\lambda_k)\hat{\beta}_k \left(\hat{d}_{k-1}^c\right)^+, \tag{11.37}$$

where $\lambda_k \in [0,1]$ is the result of evaluating the mapping **Nr**.

We propose several new line search rules for our preconditioned conjugate gradient algorithms. The first one is based on the Armijo rule stated in [10] (see also (9.54)) – we call it the *second generalized Armijo rule*: find the largest positive number α_k from the set $\{\beta^k : k = 0,1,\ldots,\ \beta \in (0,1)\}$ such that

$$f(x_k(\alpha_k)) - f(x_k) \leqslant -\mu \left[\alpha_k \sum_{i \notin I_k^+} \left(\hat{d}_k\right)_i^2 + \sum_{i \in I_k^+} (g_k)_i \left[(x_k)_i - (x_k(\alpha_k))_i\right]\right], \tag{11.38}$$

$\mu \in (0,1)$. The first sum on the right-hand side of (11.38) is related to the subspace of these variables which are scaled by the transformation D_k^+. From (8.7) and (11.34)–(11.35) we can deduce that

$$\sum_{i \notin I_k^+} (g_k)_i (d_k)_i \leqslant -\sum_{i \notin I_k^+} \left(\hat{d}_k\right)_i^2 \tag{11.39}$$

thus (11.38) can be regarded as the extension of the Armijo rule used by Bertsekas in [10]. To show that notice that if $(D_k^+)^T D_k^+$ is set to the Hessian matrix with respect to variables with indices not belonging to I_k^+ and the previous direction is not taken into account then $\lambda_k = 1$ in (10.15) and

$$\sum_{i \notin I_k^+} (g_k)_i (d_k)_i = - \sum_{i \notin I_k^+} \left(\hat{d}_k\right)_i^2 . \tag{11.40}$$

The other directional minimization rule is the extension of the rule stated in Sect. 10.2 – we call it the *second generalized Wolfe rule*: find a positive number α_k such that

$$f(x_k(\alpha_k)) - f(x_k) \leqslant -\mu \left[\alpha_k \sum_{i \notin I_k^+} \left(\hat{d}_k\right)_i^2 + \sum_{i \in I_k^+} (g_k)_i \left[(x_k)_i - (x_k(\alpha_k))_i \right] \right] \tag{11.41}$$

$$g_k(\alpha_k)^T d_k \geqslant -\eta \left[\|\hat{d}_k\|^2 - \sum_{i \in I_k(\alpha_k)} (d_k)_i^2 \right], \tag{11.42}$$

where $0 < \mu < \eta < 1$. The condition (11.41) replaces

$$f(x_k(\alpha_k)) - f(x_k) \leqslant -\mu d_k^T \left(P[x_k + \alpha_k d_k] - x_k \right) \tag{11.43}$$

which is the sufficient decrease condition of the generalized Wolfe rule advocated in Sect. 10.2, while (11.42) is used instead of

$$g_k(\alpha_k)^T d_k \geqslant -\eta \|d_k^-(\alpha_k)\|^2 \tag{11.44}$$

as applied in Sect. 10.2. Here, $\|d_k^-(\alpha_k)\|^2 = \|d_k\|^2 - \sum_{i \in I_k(\alpha_k)} (d_k)_i^2$ (cf. (10.18)). Thus, if $D_k^+ = I$ then (11.41) is equivalent to (11.44). Both modifications to the generalized Wolfe rule stated in Sect. 10.2 are enforced by the use of the scaling matrix D_k.

The advantage of using (11.41)–(11.42) stems from mimicking the Wolfe conditions which are adequate for conjugate gradient algorithms (see discussion in Sect. 10.2).

In the chapter we discuss the preconditioned version of the method of shortest residuals as defined in Sects. 7.2–7.4. However, the convergence results stated for this version apply also to the version introduced in Sect. 7.5. In this case we have to work with the direction rule

$$\hat{d}_k = -\hat{\lambda}_k \hat{g}_k + (1 - \hat{\lambda}_k) \left(\hat{d}_{k-1}^c \right)^+ \tag{11.45}$$

with

$$\hat{\lambda}_k = \frac{\|\hat{g}_k\|^2 + \hat{\beta}_k \hat{g}_k^T \left(\hat{d}_{k-1}^c\right)^+}{\|\hat{g}_k + \hat{\beta}_k \left(\hat{d}_{k-1}^c\right)^+\|^2}. \tag{11.46}$$

One advantage of using the formula (11.45)–(11.46) instead of (11.36) is that we can rely on the Wolfe conditions in a line search (see the discussion of both versions of the method of shortest residuals in Sect. 7.5) if unconstrained optimization is considered. These conditions for box constrained problems could be stated as follows

$$f(x_k(\alpha_k)) - f(x_k) \leqslant -\mu \left[\alpha_k \sum_{i \notin I_k^+} (g_k)_i \left(\hat{d}_k\right)_i + \sum_{i \in I_k^+} (g_k)_i [(x_k)_i - (x_k(\alpha_k))_i] \right] \tag{11.47}$$

and

$$g_k(\alpha_k)^T d_k \geqslant \eta \left[g_k^T d_k - \sum_{i \in I_k(\alpha_k)} (d_k)_i^2 \right], \tag{11.48}$$

where $0 < \mu < \eta < 1$.

Our general algorithm is as follows.

Algorithm 11.1. (The preconditioned shortest residuals algorithm for problems with box constraints)

Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu,\ \varepsilon > 0,\ \{\hat{\beta}_k\},\ \{D_k\},\ D_k \in \mathscr{R}^{n \times n}$ nonsingular matrix, diagonal positive definite matrix M.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute

   $$d_1 = -g_1$$

   and set $k = 1$.
2. Find α_k according to the second generalized Armijo rule, or to the second generalized Wolfe rule.
3. Compute

   $$w_{k+1} = x_{k+1} - \mathscr{P}[x_{k+1} - M g_{k+1}]. \tag{11.49}$$

   If $\|w_{k+1}\| = 0$ then STOP, otherwise calculate

   $$\varepsilon_{k+1} = \min(\varepsilon, \|w_{k+1}\|), \tag{11.50}$$

create D_k^+ due to (11.31) and find d_{k+1} by solving:

$$D_{k+1}^T \hat{g}_{k+1} = g_{k+1} \tag{11.51}$$

$$\hat{d}_k^c = D_{k+1} d_k \tag{11.52}$$

$$\hat{d}_{k+1} = -\mathbf{Nr}\left\{\hat{g}_{k+1}, -\hat{\beta}_{k+1}\left(\hat{d}_k^c\right)^+\right\} \tag{11.53}$$

$$D_{k+1} d_{k+1} = \hat{d}_{k+1}. \tag{11.54}$$

4. Substitute $\mathscr{P}[x_k + \alpha_k d_k]$ for x_{k+1}, increase k by one, go to Step 2.

The convergence analysis of Algorithm 11.1 is restricted to its version based on the second generalized Armijo rule. We can prove the following lemma.

Lemma 11.1. *Assume that x_k is a noncritical point, $\left(D_k^+\right)^T D_k^+$ is positive definite and $d_k \neq 0$ is calculated in* Step 3 *of* Algorithm 11.1. *Then there exists a positive α_k such that* (11.38) *is satisfied, or*

$$\lim_{\alpha \to \infty} f(P[x_k + \alpha d_k]) = -\infty. \tag{11.55}$$

Proof. The proof is given in the proof of the next lemma. □

To investigate the convergence of Algorithm 11.1 we begin by providing the following crucial lemma.

Lemma 11.2. *Suppose that g is Lipschitz continuous with constant L and there exist d_l, d_u such that $0 < d_l < d_u < \infty$ and*

$$d_l \|u\|^2 \leqslant u^T D_k^T D_k u \leqslant d_u \|u\|^2 \tag{11.56}$$

for all $u \in \mathscr{R}^n$ and k. If $\{x_k\}$ is generated by Algorithm 11.1, *then for any bounded subsequence $\{x_k\}_{k \in K}$ either*

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k + d_k]\| = 0, \tag{11.57}$$

or

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0. \tag{11.58}$$

Proof. Because $\{f(x_k)\}_{k \in K}$ is non-increasing and bounded from below (because $\{x_k\}_{k \in K}$ is bounded), it has to be convergent. From (11.38) and the theorem on three sequences we have

$$\lim_{k \in K} [f(x_{k+1}) - f(x_k)] = 0$$

$$\lim_{k \in K} \mu \sum_{i \in I_k^+} (g_k)_i \left[(x_k)_i - (x_k(\alpha_k))_i\right] = 0, \tag{11.59}$$

$$\lim_{k\in K}\mu\sum_{i\notin I_k^+}\alpha_k(\hat{d}_k)_i^2=0. \tag{11.60}$$

On the basis of (11.56) we also have

$$\lim_{k\in K}\mu\sum_{i\notin I_k^+}\alpha_k(d_k)_i^2=0. \tag{11.61}$$

According to (11.36) and (11.56) $\{\hat{d}_k\}_{k\in K}$ is bounded and so $\{d_k\}$ is also bounded, namely

$$\|d_k\|\leqslant c\|g_k\|,\quad \forall k\in K,$$

for some $c<\infty$. It follows that the sequence $\{x_k-\mathscr{P}[x_k+d_k]\}_{k\in K}$ is also bounded. Assume that (11.57) does not hold. It means that there exists a subsequence of $\{x_k-\mathscr{P}[x_k+d_k]\}_{k\in K}$ which is convergent and its limit does not satisfy (11.57). Without the loss of generality we can assume that the sequence $\{x_k-\mathscr{P}[x_k+d_k]\}_{k\in K}$ itself does not converge to zero. Furthermore, because $\{x_k\}_{k\in K}$, $\{d_k\}_{k\in K}$ are bounded, we can also extract their convergent subsequences. We also assume the following (again without the loss of generality):

$$\lim_{k\in K}x_k=\bar{x},\quad \lim_{k\in K}d_k=\bar{d}.$$

Thus a contradiction to (11.57) implies that there exists at least one j such that

$$(\bar{x})_j>0\quad\text{and}\quad(\bar{d})_j\neq 0, \tag{11.62}$$

or

$$(\bar{x})_j=0\quad\text{and}\quad(\bar{d})_j>0. \tag{11.63}$$

For any index i we have

$$(d_k)_i(\mathscr{P}[x_k+\alpha_k d_k]-x_k)_i=\begin{cases}\alpha_k(d_k)_i^2 & \text{if } \alpha_k<\hat{t}_k^i\\ -(x_k)_i(d_k)_i & \text{if } \alpha_k\geqslant\hat{t}_k^i\end{cases}\geqslant 0 \tag{11.64}$$

since $(d_k)_i\leqslant 0$ if $\alpha_k\geqslant\hat{t}_k^i$.

Therefore, from (11.59), for each $i\in I_k^+$ we have

$$\lim_{k\in K}(d_k)_i(\mathscr{P}[x_k+\alpha_k d_k]-x_k)_i=0. \tag{11.65}$$

Let j satisfies (11.62)–(11.63). If $j\in I_k^+$ for infinitely many k and $\alpha_k<\hat{t}_k^j$ for infinitely many $k\in K_1\subset K$ then, because $(d_k)_j\neq 0$ for sufficiently large $k\in K_1$ by (11.62)–(11.63),

$$\lim_{k\in K_1}\alpha_k=0$$

from the first part of (11.64).

If $\hat{t}_k^j \leqslant \alpha_k$ and $j \in I_k^+$ for infinitely many $k \in K_2 \subset K$ then

$$\lim_{k \in K_2} (x_k)_j = 0$$

from the second part of (11.64), since $(d_k)_j \neq 0$ for sufficiently large $k \in K_2$ by (11.62)–(11.63). Therefore, by (11.63), $(d_k)_j > 0$ for sufficiently large $k \in K_2$. However, this, (10.7) imply that $\hat{t}_k^j = -\infty$ for sufficiently large $k \in K_2$ which contradicts the definition of K_2. This guarantees, that there exists an index $\bar{k} \in K$, such that $k \in K$ and $k \geqslant \bar{k}$ imply $k \in K_1$. Thus

$$\lim_{k \in K} \alpha_k = 0. \tag{11.66}$$

Now consider the case:

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k - g_k]\| \neq 0. \tag{11.67}$$

As in Chap. 10 we can show that

$$\begin{aligned} f(x_k) - f(x_k(\alpha)) &= g_k^T (x_k - x_k(\alpha)) \\ &\quad + \int_0^1 [g(x_k - t(x_k - x_k(\alpha))) - g_k]^T \, dt \\ &\quad \times [x_k - x_k(\alpha)] \geqslant g_k^T (x_k - x_k(\alpha)) \\ &\quad - \int_0^1 \|g(x_k - t(x_k - x_k(\alpha))) - g_k\| \, dt \\ &\quad \times \|x_k - x_k(\alpha)\|. \end{aligned} \tag{11.68}$$

Due to the Lipschitz continuity of g we also have

$$\begin{aligned} f(x_k) - f(x_k(\alpha)) &\geqslant g_k^T (x_k - x_k(\alpha)) \\ &\quad - \frac{1}{2} L \|x_k - x_k(\alpha)\|^2. \end{aligned} \tag{11.69}$$

Furthermore, we have (see Chap. 10), for some $C < \infty$,

$$\sum_{i \in I_k^+} |(x_k)_i - (x_k(\alpha))_i|^2 \leqslant C\alpha \sum_{i \in I_k^+} (g_k)_i \, [(x_k)_i - (x_k(\alpha))_i] \tag{11.70}$$

and

$$\sum_{i \notin I_k^+} (g_k)_i \, [(x_k)_i - (x_k(\alpha))_i] \geqslant -\alpha \sum_{i \notin I_k^+} (g_k)_i (d_k)_i, \tag{11.71}$$

$\forall \alpha \in [0, \bar{\varepsilon}/B]$. Here, $\bar{\varepsilon} > 0$ is such that

$$\lim_{k \in K} \varepsilon_k \geqslant \bar{\varepsilon}, \tag{11.72}$$

(which happens due to (11.67)) and $B < \infty$ is such that

$$|(d_k)_i| \leqslant B \tag{11.73}$$

for all i and $k \in K$.

Since we have

$$\sum_{i\notin I_k^+}(g_k)_i(d_k)_i \leqslant -\sum_{i\notin I_k^+}|(\hat{d}_k)_i|^2 \tag{11.74}$$

it follows that

$$\sum_{i\notin I_k^+}[(x_k)_i-(x_k(\alpha))_i] \geqslant \alpha\sum_{i\notin I_k^+}|(\hat{d}_k)_i|^2 \geqslant \alpha d_l\sum_{i\notin I_k^+}|(d_k)_i|^2 \tag{11.75}$$

(due to (11.56)). Furthermore, for any i we have

$$|(x_k)_i-(x_k(\alpha))_i| \leqslant \alpha(d_k)_i\,, \tag{11.76}$$

which implies

$$\sum_{i\notin I_k^+}|(x_k)_i-(x_k(\alpha))_i|^2 \leqslant \alpha^2\sum_{i\notin I_k^+}(d_k)_i^2. \tag{11.77}$$

Taking into account (11.68)–(11.75) we can show that

$$\begin{aligned} f(x_k)-f(x_k(\alpha)) \geqslant &\left(\alpha-\frac{\alpha^2L}{2}\right)\left(-\sum_{i\notin I_k^+}(d_k)_i^2\right)\\ &+\left(1-\frac{\alpha CL}{2}\right)\sum_{i\in I_k^+}(g_k)_i\left[(x_k)_i-(x_k(\alpha))_i\right]. \end{aligned} \tag{11.78}$$

If we take α which satisfy

$$0\leqslant\alpha\leqslant\frac{\bar{\varepsilon}}{B},\quad 1-\frac{1}{2}\alpha L\geqslant\mu,\quad 1-\frac{1}{2}\alpha LC\geqslant\mu,\quad \alpha\leqslant 1 \tag{11.79}$$

then we have

$$\begin{aligned} f(x_k)-f(x_k(\alpha)) \geqslant \mu\Bigg[&-\alpha\sum_{i\notin I_k^+}(\hat{d}_k)_i^2\\ &+\sum_{i\in I_k^+}(g_k)_i\left[(x_k)_i-(x_k(\alpha))_i\right]\Bigg]. \end{aligned} \tag{11.80}$$

Therefore, due to the generalized Armijo rule we have

$$\alpha_k\geqslant\beta\min\left[\frac{\bar{\varepsilon}}{B},\frac{2(1-\mu)}{L},\frac{2(1-\mu)}{CL},1\right] \tag{11.81}$$

for all $k\in K$. This contradicts (11.66). □

Having Lemma 11.2 we can prove the following theorem (as in Chap. 10 by x^1 we denote a vector defined by indices not belonging to I_k^+).

Theorem 11.1. *Suppose that assumptions of* Lemma 11.2 *are satisfied and that* $\{x_k\}$ *is generated by* Algorithm 11.1. *Moreover, assume that for any convergent subsequence* $\{x_k\}_{k\in K}$ *whose limit is not a critical point:*

(i) $\{\hat{\beta}_k\}$ *is such that*

$$\liminf_{k\to\infty}\left(\hat{\beta}_k\|d^1_{k-1}\|\right) \geqslant \nu_1 \liminf_{k\to\infty}\|g^1_k\|\,, \tag{11.82}$$

where ν_1 *is some positive constant.*

(ii) There exists a number ν_2 *such that* $\nu_2\|D_k\|\|D_k^{-1}\| \in (0,1)$ *and*

$$\left(g^1_k\right)^T d^1_{k-1} \leqslant \nu_2\|g^1_k\|\|d^1_{k-1}\|, \quad \textit{whenever} \quad \lambda_k \in (0,1). \tag{11.83}$$

Then $\lim_{k\to\infty} f(x_k) = -\infty$*, or every accumulation point of the sequence* $\{x_k\}$ *is a critical point.*

Proof. The proof follows the lines of the proof of Theorem 10.1 and for that reason it is omitted. □

The condition (11.83) is independent of the choice of the sequence $\{\hat{\beta}_k\}$ and can be substituted by another, more easily verifiable, condition.

Lemma 11.3. *Suppose that the assumptions of* Lemma 11.2 *are satisfied. Assume that* $\{x_k\}$ *is generated by* Algorithm 11.1 *and* $\{\hat{\beta}_k\}$ *satisfies (11.82) for any convergent subsequence whose limit is not a critical point. Then either*

$$\lim_{k\to\infty} f(x_k) = -\infty, \tag{11.84}$$

or for every convergent subsequence $\{x_k\}_{k\in K}$ *such that*

$$\lim_{k\in K}\|x_{k+1} - x_k\| = 0 \tag{11.85}$$

we have

$$\lim_{k\in K}\|x_k - \mathscr{P}[x_k - g_k]\| = 0. \tag{11.86}$$

Proof. The proof is essentially the same as the proof of Lemma 10.3. □

11.4 The Preconditioned Shortest Residuals Algorithm: Global Convergence

In this section we examine Algorithm 11.1 with the sequence $\{\beta_k\}$ defined by

$$\hat{\beta}_k = \frac{\left\|\hat{g}^1((\hat{x}^1_k,\hat{x}^2_{k-1}))\right\|^2}{\left|\left(\hat{g}^1((\hat{x}^1_k,\hat{x}^2_{k-1})) - \left(\hat{g}^c_{k-1}\right)^1\right)^T \hat{g}^1((\hat{x}^1_k,\hat{x}^2_{k-1}))\right|}, \tag{11.87}$$

where $\hat{g}^c_{k-1} = D_k^{-T} g_{k-1}$ and, as usual, by $\hat{x}^1$ we mean a vector defined by indices not belonging to I_k^+ and vector $\hat{g}^1((\hat{x}^1_k, \hat{x}^2_{k-1}))$ is as follows

$$\hat{g}^1((\hat{x}^1_k, \hat{x}^2_{k-1})) = \{\nabla_{\hat{x}_i} \hat{f}((\hat{x}^1_k, \hat{x}^2_{k-1}))\}_{i \notin I_k^+}.$$

We can prove the theorem.

Theorem 11.2. *If the assumptions of* Lemma 11.2 *are satisfied and* $\{x_k\}$ *is generated by* Algorithm 11.1*, then*

$$\lim_{k \to \infty} f(x_k) = -\infty,$$

or for any convergent subsequence $\{x_k\}_{k \in K}$

$$\lim_{k \in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0 \tag{11.88}$$

provided that:

(i) $\hat{\beta}_k$ *is given by (11.87).*
(ii) There exists $S < \infty$ *such that* $\alpha_k \leqslant S$, $\forall k$.

Proof. The proof follows the lines of the proof of Theorem 10.2 and lines of the proof of Theorem 8.2. □

The next result applies to problems in which stationary points satisfy sufficient optimality conditions together with the strict complementarity condition. We show that Algorithm 11.1 determines the set of bounds that are active at the solution in a finite number of iterations.

We can prove the following theorem.

Theorem 11.3. *Let* $\bar{x}$ *be a point at which sufficient optimality conditions together with strict complementarity condition are satisfied. Assume that* $\{x_k\}$ *is generated by* Algorithm 11.1 *and that the assumptions of* Theorem 11.2 *hold. Then there exists* $\delta > 0$ *such that if*

$$\|x_{\bar{k}} - \bar{x}\| \leqslant \delta$$

for some $\bar{k}$*, then* $\{x_k\}$ *converges to* $\bar{x}$ *and*

$$\mathscr{A}(\bar{x}) = \mathscr{A}(x_k) = I_k^+, \quad \forall k \geqslant \bar{k} + 1.$$

Proof. The proof is essentially the same as the proof of Theorem 10.3. □

11.5 The Limited Memory Quasi-Newton Method for Problems with Box Constraints

We can say that the special case of our algorithm is the quasi-Newton method. It is sufficient to assume that $\hat{\beta}_k \equiv 0$ and $\lambda_k \equiv 1$ in (11.37). Therefore, all results from the previous sections will hold for the quasi-Newton algorithm since they are valid (of course except of Theorem 11.2 for these particular values of λ_k and β_k).

In order to simplify the notation we use $B_k = D_k^T D_k$, then (11.31) corresponds to

$$B_k = \begin{bmatrix} B_k^+ & 0 \\ 0 & I_{n_k} \end{bmatrix}. \tag{11.89}$$

Algorithm 11.2. (The limited memory quasi-Newton algorithm for problems with box constraints)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\varepsilon > 0$, $\{B_k\}$, $B_k \in \mathscr{R}^{n\times n}$ a symmetric positive definite matrix, diagonal positive definite matrix M.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute

   $$d_1 = -g_1$$

   and set $k = 1$.
2. Find α_k according to the second generalized Armijo rule, or the second generalized Wolfe rule.
3. Compute

   $$w_{k+1} = x_{k+1} - \mathscr{P}[x_{k+1} - Mg_{k+1}]. \tag{11.90}$$

   If $\|w_{k+1}\| = 0$ then STOP, otherwise calculate

   $$\varepsilon_{k+1} = \min(\varepsilon, \|w_{k+1}\|) \tag{11.91}$$

   and find d_{k+1} by solving:

   $$B_{k+1}d_{k+1} = -g_{k+1}. \tag{11.92}$$

4. Substitute $\mathscr{P}[x_k + \alpha_k d_k]$ for x_{k+1}, increase k by one, go to Step 2.

Following the proof of Lemma 11.2 we come to the result.

Theorem 11.4. *If g is Lipschitz continuous with constant L, there exist b_l, b_u such that $0 < b_l < b_u < \infty$ and*

$$b_l\|u\|^2 \leqslant u^T B_k u \leqslant b_u\|u\|^2 \tag{11.93}$$

for all $u \in \mathscr{R}^n$, k *and* $\{x_k\}$ *is generated by* Algorithm 11.2, *then for any convergent subsequence* $\{x_k\}_{k\in K}$

$$\lim_{k\in K} \|x_k - \mathscr{P}[x_k - g_k]\| = 0. \tag{11.94}$$

The analog of Theorem 11.3 also holds for Algorithm 11.2.

Theorem 11.5. *Let* $\bar{x}$ *be a point at which the sufficient necessary optimality together with the strict complementarity condition hold. Assume that* $\{x_k\}$ *is generated by* Algorithm 11.2 *and that (11.93) holds. Then there exists* $\delta > 0$ *such that if*

$$\|x_{\bar{k}} - \bar{x}\| \leqslant \delta$$

for some $\bar{k}$, *then* $\{x_k\}$ *converges to* $\bar{x}$ *and*

$$\mathscr{A}(\bar{x}) = \mathscr{A}(x_k) = I_k^+, \quad \forall k \geqslant \bar{k}+1.$$

11.6 Numerical Algebra

In Sect. 2 we have shown that for a given nonsingular matrix D the preconditioned conjugate gradient algorithm is globally convergent.

In this section we present the method of obtaining the scaling matrix that is based on the compact representation of the L-BFGS matrix (cf. Chap. 5). We recall that on kth iteration the Hessian matrix is approximated with B_k:

$$B_k = B_k^0 - \begin{bmatrix} B_k^0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_k^0 S_k & L_k \\ L_k^T & -G_k \end{bmatrix}^{-1} \times \begin{bmatrix} S_k^T B_k^0 \\ Y_k^T \end{bmatrix} \tag{11.95}$$

where S_k, Y_k, L_k and G_k are defined by (8.42)–(8.44).

If we assume then that $B_k^0 = \gamma_k I_n$ and introduce matrices W_k and M_k defined by (8.45)–(8.46), then (8.41) can be written as

$$B_k = \gamma_k I_n - W_k M_k W_k^T. \tag{11.96}$$

Now we have to recall, that the scaling matrix will be applied only to the variables with indices not in the set I_k^+. Therefore, instead of full scaling matrix D_k (so that $B_k = D_k^T D_k$) it will be needed to evaluate the "reduced" scaling matrix D_k^+ corresponding to B_k^+, with B_k^+ defined analogically to (11.95), but in terms of S_k^+, Y_k^+, L_k^+ and G_k^+. The matrices S_k^+, Y_k^+ are submatrices of S_k, Y_k with rows with indices not belonging to I_k^+. In turn L_k^+ and G_k^+ are equivalent to L_k and G_k, but defined in terms of s^1 and y^1. Having said all that we rewrite (8.47) as

$$B_k^+ = \gamma_k^+ I_{n_k^+} - W_k^+ M_k^+ W_k^{+T} \tag{11.97}$$

$(n_k^+ = n - n_k)$.

In order to transform the matrix B_k^+ to the form $\left(D_k^+\right)^T D_k^+$ we do the QR factorization of the matrix W_k^+:

$$W_k^+ = Q_k R_k, \tag{11.98}$$

where Q_k is $n_k^+ \times n_k^+$ orthogonal matrix and R_k the $n_k^+ \times m$ matrix which has zero elements except the elements constituting the upper $m \times m$ submatrix (cf. Appendix C). Taking into account that $Q_k Q_k^T = I_{n_k^+}$ we can write (11.97) as

$$B_k^+ = Q_k \left[\gamma_k I_{n_k^+} - R_k M_k^+ R_k^T\right] Q_k^T. \tag{11.99}$$

As in Sect. 8.4 we can express B_k^+ as

$$B_k^+ = Q_k F_k F_k^T Q_k^T, \tag{11.100}$$

where F_k has the structure

$$F_k = \begin{bmatrix} C_k & 0 \\ 0 & \sqrt{\gamma_k} I_{n_k^+ - 2m} \end{bmatrix} \tag{11.101}$$

with C_k being the Cholesky factor of a certain matrix. The desired decomposition of the matrix B_k^+ is thus given by

$$B_k^+ = \left(D_k^+\right)^T D_k^+, \quad D_k^+ = F_k^T Q_k^T, \tag{11.102}$$

where the matrix D_k^+ is nonsingular provided that $s_i^{1T} y_i^1 > 0$ for $i = k-m, \ldots, k-1$.

Recall the relations (11.32)–(11.33) whose essential parts now can be written as

$$Q_k F_k \hat{g}_k^1 = g_k^1, \quad F_k^T Q_k^T d_k^1 = \hat{d}_k^1. \tag{11.103}$$

The details on solving these equations efficiently are provided in Sect. 8.4.

Some consideration also needs to be given to the problem of updating the scaling matrix D_k. For once, to ensure adequate efficiency of the algorithm the most recent information on the objective function needs to be incorporated into D_k, therefore the algorithm should not go for too many iterations with the scaling matrix unchanged – recalculation is performed every n_r iterations (n_r is a constant parameter of the algorithm). More importantly, though, each time when some new active constraints are identified (i.e. the set I_k^+ changes) the scaling matrix has to be calculated anew – identifying new constraints triggers the second condition for updating D_k. Such an approach guarantees that L-BFGS matrices are positive definite (provided that the second generalized Wolfe rule is applied), and in the case of strongly convex functions their norms are uniformly bounded and are uniformly separated from zero.

And finally, an implementation detail adds to the cases when D_k is recalculated. Namely, as an additional safeguard, such recalculation is forced in the pathological case when the line search routine fails. If that still does not help, or when the

calculation of D_k fails itself (e.g. on failure of Cholesky factorization) the algorithm restarts by resetting the scaling matrix to unit matrix.

Algorithm 11.5 refers to (11.92) in which $(B_k^+)^{-1}$ has to be calculated. After applying the Sherman–Morrison–Woodbury formula, (cf. (11.28)) the inverse of B_{k+1}^+ can be calculated from

$$\left(B_{k+1}^+\right)^{-1} = \frac{1}{\gamma_k} I_{n_{k+1}^+} + \frac{1}{\gamma_{k+1}^2} W_{k+1}^+ K_{k+1} M_{k+1} W_{k+1}^{+T}, \tag{11.104}$$

where an auxiliary matrix K_{k+1}, is defined as

$$K_{k+1} = I_{n_{k+1}^+} - \frac{1}{\gamma_{k+1}} M_{k+1} \left(W_{k+1}^{+T} W_{k+1}^+\right)^{-1}. \tag{11.105}$$

In the L-BFGS-B implementation this matrix is efficiently calculated and stored in an implicit form, described in more detail in [213]. The appropriate fragments of that code were used in our implementation of the presented algorithm.

11.7 Algorithm 11.1 and Algorithm 11.2 Applied to Problems with Strongly Convex Functions

Theorem 11.4 is useful in proving global convergence properties of Algorithm 11.2. Using Lemma 5.2 we can show the main result of the section.

Theorem 11.6. *Suppose that* $\{x_k\}$ *is generated by* Algorithm 11.2 *and that* f *is twice continuously differentiable on an open set* $\mathcal{N}$ *which contains the convex set* $\mathcal{M}$*:*

$$\mathcal{M} = \{x : \ f(x) \leqslant f(x_1)\}. \tag{11.106}$$

Assume also that

$$m_1 \|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant m_2 \|z\|^2$$

holds for all $x \in \mathcal{N}$, $z \in \mathcal{R}^n$ *and some* $0 < m_1 \leqslant m_2 < \infty$.

If B_k *is updated according to the limited memory BFGS formula with* $B_k^0 = \gamma_k I = (y_{k-1}^T y_{k-1})/(s_{k-1}^T y_{k-1}) I$. *Then* $\{x_k\}$ *converges to the minimizer of* f.

Proof. We provide the sketch of the proof. For the simplicity of presentation we assume that $B_k^+ = B_k$ – the general case would require only minor modifications of the given proof if we take into account that if condition (11.93) holds for $x \in \mathcal{R}^n$ it is also satisfied for $x \in \mathcal{R}^m$ where $m \in \{1, 2, \ldots, n\}$. We have

$$\lambda_n(B_k) \|x\|^2 \leqslant x^T B_k x \leqslant \lambda_1(B_k) \|x\|^2, \tag{11.107}$$

where $\lambda_1(B_k)$ and $\lambda_n(B_k)$ are the largest and the smallest eigenvalues of B_k.

Consider now the sequence $\{B_k\}$. From Lemma 5.2 there exists positive constants d_1 and d_2 such that

$$\lambda_1(B_k) \leqslant d_2, \quad \lambda_n(B_k) \geqslant d_1 \tag{11.108}$$

for all k. It follows from the fact that

$$\operatorname{trace}(B_k) = \sum_{i=1}^{n} \lambda_i(B_k), \quad \det(B_k) = \prod_{i=1}^{n} \lambda_i(B_k)\,, \tag{11.109}$$

where $\lambda_1(B_k), \ldots, \lambda_n(B_k)$ are eigenvalues of B_k.

Equation (11.108) implies that

$$d_1\|x\|^2 \leqslant x^T B_k x \leqslant d_2\|x\|^2 \tag{11.110}$$

for all k and $x \in \mathscr{R}^n$. This together with Theorem 11.4 imply the theorem's thesis. □

The analogous result holds for Algorithm 11.1.

Theorem 11.7. *Suppose that the assumptions of* Theorem 11.6 *are satisfied. Assume that* $\{x_k\}$ *is generated by* Algorithm 11.1 *and that the conditions (i) and (ii) of* Theorem 11.2 *hold. Then* $\{x_k\}$ *converges to the minimizer of* f.

11.8 Numerical Experiments

In order to verify the effectiveness of the proposed algorithm they have been tested on problems from the CUTE collection [13]. The aim was to try them on problems as large as possible, therefore choosing the problems with the `select` tool was done with the criteria, that the problem's dimension was at least equal to 400 or the number of variables had to be modifiable. In addition for the largest problems it was possible to set the dimension above 10000. For the purpose of more thorough testing the author decided to carry out, on them, an extra set of runs.

Two implemented codes Algorithm 11.1 and Algorithm 11.2 were compared against the benchmark for bound-constrained large scale optimization: the L-BFGS-B routine. For ensuring fair comparison the common parameters in both algorithms and the stopping criterion were unified. Both algorithms were to terminate on

$$\|\mathscr{P}[x_k - g_k] - x_k\|_\infty \leqslant 10^{-5}.$$

The line search routine implemented in the code was augmented with the steplength selection algorithm of Morè and Thuente described in Sect. 2.4 which was used to provide initial step in the second generalized Armijo rule. According to our discussion of the Armijo rule in Sect. 2.2, if an initial step is contained in some fixed compact set $[s_{smin}, s_{max}]$ ($s_{max} \geqslant s_{min} > 0$) then the Armijo rule can be used in algorithms to guarantee their global convergence.

The parameter ε in (10.14) was set to 0.01 while the diagonal matrix M had entries equal to 0.001.

Algorithm 11.1 was executed with the parameter m set to 3 and the parameter n_r to 12 – for the other values of parameters the performance of Algorithm 11.1 was worse if we take into account the balance between the CPU time and the number of function evaluations. We applied $\mu = 10^{-4}$ and $\eta = 0.9$ in line search rules. Both L-BFGS-B and Algorithm 11.2 were run with m equal to 5. For each problem the number of algorithm's iterations, the number of function and gradient evaluations and the computing time were measured.

The results of the comparison are presented graphically with the help of performance profiles. We discuss performance profiles for Algorithms 11.1–11.2 and L-BFGS-B as their main competitor. We show plots of the distribution functions for the performance ratios defined by two measures: the number of function evaluation and the computing time.

We also present absolute values obtained by all tested methods in Tables 11.1 and 11.2. The analysis of the presented performance profiles (Figs. 11.1–11.6 in which PCGB corresponds to Algorithm 11.1 and QNB to Algorithm 11.2) indicate that our new preconditioned conjugate gradient algorithms are competitive to the benchmark code L-BFGS-B. The remark applies to both versions of the proposed method, although Algorithm 11.2 seems to be superior.

Algorithm 11.1 requires more floating point operations per iteration in comparison to Algorithm 11.2. This is the result of using QR factorization of matrix W_k^+ as stated in (11.98) – its cost is proportional to $m^2 n_k^+$ and exceeds the cost of solving (11.103). On the other hand the cost of evaluating $(B_k^+)^{-1}$ according to (11.104)–(11.105) is proportional to mn_k^+. This analysis explains why the best results of Algorithm 11.1 are obtained when $m = 3$ (in contrast to $m = 5$ in the case of Algorithm 11.2) and $n_r = 12$.

In our opinion much of the efficiency of Algorithms 11.1–11.2 can be attributed to the way the box constraints are treated. It seems that its performance could be further enhanced by improving the line search procedure and the method for selecting ε-active constraints.

In [182] both Algorithm 11.1 and Algorithm 11.2 with the second generalized Wolfe rule are analyzed in detail. Furthermore, efficiency profiles of these algorithms are compared to L-BFGS-B procedure. The reported results are better than those obtained with the second generalized Armijo rule – these results will be published elsewhere.

11.9 Notes

The chapter presents a preconditioned conjugate gradient and a limited memory quasi-Newton algorithms for problems with box constraints. The enclosed numerical results show that both methods are competitive to L-BFGS-B which is the benchmark procedure for large scale problems with box constraints. The chapter

Table 11.1 Numerical results: Algorithm 11.1 and Algorithm 11.2 with the second generalized Armijo rule against L-BFGS-B

Problem	n	Algorithm 11.2			Algorithm 11.1			L-BFGS-B		
		CPU	IT	IF	CPU	IT	IF	CPU	IT	IF
BDEXP	5000	0.08	15	16	0.13	15	16	0.09	14	15
BIGGSB1	5000	29.98	5940	6103	10.39	2405	2513	19.95	4673	4825
CHARDIS0	400	0.09	4	19	0.09	4	19	0.02	2	4
CHENHARK	5000	8.13	2016	2070	6.71	1593	1786	8.13	2119	2190
CVXBQP1	10000	0.02	1	2	0.02	1	2	0.01	1	2
GRIDGENA	6218	1.41	159	163	1.72	170	200	FAIL	FAIL	FAIL
JNLBRNG1	10000	4.32	280	286	6	315	336	8.83	654	663
JNLBRNG2	10000	5.42	432	446	6.77	503	557	5.58	458	482
JNLBRNGA	10000	2.97	234	246	2.71	195	211	2.71	232	245
JNLBRNGB	10000	14.69	1286	1326	17.68	1476	1634	14.38	1310	1347
LINVERSE	1999	0.41	138	142	0.59	162	166	0.86	292	335
MCCORMCK	10000	0.25	16	18	0.71	39	56	0.18	2	3
MINSURFO	10406	6.83	347	349	9.41	392	404	5.19	287	308
NCVXBQP1	10000	0.02	1	2	0.02	1	2	FAIL	FAIL	FAIL
NOBNDTOR	10000	2.2	161	163	3.38	206	220	2.68	201	211
NONSCOMP	10000	0.57	33	44	0.61	32	43	0.53	47	51
OBSTCLAE	10000	2.52	154	157	2.84	152	159	5.82	456	462
OBSTCLAL	10000	1.49	102	104	1.48	94	100	1.43	122	126
OBSTCLBL	10000	1.52	90	94	2.01	99	103	2.35	182	186
OBSTCLBM	10000	1.43	85	88	1.79	94	98	1.44	106	110
OBSTCLBU	10000	1.68	99	102	1.76	91	96	1.74	136	142
PENTDI	5000	0.01	1	3	0.01	1	3	0.01	1	3
PROBPENL	500	0.01	1	3	0.01	1	3	0.01	2	3
SINEALI	1000	0.06	51	58	0.09	57	73	0.03	22	31
TORSION1	10000	2.15	149	152	2.26	133	140	2.01	162	166
TORSION2	10000	2.33	151	154	3.22	174	179	3.64	274	281
TORSION3	10000	0.79	65	67	0.95	75	82	0.86	80	84
TORSION4	10000	1.2	84	85	1.66	108	112	3.56	300	304
TORSION5	10000	0.35	31	33	0.37	32	40	0.28	29	33
TORSION6	10000	0.76	57	61	0.79	59	65	2.39	220	224
TORSIONA	10000	1.89	121	125	2.71	156	166	2.34	173	178
TORSIONB	10000	2.53	153	154	4.01	199	211	3.55	249	252
TORSIONC	10000	0.94	71	73	1.11	80	90	0.86	75	78
TORSIOND	10000	1.42	93	95	1.56	98	103	3.79	296	300
TORSIONE	10000	0.4	31	33	0.45	35	44	0.38	34	39
TORSIONF	10000	0.83	58	61	0.91	62	69	2.79	232	234
Total		101.7	12710	13097	96.93	9309	10101	108.42	13443	13917

also states conditions which have to be imposed on scaling matrices in order to guarantee the global convergence of the proposed algorithms. The numerical tests have been obtained with the limited memory BFGS matrices. It remains to be shown that other choices of scaling matrices such as resulting from DFP, or SR1 approximations could result in similar competitive performance of the presented algorithms.

The chapter gives also an overview of the limited memory quasi-Newton algorithm for problems with box constraints on the which the benchmark code for large scale bound constrained optimization is based – L-BFGS-B. It uses a different scheme for the identification of the active set at the solution. The numerical results presented in the chapter suggest that using quadratic programming subproblem to identify the active set of constraints is less efficient than the gradient projection method applied directly to the objective function.

Yet another approach to the identification of active constraints is introduced in [61]. Although the authors consider a general nonlinear optimization problems

Table 11.2 Numerical results: Algorithm 11.1 and Algorithm 11.2 with the second generalized Armijo rule against L-BFGS-B

Problem	n	Algorithm 11.2			Algorithm 11.1			L-BFGS-B		
		CPU	IT	IF	CPU	IT	IF	CPU	IT	IF
JNLBRNG1	15625	8.23	337	344	12.17	420	436	22.45	1065	1072
JNLBRNG2	15625	10.32	518	529	12.36	568	625	11.06	579	600
JNLBRNGA	15625	4.81	253	260	6	268	287	5.39	290	307
JNLBRNGB	15625	30.9	1703	1766	37.77	1989	2179	31.84	1756	1820
OBSTCLAE	15625	4.71	175	177	5.69	180	186	12.3	670	673
OBSTCLAL	15625	2.76	115	118	2.67	106	111	2.45	145	147
OBSTCLBL	15625	2.95	105	110	4.33	133	136	4.8	283	288
OBSTCLBM	15625	2.82	102	105	3.56	111	116	2.96	143	147
OBSTCLBU	15625	3.5	128	132	4.11	130	135	3.78	173	178
TORSION1	14884	3.08	138	140	4.38	172	179	3.86	207	213
TORSION2	14884	4.11	170	175	7.12	234	241	10.49	519	524
TORSION3	14884	1.75	90	92	1.78	89	95	1.19	73	76
TORSION4	14884	2.41	112	115	3.26	141	150	6.8	377	381
TORSION5	14884	0.79	44	47	0.7	40	46	0.56	38	40
TORSION6	14884	1.28	63	66	1.38	68	75	5.77	337	339
TORSIONA	14884	3.74	152	157	4.99	183	193	4.01	196	202
TORSIONB	14884	4.35	167	172	7.97	250	265	8.38	383	387
TORSIONC	14884	1.8	88	90	1.53	70	74	1.54	87	89
TORSIOND	14884	2.52	108	111	2.85	114	120	7.86	402	409
TORSIONE	14884	0.84	43	46	0.72	36	44	0.58	34	38
TORSIONF	14884	1.39	62	63	2.08	90	98	6	324	327
Total		99.06	4673	4815	127.42	5392	5791	154.07	8081	8257

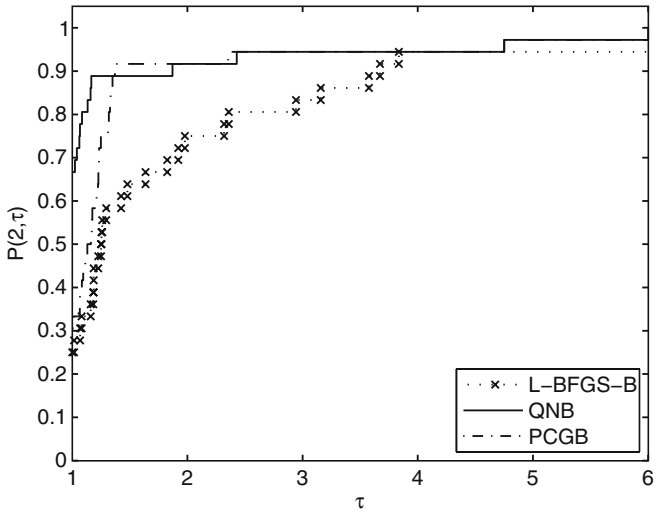

Fig. 11.1 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed Algorithm 11.1 and Algorithm 11.2 against L-BFGS-B code on problems shown in Table 11.1 – the second generalized Armijo rule was used in both algorithms

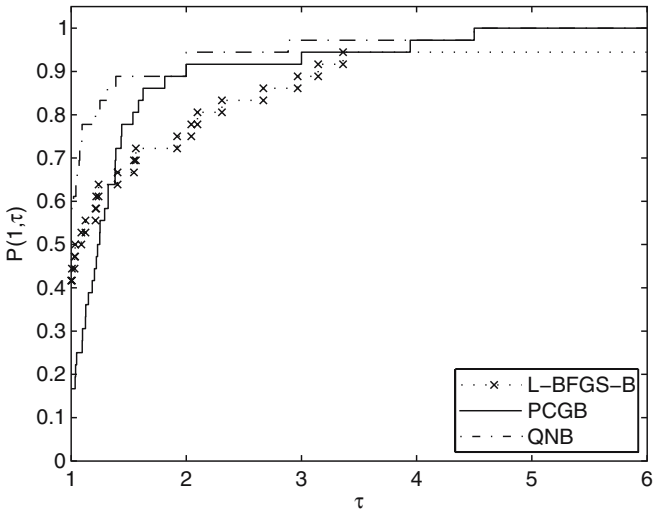

Fig. 11.2 Cumulative distribution ratio of CPU time: comparison of the proposed Algorithm 11.1 and Algorithm 11.2 against L-BFGS-B code on problems shown in Table 11.1 – the second generalized Armijo rule was used in both algorithms

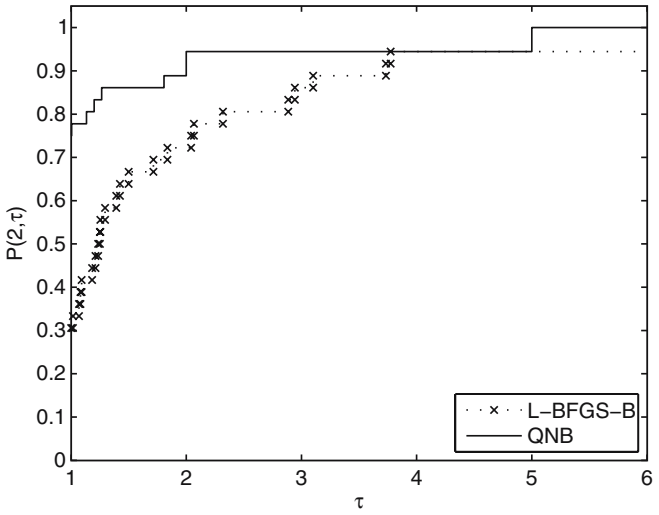

Fig. 11.3 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed Algorithm 11.2 against L-BFGS-B on problems shown in Table 11.1 – the second generalized Armijo rule was used in Algorithm 11.2

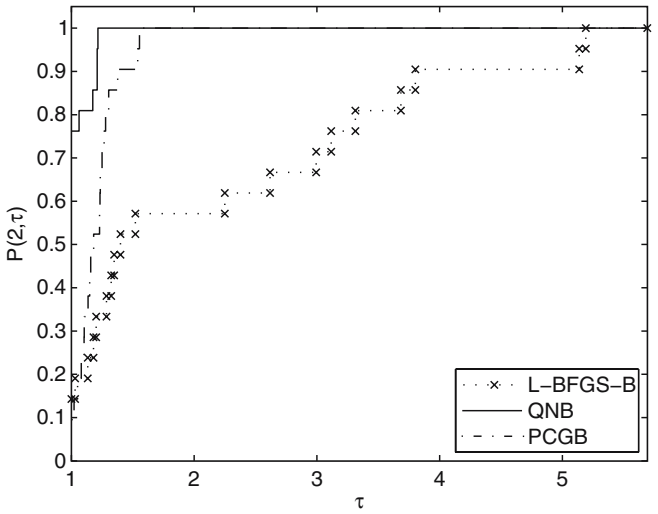

Fig. 11.4 Cumulative distribution ratio of the number of function evaluations: comparison of the proposed Algorithm 11.1 and Algorithm 11.2 against L-BFGS-B code on problems shown in Table 11.2 – the second generalized Armijo rule was used in both algorithms

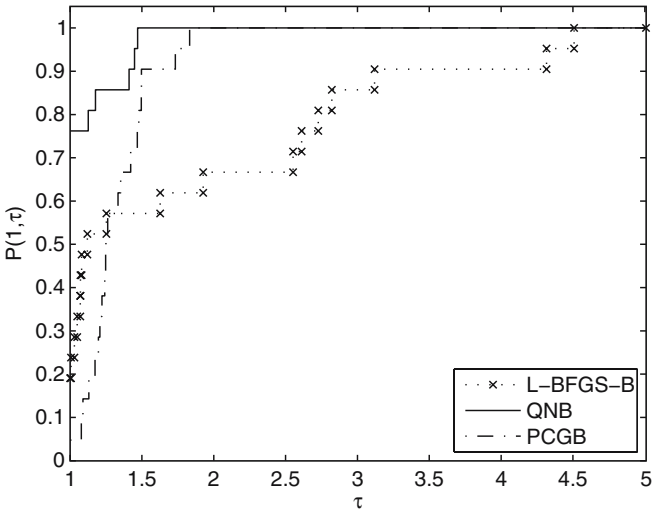

Fig. 11.5 Cumulative distribution ratio of CPU time: comparison of the proposed Algorithm 11.1 and Algorithm 11.2 against L-BFGS-B code on problems shown in Table 11.2 – the second generalized Armijo rule was used in both algorithms

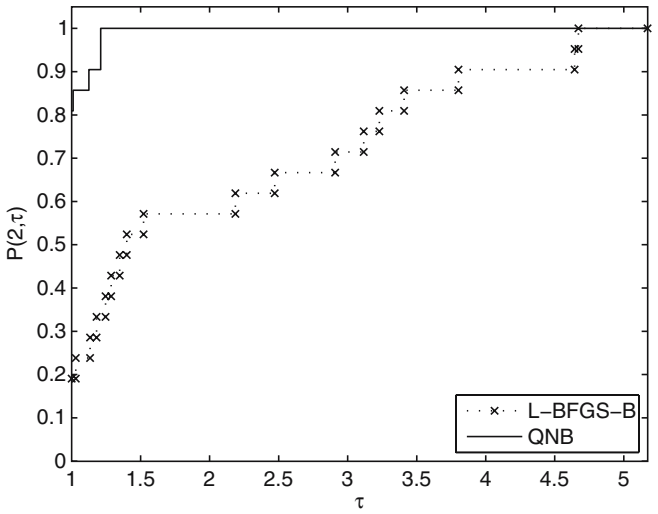

Fig. 11.6 Cumulative distribution ratio of the number of function evaluations – comparison of the proposed Algorithm 11.2 against L-BFGS-B code on problems shown in Table 11.2: the second generalized Armijo rule was used in Algorithm 11.2

the proposed algorithms can be adopted to large scale problems with only box constraints.

Consider the optimization problem

$$\min_{x \in \mathscr{R}^n} f(x) \tag{11.111}$$

subject to the constraints

$$h(x) \geqslant 0, \tag{11.112}$$

where $h : \mathscr{R}^n \to \mathscr{R}^m$. Suppose that $\bar{x}$ is the solution to problem (11.111)–(11.112). Then, $\bar{x}$ has to satisfy the Karush–Kuhn–Tucker necessary optimality conditions:

$$\nabla f(\bar{x}) - \nabla h(\bar{x})\bar{\lambda} = 0 \tag{11.113}$$

$$h(\bar{x}) \geqslant 0 \tag{11.114}$$

$$\bar{\lambda} \geqslant 0 \tag{11.115}$$

$$\bar{\lambda}^T h(\bar{x}) = 0, \tag{11.116}$$

where $\bar{\lambda} \in \mathscr{R}^m$ are Lagrange multipliers corresponding to $\bar{x}$. Lagrange multipliers corresponding to the point $\bar{x}$, which we assume that is an isolated stationary point,

do not have to be unique. Therefore, we introduce the set of Lagrange multipliers associated with $\bar{x}$ and the set of all pairs $(\bar{x},\bar{\lambda})$ which solve (11.113)–(11.116):

$$\Lambda = \left\{\bar{\lambda} :\ (\bar{x},\bar{\lambda}) \text{ solves (11.113)–(11.116)}\right\} \tag{11.117}$$

$$\mathscr{K} = \left\{(\bar{x},\bar{\lambda}) :\ \bar{\lambda} \in \Lambda\right\}. \tag{11.118}$$

Lagrange multipliers can be used to identify the active set of constraints at a stationary point. For example, the set

$$I(x,\lambda) = \{i = 1,\ldots,m :\ h_i(x) \leqslant \lambda_i\} \tag{11.119}$$

is equal to $\mathscr{A}(\bar{x})$ if (x,λ) is in the small neighborhood of $(\bar{x},\bar{\lambda})$ and the strict complementarity condition holds at $(\bar{x},\bar{\lambda})$ satisfying (11.113)–(11.116) (i.e. $\bar{\lambda}_i > 0$ for all i such that $h_i(\bar{x}) = 0$).

If the strict complementarity condition does hold then we can only assume that

$$I(x,\lambda) \subseteq \mathscr{A}(\bar{x}). \tag{11.120}$$

For that reason, in [61], the concept of an *identification function* is introduced. If ρ is an identification function then the set

$$I^{if}(x,\lambda) = \{i = 1,\ldots,m :\ h_i(x) \leqslant \rho(x,\lambda)\} \tag{11.121}$$

is such that

$$I^{if}(x,\lambda) = \mathscr{A}(\bar{x}) \tag{11.122}$$

if (x,λ) is close to $(\bar{x},\bar{\lambda})$.

In order (11.121)–(11.122) to hold a function ρ must satisfy certain requirements which are specified in the following definition.

Definition 11.1. A function $\rho : \mathscr{R}^n \to \mathscr{R}^m$ is called an identification function for $\mathscr{K}$ if:

(i) ρ is continuous on $\mathscr{K}$.
(ii) $(\bar{x},\bar{\lambda}) \in \mathscr{K}$ implies $\rho(\bar{x},\bar{\lambda}) = 0$.
(iii) if $(\bar{x},\bar{\lambda}) \in \mathscr{K}$ then

$$\lim_{\substack{(x,\lambda)\to(\bar{x},\bar{\lambda}) \\ (x,\lambda)\notin\mathscr{K}}} \frac{\rho(x,\lambda)}{\operatorname{dist}[(x,\lambda),\mathscr{K}]} = +\infty\,, \tag{11.123}$$

where $\operatorname{dist}[x,A]$ is the Euclidean distance of a point x to a set A.

The condition (iii) postulates faster convergence of (x,λ) to the set $\mathscr{K}$ than ρ to zero.

Among several identification functions proposed in [61] there is one which can have application to many optimization problems. Suppose that

$$L(x,\lambda) = f(x) - \lambda^T h(x) \tag{11.124}$$

and

$$\Phi(x,\lambda) = \begin{bmatrix} \nabla_x L(x,\lambda) \\ \min[h(x),\lambda] \end{bmatrix}, \tag{11.125}$$

then

$$\rho(x,\lambda) = \sqrt{\|\Phi(x,\lambda)\|}\,, \tag{11.126}$$

under some assumptions (see [61] for details), is an identification function. In particular, if $\bar{x}$ is a stationary point for problem (11.111)–(11.112) at which second order sufficient condition for optimality is satisfied and the set $\mathscr{K}$ is compact then ρ stated in (11.125)–(11.126) is an identification function. The elaborate proof of that is given in [61] – only the requirement (ii) of the definition is easy to check. If $(x,\lambda) \in \mathscr{K}$ then $\Phi(x,\lambda) = 0$.

The crucial theorem stated in [61] is as follows.

Theorem 11.8. *Suppose that ρ is an identification function for $\mathscr{K}$. Then, for any $\bar{\lambda} \in \Lambda$ there exists $\varepsilon(\bar{\lambda}) > 0$ such that*

$$I^{if}(x,\lambda) = \mathscr{A}(\bar{x}), \quad \forall (x,\lambda) \in \mathscr{B}\left((\bar{x},\bar{\lambda}),\varepsilon(\bar{\lambda})\right), \tag{11.127}$$

where $\mathscr{B}(x,\varepsilon)$ is the open Euclidean ball with radius ε and centered at x.

Identification functions are means for transforming, locally, problems with inequality constraints to problems with only equality constraints which are much easier to deal with – the difficult combinatorial aspect of problems is removed. These functions can be applied to problems with box constraints giving rise to new algorithms which would be alternatives to the methods discussed in this chapter. Several computational aspects would have to be resolved to make these alternatives competitive to L-BFGS-B, or to Algorithm 11.2. However, these new algorithms would have the property of identifying active constraints at a solution in a finite number of iterations even when complementarity condition does not hold since identification functions discussed in [61] posses this property. More on the use of identifications functions (different from those described in [61]) in algorithms for large scale problems with box constraints can be found in [62, 63, 93] and in more general setting in [147].

Chapter 12
Preconditioned Conjugate Gradient Based Reduced-Hessian Methods

12.1 Introduction

We end our overview of conjugate gradient algorithms by presenting a new approach to building preconditioning matrices.

In Chap. 4 we have introduced an approach to the quasi-Newton equation

$$B_k d_k = -g_k \tag{12.1}$$

based on factorizing the matrix B_k:

$$B_k^{-1} = D_k D_k^T, \tag{12.2}$$

where the matrix D_k is easily updated using vectors $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. We have discussed (12.1)–(12.2) to show that the factorization of the kind (12.2) is possible and could be used in a preconditioned version of the method of shortest residuals presented in Chap. 8.

The factorization (12.2) as discussed in Chap. 4 was introduced by Powell [170]. It is not suitable for large scale problems since D_k is a dense matrix so its direct use in the method of shortest residuals is not recommended. But (12.1)–(12.2) were introduced to improve scaling properties of quasi-Newton matrices. It appears that there are several factorizations of the kind (12.2) associated with the BFGS updating matrix, and among them we have representations of a matrix B_k which not only lead to more robust versions of quasi-Newton algorithms but can also be used when n is large. In addition these representations can be applied in a preconditioned version of the method of shortest residuals. Notice that factorization (12.2) can be used directly in the method of shortest residuals – it corresponds to the change of variables

$$D\hat{x} = x$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

and the equations which have to be solved at every iteration of Algorithm 8.1 are

$$\hat{g}_{k+1} = D_{k+1}^T g_{k+1},$$
$$d_{k+1} = D_{k+1}\hat{d}_{k+1},$$

instead of (8.12), (8.15).

12.2 BFGS Update with Column Scaling

Before presenting a new factorization of B_k which could be used efficiently in a preconditioned version of the method of shortest residuals we give some insight into the representation (12.1) and its possible use in improving the BFGS scheme of the Hessian approximation.

Following Siegel [197] consider a general nonlinear function f which satisfies:

(1) The curvature of f is of order of one for all directions.
(2) The current approximation to its Hessian matrix, denoted by B_k, has eigenvalues $0 < \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_{n-1} \leqslant \lambda_n$ with the corresponding normalized eigenvectors $\{v_1, v_2, \ldots, v_n\}$.
(3) The greatest eigenvalue is large while the others are close to unity – $\lambda_i \approx 1$, for $i = 1, \ldots, n-1$ and $\lambda_n \gg 1$.
(4) $s_k^T y_k > 0$.
(5) $s_k = d_k$.
(6) v_n is not parallel to g_k.

We will show that the curvature of B_{k+1} on the space spanned by $v_1, \ldots, v_{n-1}$ is still close to one and that v_n is close to the eigenvector of the greatest eigenvalue of B_{k+1}.

First, we need the representation of d_k in terms of λ_i, v_i, $i = 1, \ldots, n$ and g_k. From the definition we have

$$B_k v_i = \lambda_i v_i, \quad i = 1, \ldots, n$$

and since v_i are normalized we can write

$$v_i^T B_k v_i = \lambda_i, \quad i = 1, \ldots, n,$$

from which B_k can be represented as

$$B_k^{-1} = V \Lambda^{-1} V^T \tag{12.3}$$

with

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \tag{12.4}$$

and $V = [v_1\ v_2\ \cdots v_n]$.

Using the representation of B_k^{-1} and the quasi-Newton equation (12.1) we can evaluate d_k as

$$\begin{aligned} d_k &= -V\Lambda^{-1}V^T g_k \\ &= -[v_1\ v_2 \cdots v_n] \begin{bmatrix} \lambda_1^{-1} & & & \\ & \lambda_2^{-1} & & \\ & & \ddots & \\ & & & \lambda_n^{-1} \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix} g_k \\ &= -[v_1\ v_2 \ldots v_n] \begin{bmatrix} \lambda_1^{-1} v_1^T g_k \\ \lambda_2^{-1} v_2^T g_k \\ \vdots \\ \lambda_n^{-1} v_n^T g_k \end{bmatrix} \\ &= -\sum_{i=1}^{n} \lambda_i^{-1} \left(v_i^T g_k\right) v_i \,. \end{aligned} \tag{12.5}$$

Taking into account (12.5) we can estimate the curvature of B_{k+1} along the direction v_n. To this end we make another assumption that $v_i^T g_k$, $i = 1, \ldots, n$, are of the same magnitude. Then, from the BFGS update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k} \tag{12.6}$$

and (12.5) the following relation holds (since we assume (3) and (4))

$$\begin{aligned} v_n^T B_{k+1} v_n &= v_n^T B_k v_n - \frac{\left(v_n^T B_k s_k\right)^2}{s_k^T B_k s_k} + \frac{\left(v_n^T y_k\right)^2}{s_k^T y_k} \\ &\geqslant \lambda_n - \frac{\left(v_n^T g_k\right)}{\sum_{i=1}^n \lambda_i^{-1} \left(v_i^T g_k\right)^2} \\ &= \frac{\lambda_n \left(\sum_{i=1}^n \lambda_i^{-1} \left(v_i^T g_k\right)^2\right) - \left(v_n^T g_k\right)^2}{\sum_{i=1}^n \lambda_i^{-1} \left(v_i^T g_k\right)^2} \\ &= \frac{\sum_{i=1}^{n-1} \lambda_i^{-1} \left(v_i^T g_k\right)^2}{\sum_{i=1}^n \lambda_i^{-1} \left(v_i^T g_k\right)^2} \lambda_n \\ &= \frac{\sum_{i=1}^{n-1} \lambda_i^{-1} \left(v_i^T g_k\right)^2}{\sum_{i=1}^n \lambda_i^{-1} \left(v_i^T g_k\right)^2} v_n^T B_k v_n. \end{aligned} \tag{12.7}$$

Due to our assumption (3)

$$\sum_{i=1}^{n-1} \lambda_i^{-1} \left(v_i^T g_k\right)^2 \approx \sum_{i=1}^{n} \lambda_i^{-1} \left(v_i^T g_k\right)^2,$$

thus we have

$$v_n^T B_{k+1} v_n \approx v_n^T B_k v_n \tag{12.8}$$

and it remains to be shown that v_n corresponds to the greatest eigenvalue of B_{k+1}. Indeed,

$$\begin{aligned} v_i^T B_{k+1} v_i &= v_i^T B_k v_i - \frac{\left(v_i^T B_k s_k\right)^2}{s_k^T B_k s_k} + \frac{\left(v_i^T y_k\right)^2}{s_k^T y_k} \\ &\leqslant \lambda_i + \frac{\left(v_i^T y_k\right)^2}{s_k^T y_k} \end{aligned} \tag{12.9}$$

and due to the assumption (1), B_{k+1} has $n-1$ eigenvalues which are essentially smaller than the eigenvalue to which v_n corresponds.

Since the situation described above can repeat at iteration $k+2$ the analysis shows that if we start from ill-conditioned approximation of the Hessian matrix then for several iterations the quasi-Newton algorithm minimizes f on the subspace which does not include vector v_n (notice that for large value of λ_n the eigenvector v_n little contributes to d_k – see (12.5)).

That potential inefficiency of the BFGS based quasi-Newton algorithm can occur if a starting point is badly chosen. Variable metric algorithms are invariant under linear transformations and the quasi-Newton algorithm is considered in the space of $\hat{x}$ variables which are related to x variables via the equation $\hat{x} = D_1^{-1} x$. If D_1 has a large eigenvalue, then the algorithm in the space of $\hat{x}$ variables has B_1^{-1} near singular and the above analysis applies to this case.

Another example is given in [197]. Suppose that function f is flat. In this case initial $B_1 = I$ is a very poor approximation to the Hessian matrix and at the next iteration B_2^{-1} will have one large eigenvalue and the others close to one (notice the difference between the considered situation and the case discussed so far – B_1^{-1} has one small eigenvalue and the others are close to one). Therefore, the BFGS algorithm first minimizes f on the subspace spanned by the eigenvector corresponding to the greatest eigenvalue, then the minimization takes place with respect to the space spanned by two eigenvectors, then by three and so on. The algorithm progresses very slowly towards the minimum point of f.

Powell's idea for improving the behavior of the BFGS algorithm stems from the observation that the columns of D_k of small lengths can lead to the presented inefficiencies. Powell [170] proposes to rescale the lengths of columns of the matrix D_k satisfying (12.2). If the rescaling is applied to matrix D_k and then the modified matrix is used to calculate d_k we cannot guarantee that directions $d_1, d_2, \ldots, d_k$ are conjugate (in the case of strongly convex quadratic function f). However, Powell shows that his BFGS update formula with arbitrary column scaling still leads to finite termination when applied to quadratics.

Powell's updating scheme for the matrix D_k is briefly described in Chap. 4. We remind the multiplicative form of the BFGS updating formula (cf. (4.47)–(4.48)):

$$B_{k+1}^{-1} = \left(I - s_k z_k^T\right) B_k^{-1} \left(I - z_k s_k^T\right)$$

with

$$z_k = \frac{B_k s_k}{\sqrt{(s_k^T y_k)(s_k^T B_k s_k)}} + \frac{y_k}{s_k^T y_k}.$$

The corresponding update formula for D_{k+1} is then given by

$$D_{k+1} = \left(I - s_k z_k^T\right) D_k. \tag{12.10}$$

Powell takes formula (12.10) as the starting point for his analysis. First, suppose that d_k^1 (d_k^i means the ith column of the matrix D_k) is a multiple of s_k (which we denote by $d_k^1 \parallel s_k$) which implies, from the secant equation and the stipulated condition (which follows directly from (12.2) with k replaced by $k+1$):

$$D_{k+1}^T B_{k+1} D_{k+1} = I,$$

that

$$d_{k+1}^1 = \frac{s_k}{\sqrt{s_k^T B_k s_k}} = \frac{s_k}{\sqrt{s_k^T y_k}}. \tag{12.11}$$

In order to derive the formula for d_{k+1}^i, $i = 2,3,\ldots,n$ observe that $d_k^i \parallel s_k$ and $D_k^T B_k D_k = I$ imply that $s_k^T B_k d_k^i = 0$, $i = 2,3,\ldots,n$. Therefore, taking into account (12.3)–(12.4) (and (12.11)) leads to

$$d_{k+1}^i = \begin{cases} s_k / \sqrt{s_k^T y_k}\,, & i = 1 \\ d_k^i - \left(\frac{y_k^T d_k^i}{s_k^T y_k}\right) s_k\,, & i = 2,3,\ldots,n. \end{cases} \tag{12.12}$$

The formula (12.12) has been derived under the assumption that $d_k^1 \parallel s_k$. Powell shows in [170] that it is still valid for a general case if s_k can be expressed by some vector t_k through the relation

$$s_k = D_k t_k. \tag{12.13}$$

Notice that if d_k is determined by a quasi-Newton iteration then $t_k = -\alpha_k D_k^T g_k$ will satisfy (12.13):

$$\begin{aligned} s_k &= \alpha_k d_k = -\alpha_k B_k^{-1} g_k = -\alpha_k D_k D_k^T g_k \\ &= D_k t_k, \end{aligned}$$

where α_k is the step-length.

Having vector s_k defined by (12.13) the procedure for obtaining the update rule for D_{k+1} can be reduced to that outlined for the case $d_k^1 \parallel s_k$. To this end build the orthogonal matrix Ω_k^T which transforms vector t_k to the vector which has zero components except the first one. Since Ω_k is an orthogonal matrix it does not change the Euclidean norm, thus we must have

$$\Omega_k^T t_k = \pm \|t_k\| e_1, \tag{12.14}$$

where e_1 is the unit vector.

Powell suggests using the sequence of Givens rotations Ω_k^i, $i = 1, \ldots, n-1$ to transform t_k to the vector $\pm\|t_k\|e_1$ – each matrix Ω_k^i introduces a zero to the vector t_k. However, a Householder matrix could also be used to accomplish the transformation (12.14) – cf. Appendix C.

Having the relation (12.14) we can use the matrix

$$\bar{D}_k = D_k \Omega_k$$

in relation (12.2) instead of D_k since Ω_k is orthogonal:

$$\begin{aligned} B_k^{-1} &= D_k D_k^T \\ &= \bar{D}_k \Omega_k^T \Omega_k \bar{D}_k^T \\ &= \bar{D}_k \bar{D}_k^T \end{aligned}$$

and, from (12.14) by taking into account $\Omega_k \Omega_k^T = I$,

$$\bar{d}_k^1 = \bar{D}_k e_1 = D_k \Omega_k e_1 = \pm \frac{D_k t_k}{\|t_k\|} = \pm \frac{s_k}{\|t_k\|}$$

thus $\bar{d}_k^1$ is a multiple of s_k.

Now we can follow the analysis which resulted in (12.12) to obtain the analogous result

$$d_{k+1}^i = \begin{cases} s_k / \sqrt{s_k^T y_k}\,, & i = 1 \\ \bar{d}_k^i - \left(\frac{y_k^T \bar{d}_k^i}{s_k^T y_k} \right) s_k\,, & i = 2, 3, \ldots, n\,. \end{cases} \tag{12.15}$$

A quasi-Newton algorithm with the factorization of B_k given by (12.2), where D_k is updated according to (12.15), in exact arithmetic is equivalent to the BFGS method. One advantage of using the quasi-Newton method based on the factorization (12.2) is that if $\|d_{k+1}^1\| \ll \|d_k^1\|$ happens it is through the scaling of s_k instead of by cancellation. However, that approach further developed can remove some deficiencies of the BFGS algorithm. First, as we have already mentioned, columns d_k^i of small lengths can lead to the inefficiency described by (12.5)–(12.9). Powell thus observes that if we rescale these columns the numerical behavior of the quasi-Newton method significantly improves. Although such rescaling is the departure from the BFGS variable metric algorithm it does not destroy the finite termination on a quadratic [170].

Powell proposes the following column rescaling scheme. First, the matrix obtained in (12.15) we denote by $\hat{D}_{k+1}$ while D_{k+1} is given by

$$d_{k+1}^i = \begin{cases} \hat{d}_{k+1}^i\,, & i = 1 \\ \hat{d}_{k+1}^i \max\left[1, \sigma_k / \|\hat{d}_{k+1}^i\|\right] \hat{d}_{k+1}^i\,, & i = 2,3,\ldots,n\,. \end{cases} \tag{12.16}$$

with

$$\sigma_k = \max_{1 \leqslant l \leqslant k+1} \|\hat{d}_{k+1}^l\|. \tag{12.17}$$

Another scaling rule is proposed by Siegel [197] who suggests using

$$\sigma_k = \|\hat{d}_{k+1}^1\|$$

instead of (12.17) and shows that the BFGS algorithm with his modification to column scaling can only be linearly convergent when applied to quadratics with the Wolfe line search rules provided that line search rules parameters μ, η satisfy $2\eta + \mu > 1$. This negative result deepens the Powell's analysis and motivates Siegel to propose another strategy for the column scaling.

Siegel [197] refers to the important property of Powell's algorithm when applied to strongly convex quadratic functions $f = 1/2x^T Qx - b^T x$. Suppose that we start with any nonsingular matrix D_1, then after k iterations we have the relations

$$\begin{gathered} d_k^i \parallel s_{k-1}, \quad 2 \leqslant i \leqslant k \\ \left(d_k^i\right)^T Q d_k^j = \delta_{ij}, \quad 2 \leqslant i, j \leqslant k \end{gathered}$$

and

$$\left(d_k^i\right)^T Q d_k^j = 0, \quad 1 \leqslant i \leqslant k < j \leqslant n\,, \tag{12.18}$$

where δ_{ij} is the Kronecker symbol. Furthermore, the minimum of f on the space $\mathscr{H}_k$ spanned by $d_k^1, d_k^2, \ldots, d_k^k$ is achieved and the curvature of f on the space $\mathscr{H}_k$ is reflected in D_k, i.e. for any $v \in \mathscr{H}_k$ the following holds

$$v^T Q v = v^T \left(D_k D_k^T\right)^{-1} v. \tag{12.19}$$

The property (12.19) is destroyed if columns $d_k^1, \ldots, d_k^k$ are rescaled.

On the other hand, for any v belonging to the space spanned by $d_k^{k+1}, d_k^{k+2}, \ldots, d_k^n$, from (12.18), we have

$$v^T Q d_k^i = 0, \quad 1 \leqslant i \leqslant k$$

thus v belongs to the conjugate complement of $\mathscr{H}_k$. Siegel concludes that if we want to avoid destroying the curvature information gained so far we shouldn't scale columns spanning $\mathscr{H}_k$.

Based on the above observations, made under the assumptions that f is a strictly convex function and line searches are exact, Siegel proposes to scale, at iteration k, only columns with indices greater than some index l. According to his analysis at iteration $k+1$, columns of D_{k+1} with indices greater than $l+1$ should be scaled. However, he allows to scale also the $l+1$ column arguing that if d_k is a linear combination of the first l columns of matrix D_k and that it is appropriate to scale column $l+1$ of D_k then it would also be appropriate to scale the $l+1$ column of matrix D_{k+1} – see [197] for details. Using the same arguments he suggests to define the direction d_k according to the rule

$$d_k = -\check{D}_k \check{D}_k^T g_k, \tag{12.20}$$

where $\check{D}_k$ is the matrix composed of the first l columns of D_k:

$$\check{D}_k = [d_k^1 \; d_k^2 \; \ldots \; d_k^l]. \tag{12.21}$$

Direction d_k given by (12.20) is applied only when sufficient reduction of the quadratic approximation of f (denoted by r) is guaranteed. Siegel introduces the criterion which helps to choose between directions specified by (12.20)–(12.21) and by

$$d_k = -D_k D_k^T g_k.$$

Notice that

$$\begin{aligned} r_k &= -d_k^T g_k - \frac{1}{2} d_k^T B_k d_k \\ &= g_k^T D_k D_k^T g_k - \frac{1}{2} g_k^T D_k D_k^T B_k D_k D_k^T g_k \\ &= \frac{1}{2} \|t_k\|^2, \end{aligned}$$

where t_k is as specified in (12.13). On the other hand, we can show that for d_k defined by (12.20) the following holds

$$\begin{aligned} \check{r}_k &= g_k^T \check{D}_k \check{D}_k g_k - \frac{1}{2} g_k^T \check{D}_k \check{D}_k^T B_k \check{D}_k \check{D}_k^T g_k \\ &= \frac{1}{2} \|\check{t}_k\|^2, \end{aligned}$$

where $\check{t}_k = [t_k^1 \; t_k^2 \; \ldots \; t_k^l]^T$. If we denote by $\tilde{t}_k$ the other part of vector t_k, i.e. $t_k^T = [\check{t}_k^T \; \tilde{t}_k^T]$, then d_k stated in (12.20) is preferred over that specified by the full matrix D_k if

$$\|\tilde{t}_k\|^2 < \delta \|\check{t}_k\|^2, \tag{12.22}$$

where $\delta \in (0,1)$ – Siegel suggests using $\delta = 0.1$.

Eventually, the essential part of the Siegel algorithm can be read as follows. Suppose that at the kth iteration

$$\begin{aligned} D_k &= [d_k^1\ d_k^2\ \cdots d_k^{l_k}\ d_k^{l_k+1}\ \cdots d_k^n] \\ &= [\check{D}_k\ \tilde{D}_k] \end{aligned}$$

has been determined, then calculate

$$\begin{aligned} \check{t}_k &= -\check{D}_k^T g_k \\ \tilde{t}_k &= -\tilde{D}_k g_k. \end{aligned}$$

If $\|\tilde{t}_k\|^2 < \delta\|\check{t}_k\|^2$ then set

$$t_k = [\check{t}_k^T\ 0^T]^T, \quad d_k = \check{D}_k t_k \quad \text{and} \quad l_{k+1} = l_k,$$

otherwise set

$$t_k = [\check{t}_k^T\ \tilde{t}_k^T]^T, \quad d_k = \check{D}_k \check{t}_k + \tilde{D}_k \tilde{t}_k \quad \text{and} \quad l_{k+1} = l_k + 1.$$

Siegel proves in [197] that his algorithm with the Wolfe line search rules is globally and superlinearly convergent for strongly convex functions whose Hessian matrices are Lipschitz continuous.

Another column scaling strategy is proposed in [112]. Lalee and Nocedal follow original work by Powell, however they make several modifications to the Powell's scheme in order to have a globally convergent algorithm for strictly convex functions. Their starting point is a factorization of the matrix B_k in the form

$$B_k = V_k V_k^T,$$

where V_k is a lower Hessenberg matrix. Then, since V_k differs from a triangular matrix only by the elements lying just above the diagonal, it is easy to construct an orthogonal matrix Q_k which transforms V_k to the lower triangular matrix $L_k = V_k Q_k$. Since Q_k is orthogonal we have

$$B_k = V_k Q_k Q_k^T V_k = L_k L_k^T.$$

It means that d_k can be easily evaluated by using forward and backward substitutions:

$$d_k = -L_k^{-T} L_k^{-1} g_k.$$

Next, the BFGS update to the matrix B_k is made

$$B_{k+1} = W_k W_k^T \tag{12.23}$$

in such a way that the matrix W_k is lower Hessenberg. To this end vector r_k is determined by $r_k = L_k^T s_k$ and then an orthogonal matrix Ω_k is found such that

$$\|r_k\|e_1 = \Omega_k^T r_k. \tag{12.24}$$

The obvious proposition for Ω_k is the product of at most n Givens rotations. Eventually, columns of the matrix W_k are evaluated by

$$w_k^i = \begin{cases} y_k/\sqrt{y_k^T s_k}, & i = 1 \\ L_k \Omega_k e_i, & i = 2,3,\ldots,n. \end{cases} \tag{12.25}$$

In order to verify (12.23) notice that if $B_k = L_k L_k^T$ then B_{k+1} as the BFGS update of B_k should satisfy

$$B_{k+1} = L_k L_k^T - \frac{L_k L_k^T s_k s_k^T L_k L_k^T}{\|L_k^T s_k\|^2} + \frac{y_k y_k^T}{y_k^T s_k}.$$

But, from (12.25), we have

$$\begin{aligned} W_k W_k^T &= \sum_{i=1}^{n} w_k^i \left(w_k^i\right)^T \\ &= \frac{y_k y_k^T}{y_k^T s_k} + \sum_{i=1}^{n} L_k \Omega_k e_i e_i^T \Omega_k^T L_k^T \end{aligned} \tag{12.26}$$

and, from (12.24) and the definition of r_k,

$$\begin{aligned} L_k L_k^T &= L_k \Omega_k \Omega_k^T L_k^T \\ &= \sum_{i=1}^{n} L_k \Omega_k e_i e_i^T \Omega_k^T L_k^T \\ &= \frac{L_k L_k^T s_k s_k^T L_k L_k^T}{\|L_k^T s_k\|^2} + \sum_{i=2}^{n} L_k \Omega_k e_i e_i^T \Omega_k^T L_k^T. \end{aligned} \tag{12.27}$$

Solving for the sum on the right of (12.27) and substituting it in (12.26) gives us (12.23). We can check that W_k is a lower Hessenberg matrix.

Next, some columns of W_k are scaled giving the matrix V_{k+1}. Two parameters $\sigma_k > 0$ and $\rho_k > 0$ such that $\sigma_k \leqslant \rho_k$ are used to define scaling factors c_k^i:

$$c_k^i = \begin{cases} \sigma_k/\sqrt{\|w_k^i\|} & \text{if } \|w_k^i\| < \sigma_k \\ \rho_k/\|w_k^i\| & \text{if } \|w_k^i\| > \rho_k \\ 1 & \text{otherwise.} \end{cases} \tag{12.28}$$

The coefficients c_k^i, $i = 1,\ldots,n$ define a scaling diagonal matrix $C_k = \text{diag}[c_k^1,\ldots,c_k^n]$ which applied to matrix W_k gives the lower Hessenberg matrix V_{k+1}:

$$V_{k+1} = W_k C_k.$$

Lalee and Nocedal show in [112] that if the sequences $\{\sigma_k\}$ and $\{\rho_k\}$ are bounded – for all k

$$\sigma_k \leqslant \sigma_{\max}, \quad \rho_k \geqslant \rho_{\min},$$

for some positive constants $\sigma_{\max}$, $\rho_{\min}$, then $\{x_k\}$ is globally convergent for strictly convex functions and the iterates converge linearly.

Furthermore, if $\sigma_{\max} = \lambda_{\min}$, $\rho_{\min} = \lambda_{\max}$, where $\lambda_{\min}$, $\lambda_{\max}$ are the smallest and largest eigenvalues of the Hessian matrix at the solution, then the rate of convergence is superlinear.

In general $\lambda_{\min}$, $\lambda_{\max}$ are not known in advance, thus Lelee and Nocedal propose adaptive scheme for $\{\sigma_k\}$ and $\{\rho_k\}$ which also leads to an efficient algorithm. Denote by I_k and J_k sets of indices of columns to be scaled

$$\begin{aligned} I_k &= \left\{ i \in \{1,\ldots,n\} : \|w_k^i\| < \sigma_k \right\}, \\ J_k &= \left\{ i \in \{1,\ldots,n\} : \|w_k^i\| > \rho_k \right\}. \end{aligned} \tag{12.29}$$

It can be shown that if

$$\begin{aligned} \sigma_k^2 &= \frac{1}{n}\left((n - |I_{k-1}|)\,\sigma_{k-1} + \sum_{i \in I_{k-1}} \|w_{k-1}^i\|^2 \right), \\ \rho_k^2 &= \frac{1}{n}\left((n - |J_{k-1}|)\,\rho_{k-1} + \sum_{i \in J_{k-1}} \|w_{k-1}^i\|^2 \right), \end{aligned} \tag{12.30}$$

then $\{x_k\}$ converges superlinearly to the minimum point of f provided that f is strictly convex and its Hessian is Lipschitz continuous.

Lalee–Nocedal algorithm has been presented with the matrix B_k factorization: $B_k = W_k W_k^T$. However, it could also be stated when the factorization is applied to the inverse of B_k, i.e. when $B_k^{-1} = S_k S_k^T$ – the analysis of this version of Lalee–Nocedal algorithm would be similar to that carried out in this section.

12.3 The Preconditioned Method of Shortest Residuals with Column Scaling

Incorporating column scaling to conjugate gradient algorithms could follow two strategies discussed in Chap. 8. We remind that strategy S1 assumes that during some number of iterations, say m, positive definite matrices B_k are updated according to the BFGS rule and then applied in n_r preconditioned conjugate gradients iterates. In strategy S2 every n_r iterations the quasi-Newton matrix is built from the previous vectors $s_{n_r}, s_{n_r-1}, \ldots, s_{n_r-m+1}$, $y_{n_r}, y_{n_r-1}, \ldots, y_{n_r-m+1}$ and then used for the fixed number of iterations in a preconditioned conjugate gradient algorithm as in the strategy S1. Column scaling, which improves the Hessian approximation, is used while updating B_k, or when the quasi-Newton approximation is built if the strategy S2 is used. Convergence analysis of both versions of a preconditioned conjugate

gradient algorithm with column scaling refers to the same properties of matrices approximating the Hessians therefore we present it in the case of the strategy S1.

If we use strategy S1 we have the following algorithm.

Algorithm 12.1. (The preconditioned method of shortest residuals with column scaling - strategy S1)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\{\hat{\beta}_k\}$, m, n_r, $m < n_r$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, compute

$$d_1 = -g_1$$

and set $k = 1$, $r = 0$.

2. Find a positive number α_k satisfying the modified Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
3. If $\|g_{k+1}\| = 0$ then STOP.
If $k+1 < (r+1)n_r$ and $k+1 > rn_r + m$ go to Step 4.
If $k+1 = (r+1)n_r$ then set $V_{k+1} = \sqrt{\gamma_{k+1}}I$, $Q_{k+1} = I$ and increase r by one.
If $k+1 = (r+1)n_r + m$ then substitute L_{k+1} for L_r. Evaluate d_{k+1} by solving the equations

$$\begin{aligned} L_r z &= -g_{k+1} \\ L_r^T d_{k+1} &= z \end{aligned}$$

and go to Step 4.
Compute d_{k+1} by solving the equations

$$\begin{aligned} L_{k+1} z &= -g_{k+1} \\ L_{k+1}^T d_{k+1} &= z \end{aligned}$$

where L_{k+1} is such that $V_{k+1}Q_{k+1} = L_{k+1}$, $V_{k+1} = W_{k+1}C_{k+1}$, $B_{k+1} = W_{k+1}W_{k+1}^T$, B_{k+1} is obtained from $L_k L_k^T$ by the BFGS formula and C_{k+1} is given by (12.28). Go to Step 5.

4. Solve the equations with respect to $\hat{d}_{k+1}$ and d_{k+1}:

$$\begin{aligned} L_r \hat{g}_{k+1} &= g_{k+1} \\ \hat{d}_{k+1} &= -\mathbf{Nr}\{\hat{g}_{k+1}, -\hat{\beta}_{k+1}\hat{d}_k\} \\ L_r^T d_{k+1} &= \hat{d}_{k+1} \end{aligned}$$

5. Increase k by one and go to Step 2.

The global convergence of Algorithm 12.1 is a straightforward conclusion of Theorem 8.2 and analog of Lemma 5.2 which we state below.

Lemma 12.1. *Assume that:*

(i) The level set

$$\mathscr{M} = \{x \in \mathscr{R}^n : \ f(x) \leqslant f(x_1)\}$$

is convex.

(ii) There exist positive constants m and M such that

$$m\|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M\|z\|^2$$

for all $x \in \mathscr{M}$ *and* $z \in \mathscr{R}^n$.

Let α_k *be chosen according to the Wolfe conditions. If matrix* B_k *is obtained from the matrix* B_k^0 *by at most m updates using the scheme:*

(1) B_{k+1} *is the BFGS update of the matrix* $L_k L_k^T$.
(2) $B_{k+1} = W_{k+1} W_{k+1}^T$.
(3) $V_{k+1} = W_{k+1} C_{k+1}$ *where* C_{k+1} *is the diagonal matrix based on* σ_{k+1} *and* ρ_{k+1} *such that* $0 < \rho_{\min} \leqslant \rho_{k+1} < \infty$, $0 < \sigma_{k+1} \leqslant \sigma_{\max} < \infty$ *(cf. (12.28)).*
(4) $V_{k+1} Q_{k+1} = L_{k+1}$ *where* Q_{k+1} *is an orthogonal matrix and* L_{k+1} *upper triangular.*

Then, there exist positive constants c_1 *and* c_2 *such that*

$$\begin{aligned} \operatorname{trace}(L_k L_k^T) &\leqslant c_1, \\ \det(L_k L_k^T) &\geqslant c_2. \end{aligned}$$

Proof. The proof heavily borrows from the analysis presented in the proofs of Lemma 5.1, Lemma 5.2 and Lemma 3.3 in [112] and so it is omitted. □

Under the assumptions of Lemma 12.1 eigenvalues of matrices $L_k L_k^T$ are uniformly bounded and are greater than zero. The direct consequence of that is the following theorem.

Theorem 12.1. *Suppose that* $\{x_k\}$ *is generated by* Algorithm 12.1 *and:*

(i) The assumptions of Lemma 12.1 *hold.*
(ii) β_k *is given by (8.33).*

Then $\{x_k\}$ *converges to the minimizer of f.*

Algorithm 12.1 is not intended for problems with many variables. Since matrices L_k are in general dense the cost of one iteration is of order n^2 floating point operations. In the next section we introduce an algorithm which could be efficiently applied to large scale unconstrained problems.

12.4 Reduced-Hessian Quasi-Newton Algorithms

The essence of the approach described in Sect. 1 is the accumulation of curvature information on the certain gradient subspace. In Chap. 1 we show that this property is the characteristic of conjugate gradient algorithms. Furthermore, the subspaces on which it takes place are identical to those associated with the BFGS method applied to a quadratic. Powell's conjugate direction scaling of the approximation of the inverse of the Hessian matrix (cf. (12.2)) tries to build the subspace on which we have good curvature information and at the same time to rescale vectors which define the subspace in order to increase the efficiency of quasi-Newton iterates. Each iterate of the Powell algorithm expands the subspace by fixing the first column of the matrix D_{k+1} to a multiple of s_k (which is defined by newly calculated g_{k+1}) and by appropriately modifying the other columns of D_k which contain the already accumulated curvature information.

Siegel's approach goes one step further. He observes that it is often beneficial not to expand the subspace at a given iterate if the substantial reduction of the objective function can be achieved using the curvature information available on the current subspace. He also notices that the search direction is the sum of two vectors: one with the scale of estimated derivatives and the other with the scale of the initial approximate Hessian. He suggests to rescale only the second vector using newly available approximate curvature.

That interpretation of the BFGS method with column scaling and the accompanying Lalee–Nocedal convergence analysis suggest that the method has the potential for developing efficient algorithms for large scale problems. It must be stressed that Algorithm 12.1 is not intended for these problems since the cost of its iteration is of order n^2 floating point operations (linear equations of lower, or upper triangular dense matrices have to be solved) and the memory requirement is also of the same order.

Siegel in [195] and independently Fenelon [65] adopt the conjugate-direction scaling algorithm to large scale problems by referring directly to the concept of the gradient space on which the curvature information is accumulated by quasi-Newton iterates. In the case of large scale problems the number of iterations performed by quasi-Newton algorithms before the approximate solution is found is usually much smaller than the number of variables. This means that the dimension of the gradient space which is relevant in quasi-Newton algorithms for large scale problems is significantly smaller than n and can be represented by its basis consisting of few, say m, vectors. Since the curvature information concerns only the small dimensional space it can be represented by a symmetric matrix of low m dimension. This implies that memory requirements could be reduced to m vectors of dimension n plus some fixed number of n dimensional vectors. Besides that we would need the square matrix of dimension $m \times m$ which could be used to provide second order approximation of f on the subspace – as the result the cost of one iteration could be estimated as of order kn (if we consider only the dominant cost of updating the basis of the subspace). As far as the orthogonal space to the gradient space is concerned, as in the

conjugate-direction scaling algorithm, the curvature information on it is represented by a diagonal matrix which can be updated at every iteration.

Siegel in [195] (see also [65]) shows that the gradient space can be associated with the space spanned by the gradients evaluated up to the kth iteration – $\mathscr{G}_k = \text{span}\{g_1, g_2, \ldots, g_k\}$. For further discussion crucial is the following result which we present in more general setting than it is stated in [195] – its proof is an obvious modification of the proof of Lemma 5.1 given in [196].

Lemma 12.2. *Consider an algorithm applied to minimize function f whose second order derivatives are continuous and which has the kth iteration defined as*

$$d_k = -\lambda_k B_k^{-1} g_k + (1-\lambda_k)\,\beta_k d_{k-1}, \quad \lambda_k \in [0,1], \tag{12.31}$$

where β_k is some number, B_k is obtained by applying the BFGS update to the matrix B_{k-1} with $s_k^T y_k > 0$ and beginning from the diagonal matrix $B_1 = \sigma I$. Then, $d_k \in \mathscr{G}_k$ and if $z \in \mathscr{G}_k$, $w \in \mathscr{G}_k^{\perp}$ we have

$$B_k z \in \mathscr{G}_k, \quad B_k w = \sigma w,$$
$$H_k z \in \mathscr{G}_k, \quad H_k w = \frac{1}{\sigma} w,$$

where H_k is the BFGS matrix for an approximation of the inverse of the Hessian matrix beginning from $H_1 = \frac{1}{\sigma} I$. Here, $\mathscr{G}_k^{\perp}$ is the subspace orthogonal to $\mathscr{G}_k$.

Lemma 12.2 is the basis of the reduced-Hessian algorithm introduced in [80]. Suppose that $h_k = \dim(\mathscr{G}_k)$ and that the matrix $T_k \in \mathscr{R}^{n \times h_k}$ has columns which form the basis of $\mathscr{G}_k$. Since $\text{rank}(T_k) = h_k$ there exist matrices $Q_k \in \mathscr{R}^{n \times n}$ and $R_k \in \mathscr{R}^{n \times h_k}$ such that Q_k is an orthogonal matrix and R_k is a nonsingular upper triangular matrix and the following holds (cf. Appendix C)

$$T_k = Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix}.$$

If we divide Q_k into full rank matrices Z_k and W_k: $Q_k = [Z_k \; W_k]$ and transform the space of x variables to the space of x^Q by $x = Q_k x^Q$ then the matrix B_k in the new space is stated as

$$Q_k^T B_k Q_k = \begin{bmatrix} Z_k^T B_k Z_k & 0 \\ 0 & \sigma I_{n-h_k} \end{bmatrix} \tag{12.32}$$

and the gradient g_k as

$$Q_k^T g_k = \begin{bmatrix} Z_k^T g_k \\ 0 \end{bmatrix}. \tag{12.33}$$

$Z_k B_k Z_k$ is known in the literature (cf. e.g. [146]) as a reduced approximate Hessian while $Z_k^T g_k$ as a reduced gradient.

In the space of x^Q variables the quasi-Newton iterate has the direction $\bar{q}_k = Q_k^T d_k$ (notice that $Q_k^T = Q_k^{-1}$) defined by the equation

$$\left(Q_k^T B_k Q_k\right) \bar{q}_k = \left(Q_k^T B_k Q_k\right) Q_k^T d_k = Q_k^T g_k. \tag{12.34}$$

Equations (12.32)–(12.34) imply that $q_k \in \mathscr{R}^{h_k}$, related to d_k by $d_k = Z_k q_k$, satisfies

$$Z_k^T B_k Z_k q_k = -Z_k^T g_k. \tag{12.35}$$

Equation (12.35) can be solved by doing the Cholesky factorization of $Z_k^T B_k Z_k = C_k^T C_k$ and then by solving two sets of equations

$$C_k^T z_k = -Z_k^T g_k \tag{12.36}$$

$$C_k q_k = z_k. \tag{12.37}$$

Equations (12.36) are solved with respect to z_k at the cost of forward substitution proportional to h_k^2 floating point operations [87] and then q_k is calculated from (12.37) by backward substitution also at the cost proportional to h_k^2. Since the Cholesky factorization requires the number of floating point operations of order h_k^3 the total cost associated with the q_k calculation is proportional to h_k^3. Then, we have to evaluate d_k from the relation $d_k = Z_k q_k$ at the cost proportional to $h_k n$ thus if building the matrix Z_k can be done at a reasonable cost the approach based on the reduced Hessian is justified.

Suppose that at the $(k+1)$th iteration the matrix Z_k has to be updated to Z_{k+1}. To this end we evaluate

$$\rho_{k+1} = \left\| \left(I - Z_k Z_k^T\right) g_{k+1} \right\| \tag{12.38}$$

and if $\rho_{k+1} = 0$ then $g_{k+1} \in \mathscr{G}_k$ – notice that in this case $g_{k+1} = Z_k Z_k^T g_{k+1}$ and g_{k+1} lies in range(Z_k) thus g_{k+1} does not have components outside $\mathscr{G}_k$ so we assume $h_{k+1} = h_k$. In the case $\rho_{k+1} \neq 0$ we take z_{k+1} satisfying

$$\rho_{k+1} z_{k+1} = \left(I - Z_k Z_k^T\right) g_{k+1} \tag{12.39}$$

as the new basis vector in $\mathscr{G}_{k+1}$. Indeed, from the definition of ρ_{k+1}, $\|z_{k+1}\| = 1$ and

$$\begin{aligned} Z_k^T z_{k+1} &= Z_k^T \left(I - Z_k Z_k^T\right) g_{k+1} / \rho_{k+1} \\ &= \left(Z_k^T g_{k+1} - Z_k^T Z_k Z_k^T g_{k+1}\right) / \rho_{k+1} \\ &= 0 \end{aligned}$$

since the columns of Z_k are orthogonal. The approximate number of floating point operations needed to transform Z_k to Z_{k+1} can be estimated as $2h_k n$.

The Cholesky factor C_k has to be updated when B_k is updated to B_{k+1} and (or) Z_k is transformed to Z_{k+1}. According to Gill and Leonard this should be performed in two steps. First, if g_{k+1} is used to expand the space $\mathscr{G}_k$, the matrix

$$Z_{k+1}^T B_k Z_{k+1}$$

is formed (if $g_{k+1} \in \mathscr{G}_k$ we assume $Z_{k+1} = Z_k$) by taking into account that $Z_{k+1} = [Z_k \; z_{k+1}]$, $Z_k^T z_{k+1} = 0$ ($z_{k+1} \in \mathscr{G}_k$) and Lemma 12.2:

$$\begin{aligned} Z_{k+1}^T B_k Z_{k+1} &= \begin{bmatrix} Z_k^T B_k Z_k & Z_k^T B_k z_{k+1} \\ z_{k+1}^T B_k Z_k & z_{k+1}^T B_k z_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} Z_k^T B_k Z_k & 0 \\ 0 & \sigma \end{bmatrix}. \end{aligned} \tag{12.40}$$

On the basis of (12.40) the Cholesky factor of $Z_{k+1}^T B_k Z_{k+1} - \bar{C}_k$ is expressed by

$$\bar{C}_k = \begin{bmatrix} C_k & 0 \\ 0 & \sqrt{\sigma} \end{bmatrix} \tag{12.41}$$

if $h_{k+1} = h_k + 1$ and $\bar{C}_k = C_k$ for $h_{k+1} = h_k$.

In the next step the matrix $\bar{C}_k^T \bar{C}_k$ is updated according to the BFGS rule. In [84] (see also [54]) it is shown that C_{k+1} is also the upper triangular factor in the QR factorization of the matrix $\bar{C}_k + w_1 w_2^T$ with

$$w_1 = \frac{1}{\|\bar{C}_k \bar{s}_k\|} \bar{C}_k \bar{s}_k \tag{12.42}$$

$$w_2 = \frac{1}{\sqrt{\bar{y}_k^T \bar{s}_k}} \bar{y}_k - \frac{1}{\|\bar{C}_k \bar{s}_k\|} \bar{C}_k^T \bar{C}_k \bar{s}_k \tag{12.43}$$

$$\bar{s}_k = Z_{k+1}^T s_k \tag{12.44}$$

$$\bar{y}_k = Z_{k+1}^T y_k. \tag{12.45}$$

This upper triangular factor can be computed from $\bar{C}_k$ in $4h_k^2 + O(h_k)$ operations using Givens rotations (or in $3h_k^2 + O(h_k)$ using a modified rotation as described in [79]).

Summing up, at every iteration of the reduced-Hessian algorithm we solve (12.36)–(12.37) getting q_k and $d_k = Z_k g_k$, then Z_k is expanded to Z_{k+1} which in turn is used to update the Cholesky factor of the reduced Hessian from C_k to C_{k+1} (which is used at the next iteration). It is shown in [80] that iteration k requires approximately $(3h_k + 1)n + 4h_k^2 + O(h_k)$ floating point operations when $h_k < n$ and $7n^2 + O(n)$ otherwise. Thus, the cost per iteration measured in flops is comparable to that associated with a quasi-Newton method which performs the Cholesky factorization to the matrix B_k: $B_k = R_k^T R_k$, and to a quasi-Newton algorithm based on the conjugate-direction identity: $D_k^T B_k D_k = I$. However, it must be stressed that when $k \geqslant n$ the number of flops is greater than in the method based on the Cholesky factorization applied directly to the matrix B_k – cf. [80] and the remarks below.

The benefits of using the reduced-Hessian algorithm are thus attributed to the better memory management, especially when the method is applied to large scale problems, and to the significant reduction of the number of flops per iteration in its version developed for large scale problems and based on the subspace $\mathscr{G}_m$ with $m \ll n$. This version of the reduced-Hessian method is presented in the next

section, this section is concluded with the presentation of the basic reduced-Hessian algorithm.

Algorithm 12.2. (The basic reduced-Hessian quasi-Newton algorithm)

Parameters: $\sigma > 0$, $0 < \mu < \eta < 1$.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$. Set

$$Z_1 = \frac{g_1}{\|g_1\|}, \quad C_1 = [\sqrt{\sigma}],$$

   solve (12.36)–(12.37) for d_k and set $k = 1$.
2. Find α_k which satisfies the Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \mu \alpha_k g_k^T d_k$$
$$g(x_k + \alpha_k d_k)^T d_k \geqslant \eta g_k^T d_k.$$

3. If $\|g_{k+1}\| = 0$ then STOP. Otherwise calculate ρ_{k+1} according to (12.38) and if $\rho_{k+1} \neq 0$ expand Z_k to $Z_{k+1} = [Z_k \; z_{k+1}]$ where z_{k+1} solves (12.39). If $\rho_{k+1} = 0$ then assume $Z_{k+1} = Z_k$.
   If $\rho_{k+1} > 0$ then evaluate $\bar{C}_k$ according to (12.41), otherwise substitute C_k for $\bar{C}_k$. Calculate C_{k+1} as the upper triangular factor in the QR factorization of the matrix $\bar{C}_k + w_1 w_2^T$ with w_1 and w_2 defined by (12.42)–(12.45).
4. Substitute $x_k + \alpha_k d_k$ for x_{k+1}, increase k by one and go to Step 2.

In the exact arithmetic Algorithm 12.2 generates the same sequence of the consecutive points x_k as the standard BFGS method applying the Wolfe line search rules with the same parameters and taking as the initial approximation to the Hessian matrix $B_1 = \sigma I$. Thus, it is not proven efficiency which makes Algorithm 12.2 a viable alternative to implementations of quasi-Newton methods based on the Cholesky factorization of the Hessian matrix approximation, or on a conjugate-direction scaling approach. The decomposition of the Hessian matrix approximation represented by (12.40) enables us to introduce modifications which increase the efficiency of a quasi-Newton algorithm. Following [80] we describe two enhancements of Algorithm 12.2 which, according to the results presented in [80], lead to the method which is very competitive to the other implementations of quasi-Newton algorithms. Both of these enhancements have been already discussed while presenting several variants of the BFGS approach with column scaling.

The first modification is related to the Siegel's proposition not to expand the subspace on which the curvature information is built if, according to some test – (12.22) in the case of Siegel's algorithm, the better choice is to improve the curvature on the current subspace. Gill and Leonard introduce *lingering* iterations on a manifold $\mathscr{M}(\mathscr{U}_k)$. The manifold $\mathscr{M}(\mathscr{U}_k)$ has the properties: the subspace $\mathscr{U}_k$ is a subset of $\mathscr{G}_k$ ($\mathscr{U}_k \subset \mathscr{G}_k$); the *lingering direction* d_k forces x_{k+1} to remain on $\mathscr{M}(\mathscr{U}_k)$. The other iteration, which is used by the algorithm with lingering iterations, is called *RH*

iteration and has the property that its *RH direction* is computed using the reduced Hessian associated with the entire matrix Z_k.

According to [80] the use of RH direction, or lingering direction shouldn't be influenced by accepting (or rejecting) the current gradient g_k to expand (or not) the current space $\mathscr{G}_k$. This means that some subspace $\mathscr{U}_k \subset \mathscr{G}_k$ must be associated with lingering iterations. Gill and Leonard propose to use the subspace generated by directions d_k. Their choice is motivated by the following lemma which can be proved following [80] and [121].

Lemma 12.3. *Suppose that Z_k is the result of orthogonalizing the gradients $g_1, \ldots, g_k$ generated by the method based on the direction formula (12.31) where B_k is defined as in* Lemma 12.2. *If D_k and G_k are matrices of search directions and gradients associated with iterations at which a gradient is accepted, then there are nonsingular upper triangular matrices S_k and $\bar{S}_k$ such that*

$$G_k = Z_k S_k, \quad D_k = Z_k \bar{S}_k.$$

Lemma 12.3 can be used to show that Z_k provides also the basis for the space $\mathscr{D}_k$ of search directions $d_1, \ldots, d_k$.

Theorem 12.2. *The spaces $\mathscr{G}_k$ and $\mathscr{D}_k$ generated by gradients and search directions of the methods based on (12.31) with B_k defined as in* Lemma 12.2 *are identical.*

Proof. The definition of d_k implies that

$$\mathscr{D}_k \subseteq \mathscr{G}_k \tag{12.46}$$

and Lemma 12.3 that

$$\mathscr{G}_k = \text{range}(D_k) \subseteq \mathscr{D}_k.$$

This together with (12.46) give the thesis. □

It remains to show how Theorem 12.2 could be used to partition the matrix Z_k to facilitate lingering. Suppose that $U_k \in \mathscr{R}^{n \times l_k}$ forms the basis for the space $\mathscr{U}_k$ associated with lingering. Then

$$Z_k = [U_k \; Y_k]$$

and U_k has columns spanning the space $\mathscr{U}_k$. If $\check{D}_k \in \mathscr{R}^{n \times l_k}$ denotes the matrix of RH directions computed up to iteration k and $\check{G}_k \in \mathscr{R}^{n \times (h_k - l_k)}$ denotes the matrix of the subset of accepted so far gradients, then

$$\begin{aligned} \left[\check{D}_k \; \check{G}_k\right] &= Z_k S_k \\ &= [U_k \; Y_k] \begin{bmatrix} S_k^U & S_k^{UY} \\ 0 & S_k^Y \end{bmatrix}. \end{aligned}$$

The intermediate problem associated with calculating d_k (cf. (12.36)) and reflecting the partitioning of Z_k into U_k and Y_k can be stated as

$$\begin{bmatrix} C_k^U & C_k^{UY} \\ 0 & C_k^Y \end{bmatrix}^T \begin{bmatrix} z^U \\ z^Y \end{bmatrix} = - \begin{bmatrix} U_k^T g_k \\ Y_k^T g_k \end{bmatrix} \tag{12.47}$$

with $C_k^U \in \mathscr{R}^{l_k \times l_k}$, $C_k^{UY} \in \mathscr{R}^{(h_k - l_k) \times l_k}$, $C_k^Y \in \mathscr{R}^{(h_k - l_k) \times (h_k - l_k)}$. The solution to (12.47) has two components z_k^U and z_k^Y satisfying

$$\begin{aligned} \left(C_k^U\right)^T z_k^U &= -U_k^T g_k \\ \left(C_k^Y\right)^T z_k^Y &= -\left(C_k^{UY}\right)^T z_k^U - Y_k^T g_k. \end{aligned}$$

If lingering iteration is performed then the next iterate x_{k+1} should lie in range(U_k). This can be enforced if the solution to the quadratic approximation problem

$$\min_d \left[d^T g_k + \frac{1}{2} d^T B_k d \right] \tag{12.48}$$

lies in the space range(U_k). By analogy to (12.34)–(12.35) (with Q_k replaced by Z_k and Z_k by U_k) if we introduce the reduced gradient and Hessian (with the respect to that space) $U_k^T g_k$ and $U_k^T B_k U_k$ respectively then the solution to (12.48) satisfying the additional requirement is given by

$$d_k^l = -U_k \left(U_k^T B_k U_k\right)^{-1} U_k^T g_k \tag{12.49}$$

which can also be expressed by

$$d_k^l = U_k \left(C_k^U\right)^{-1} z_k^U.$$

We use notation d_k^l to distinguish the lingering direction d_k from the RH direction which is denoted by d_k^{RH}.

Suppose now that f is a strictly convex function and we apply the BFGS method with an exact line search. Then, we can show that the difference $\phi(d_k) = f(x_k + d_k) - f(x_k)$ is given by

$$\begin{aligned} \phi(d_k^{RH}) &= -g_k^T \left(C_k^T C_k\right)^{-1} g_k \\ &\quad + \frac{1}{2} g_k^T \left(C_k^T C_k\right)^{-1} C_k^T C_k \left(C_k^T C_k\right)^{-1} g_k \\ &= -g_k^T C_k^{-1} C_k^{-T} g_k + \frac{1}{2} g_k^T C_k^{-1} C_k^{-T} g_k \\ &= -\frac{1}{2} g_k^T C_k^{-1} C_k^{-T} g_k \\ &= -\frac{1}{2} \|z_k\|^2 \end{aligned}$$

and since (12.49) holds

$$\begin{aligned}\phi(d_k^l) &= -g_k^T U_k \left(U_k^T B_k U_k\right)^{-1} U_k^T g_k \\ &\quad + \frac{1}{2} g_k^T U_k \left(U_k^T B_k U_k\right)^{-1} U_k^T B_k U_k \left(U_k^T B_k U_k\right)^{-1} U_k^T g_k \\ &= -\frac{1}{2} g_k^T U_k \left(U_k^T B_k U_k\right)^{-1} U_k^T g_k \\ &= -\frac{1}{2} g_k^T U_k \left(C_k^U\right)^{-1} \left(C_k^U\right)^{-T} U_k^T g_k \\ &= -\frac{1}{2} \|z_k^U\|^2.\end{aligned}$$

In the quadratic case g_k is orthogonal to the previously evaluated gradients which span the same space as the directions computed up to iteration k. Thus both $U_k^T g_k$ and z_k^U are zero and the lingering iterations are ruled out. However, when f is general nonlinear function the situation $\|z_k\| \approx \|z_k^U\|$ is possible and nearly all reduction of the quadratic model of f is obtained on $\mathscr{M}(\mathscr{U}_k)$. Gill and Leonard propose to step off of $\mathscr{M}(\mathscr{U}_k)$ if the condition $\|z_k^U\| \leqslant \tau \|z_k\|$ with $\tau \in (0.5, 1)$ is met. On the other hand, if $\|z_k^U\|$ is close to $\|z_k\|$ then lingering takes place.

The second important modification introduced into Algorithm 12.2 by Gill and Leonard concerns the initial approximation of the Hessian matrix B_1. The poor guess for σI propagates through the next iterations since consecutive BFGS updates are influenced by it. This is the major drawback of the standard quasi-Newton algorithm which does not allow to change the initial approximation as the algorithm progresses unless the approximation is built from the scratch at some later iteration. The limited memory BFGS method by Liua and Nocedal [125] overcomes this inefficiency of the standard BFGS method since at every iteration the Hessian BFGS approximation (or its inverse) is constructed from the iteration dependent $\sigma_k I$ and the last evaluated vectors s_k, $s_{k-1}, \ldots, s_{k-m+1}$ and y_k, $y_{k-1}, \ldots, y_{k-m+1}$. We should emphasize that this property of L-BFGS algorithm is also the feature of the reduced-Hessian approach. It occurs due to the reduced form of the Hessian approximation stated in (12.32). The diagonal part of $Q_k^T B_k Q_k$: σI_{n-h_k} can be changed at every iteration giving the better approximation of the curvature of f on $\mathscr{G}_k^{\perp}$ without destroying the accumulated so far the curvature information of f on $\mathscr{G}_k$.

Gill and Leonard consider the following choices for σ_{k+1}:

$$\sigma_{k+1}^{R0} = 1, \quad \sigma_{k+1}^{R1} = \frac{\|y_1\|^2}{s_1^T y_1},$$

where σ_{k+1}^{R1} is discussed by Shanno and Phua [193];

$$\sigma_{k+1}^{R2} = \min_{1 \leqslant i \leqslant k} \frac{s_i^T y_i}{\|s_i\|^2}$$

advocated by Siegel in his conjugate-direction scaling algorithm [197];

$$\sigma_{k+1}^{R3} = \frac{\|y_k\|^2}{s_k^T y_k}, \tag{12.50}$$

which is recommended in L-BFGS method (cf. Chap. 5). In [80] these reinitializing schemes are numerically compared and the formula (12.50) seems to be the best choice.

12.5 Limited Memory Reduced-Hessian Quasi-Newton Algorithms

The reduced-Hessian method is effective, even without applying lingering iterations, or reinitialization, when the dimension $h_k \ll n$. In that situation the amount of work needed per iteration is dependent on the dimension of the reduced space which is proportional to h_k. The BFGS update takes place in the reduced space so the computational effort related to the Cholesky factorization (in fact its update) and to the solution of linear equations is proportional to h_k^2 and thus is significantly lower than in standard quasi-Newton methods in which case the number of flops is of order n^2.

But, in general we can have the situation that $h_k \approx n$ and then the flops count of the reduced-Hessian method increases to $7n^2 + O(n)$ and compares unfavorably with that of standard quasi-Newton algorithms based on the Cholesky factorization – that count is estimated as $4n^2 + O(n)$ (see [80] and the count of NPSOL solver – [82]), or with the conjugate-direction scaling methods such as that of Lalee and Nocedal with the same flops count per iteration [80].

Therefore, it seems quite natural to develop a version of a reduced-Hessian method in which the dimension of space $\mathscr{G}_k$ is kept significantly lower than n by assuming that $\dim(\mathscr{G}_k) \leqslant m$ where m is some prespecified number such that $m \ll n$. Gill and Leonard in [81] (see also [195]) follow that idea and propose the limited memory reduced-Hessian method. The basic question they had to answer was how to choose the basis for the reduced space. The natural choice, which mimics that applied in the limited memory BFGS method where at iteration k quasi-Newton matrices are built with the help of (s_k, y_k), $(s_{k-1}, y_{k-1}), \ldots, (s_{k-m+1}, y_{k-m+1})$, is to work with the space spanned by the m most recently evaluated gradients g_k, $g_{k-1}, \ldots, g_{k-m+1}$. Although such an approach is feasible it has, according to Gill and Leonard, one major drawback – when the limited memory reduced-Hessian algorithm with that strategy is applied to strongly convex quadratic functions the finite iteration termination is not guaranteed.

Theorem 12.2 states that both directions $d_1, d_2, \ldots, d_m$ and gradients $g_1, g_2, \ldots, g_m$ can be used to build the current reduced space (notice that $p_1 \parallel g_1$ and we assume that all computed so far gradients have been accepted). However, when we move to iteration $m+1$ then in order to keep the dimension of the reduced space unchanged the new gradient g_{m+1} has to replace one of the gradients g_i, $i = 1, \ldots, m$. If we substitute g_1 and p_1 by g_{m+1} then the spaces spanned by $g_2, g_3, \ldots, g_{m+1}$ and by $d_2, d_3, \ldots, g_{m+1}$ can no longer be the same. Notice that $d_m \in \mathscr{G}_m$ can have components associated with g_1 and since $d_m \in \mathscr{D}_m$ it follows that $\text{range}([g_2 \ \cdots \ g_m \ g_{m+1}]) \neq \text{range}([d_2 \ \cdots \ d_m \ g_{m+1}])$.

The choice of the space used in the limited memory reduced-Hessian method influences the efficiency of the method. In [81] (see also [121]) the convergence in a finite number of iterations, in the case of strongly convex quadratic functions and with exact line searches, is established for the method based on the space generated by directions d_k (e.g. Sect. 1.10). Furthermore, it is claimed therein that the method exhibits inferior performance if the space defined by gradients g_k is used instead.

The scheme for updating the used space goes as follows. At the start of iteration k the reduced space used to compute d_k has the basis consisted of columns of the matrix

$$D_k^b = [d_l \; d_{l+1} \cdots d_{k-1} \; g_k],$$

where $l = k - m + 1$ (we assume that $k > m + 1$). Then the creation of D_{k+1}^b is done in three stages (for the simplicity of presentation we assume that g_{k+1} is accepted):

1. d_k is evaluated using D_k^b – the basis contains g_k in order to utilize the most recent information on function f.
2. g_k in D_k^b is replaced by d_k giving $\bar{D}_k^b$ – notice that $\bar{D}_k^b$ and D_k^b differ by a single column yet according to Theorem 12.2 they span the same subspace.
3. The first column is removed from $\bar{D}_k^b$ and the gradient g_{k+1} is added to $\bar{D}_k^b$ as its last column creating D_{k+1}^b.

In Stage 1 the following equations are solved

$$\begin{aligned} C_k^T z_k &= -Z_k^T g_k \\ C_k d_k &= z_k \,, \end{aligned}$$

with respect to z_k and d_k. C_k is the Cholesky factor of the reduced BFGS approximation to the Hessian matrix:

$$C_k^T C_k = Z_k^T B_k Z_k$$

and Z_k is such that

$$D_k^b = Z_k S_k$$

with $S_k \in \mathscr{R}^{m \times m}$ being an upper triangular matrix.

In Stage 2 the matrix Z_k does not have to be changed (according to Lemma 12.3) but the last column of S_k, which is equal to $Z_k^T g_k$, has to be replaced by $Z_k^T d_k$ forming the matrix $\bar{S}_k$. The form of the last column in S_k and $\bar{S}_k$ is the consequence of using the Gram–Schmidt orthogonalization method in the basis creation (cf. Appendix C). Suppose that g_k is added to the basis. Then

$$D_k^b = [\bar{D}_{k-1}^b \; g_k], \quad \bar{D}_{k-1}^b = Z_{k-1} \bar{S}_{k-1}$$

and

$$Z_k = [Z_{k-1} \; z_k], \quad S_k = \begin{bmatrix} \bar{S}_{k-1} & v_k \\ 0 & \rho_k \end{bmatrix},$$

where ρ_k is defined as in (12.39) stated for iteration k instead of $k+1$. Here, $v_k = Z_{k-1}^T g_k$ and if g_k is swapped for d_k then v_k is substituted by $Z_{k-1}^T d_k$ and ρ_k by $z_k^T d_k$

which we can show that is different from zero [121]. Thus, after the swap, instead of the matrix S_k we have the matrix $\bar{S}_k$

$$\bar{S}_k = \begin{bmatrix} \bar{S}_{k-1} & u_k \\ 0 & z_k^T d_k \end{bmatrix} \tag{12.51}$$

with $u_k = Z_{k-1}^T d_k$.

In Stage 3 the gradient g_{k+1} is added as the last column to $\bar{D}_k^b$ yielding

$$\check{D}_k^b = \left[\bar{D}_k^b \; g_{k+1}\right], \quad \check{Z}_k = [Z_k \; z_{k+1}] \tag{12.52}$$

and

$$\check{S}_k = \begin{bmatrix} \bar{S}_k & v_{k+1} \\ 0 & \rho_{k+1} \end{bmatrix}. \tag{12.53}$$

In the next step of Stage 3 one column from $\check{D}_k^b$ has to be removed – we assume that as the rule the column d_l leaves $\check{D}_k^b$. In that case we have

$$\check{D}_k^b = \left[d_l \; D_{k+1}^b\right]$$

and we have to find Z_{k+1} and S_{k+1} such that

$$D_{k+1}^b = Z_{k+1} S_{k+1}.$$

In order to transform $\check{S}_k^b$ to the matrix which as a submatrix has the upper triangular matrix $S_{k+1} \in \mathscr{R}^{m \times m}$ we use Givens rotations to zero the last m elements of row $(m+1)$ of the matrix $\check{S}_k^b$ giving

$$\check{S}_k^Q = \check{Q}_k \check{S}_k = \begin{bmatrix} \check{s}_k & S_{k+1} \\ \check{\tau}_k & 0 \end{bmatrix}. \tag{12.54}$$

Here, $\check{Q}_k$ represents the product of the Givens rotations and is an orthogonal matrix (cf. Appendix C).

Since $\check{Q}_k$ is orthogonal the following holds

$$\begin{aligned} \check{D}_k^b &= \check{Z}_k \check{S}_k = \check{Z}_k \check{Q}_k^T \check{Q}_k \check{S}_k \\ &= \check{Z}_k^Q \check{S}_k^Q \end{aligned} \tag{12.55}$$

with

$$\check{Z}_k^Q = \check{Z}_k \check{Q}_k^T. \tag{12.56}$$

Due to (12.54), if $\check{Z}_k^Q$ is represented as

$$\check{Z}_k^Q = [Z_{k+1} \; \check{z}_k] \tag{12.57}$$

then

$$\check{D}_k^b = \left[d_l \; D_{k+1}^b \right] = \check{Z}_k^Q \check{S}_k^Q = [Z_{k+1}\check{s}_k + \check{\tau}_k \check{z}_k \;\; Z_{k+1} S_{k+1}] \tag{12.58}$$

which implies that

$$D_{k+1}^b = Z_{k+1} S_{k+1}$$

as required.

In order to complete the description of the kth iteration of the limited memory reduced-Hessian method we have to show how to obtain the Cholesky factor C_{k+1}. In Stages 1 and 2 we do not change the matrix Z_k and, since the Hessian approximation is not altered either, we do nothing as far as C_k is concerned. In Stage 3 we add the vector g_{k+1} to the basis, its dimension increases so it has to be reflected in C_k. Using the same procedure as in the reduced-Hessian algorithm (cf. relations (12.41)–(12.45)) we evaluate $\check{C}_k$ such that

$$\check{Z}_k^T B_{k+1} \check{Z}_k = \check{C}_k^T \check{C}_k.$$

It remains to show what steps should be undertaken with respect to $\check{C}_k$ when d_l leaves the basis.

The orthogonal basis Z_{k+1} is related to $\check{Z}_k^Q$ by $Z_{k+1} = \check{Z}_k^Q E_m$ where E_m is composed from the first m columns of the identity matrix. We have

$$\begin{aligned} Z_{k+1}^T B_{k+1} Z_{k+1} &= E_m^T \left(\check{Z}_k^Q \right)^T B_{k+1} \check{Z}_k^Q E_m \\ &= E_m^T \check{Q}_k \check{Z}_k^T B_{k+1} \check{Z}_k \check{Q}_k^T E_m \\ &= E_m^T \check{Q}_k \check{C}_k^T \check{C}_k \check{Q}_k^T E_m. \end{aligned}$$

Since $\check{C}_k \check{Q}_k^T$ is not upper triangular, in general, we transform it (cf. [81]) to this form by using the orthogonal matrix $\tilde{Q}_k$ (which is the product of Givens matrices). As the result we get $\tilde{C}_k$ such that

$$\begin{aligned} \tilde{C}_k &= \tilde{Q}_k \check{C}_k \check{Q}_k^T \\ Z_{k+1}^T B_{k+1} Z_{k+1} &= E_m^T \check{Q}_k \check{C}_k^T \tilde{Q}_k^T \tilde{Q}_k \check{C}_k \check{Q}_k^T E_m \\ &= E_m^T \tilde{C}_k^T \tilde{C}_k E_m \end{aligned} \tag{12.59}$$

and C_{k+1} is the leading $m \times m$ block of $\tilde{C}_k E_m$.

The limited memory reduced-Hessian algorithm can also use reinitialization by adjusting at every iteration σ_k and lingering iterations in order to utilize in the best possible way the curvature information. The limited memory version of Algorithm 12.2 could be stated by incorporating into the description of Algorithm 12.2 the linear algebra operations outlined in the three stages described above. In the next section we present the preconditioned conjugate gradient algorithm which applies the reduced-Hessian approach in building scaling matrices, the limited memory reduced-Hessian quasi-Newton algorithm is a particular version of this algorithm.

12.6 Preconditioned Conjugate Gradient Based Reduced-Hessian Algorithm

The approach based on the limited memory reduced-Hessian can be applied in a preconditioned conjugate gradient algorithm. If we want to use the shortest residuals algorithm as a conjugate gradient algorithm in which we apply preconditioning we have to define scaling matrix D_k such that $\hat{x}_k = D_k x_k$ at every iteration. The matrix should be nonsingular and $D_k^T D_k$ should be equal to the quasi-Newton approximation of the Hessian matrix: $B_k = D_k^T D_k$. We define D_k noting that the reduced-Hessian quasi-Newton method also is based on the transformation of variables by orthogonal matrices Q_k which are factors in the QR decompositions of the basis matrices, for example, when the basis T_k is formed by matrix composed of gradients, we have

$$T_k = Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix}.$$

In the space of x^{Q_k} variables defined by the equation $x = Q_k x^{Q_k}$ the approximation to the Hessian is represented by $Q_k^T B_k Q_k$ and has the form (12.32) and the gradient is represented by $Q_k^T g_k$ as given by (12.33). The next transformation $\hat{x} = \bar{C}_k x^{Q_k}$, with $\bar{C}_k$ being the Cholesky factor of $Q_k^T B_k Q_k$, brings the Hessian approximation in the space of $\hat{x}$ variables to the identity matrix. Eventually, the transformation from the space of x variables to the space of $\hat{x}$ variables is expressed by

$$\hat{x} = \bar{C}_k Q_k^T x$$

and the matrix D_k used in the preconditioned shortest residuals method is stated as

$$D_k = \bar{C}_k Q_k^T. \tag{12.60}$$

The kth iteration of the preconditioned shortest residuals algorithm is as follows

$$\hat{d}_k = -\lambda_k \hat{g}_k + (1-\lambda_k)\hat{\beta}_k \hat{d}_{k-1}^c \tag{12.61}$$

(here $\hat{d}_{k-1}^c$ is vector transformed by the transformation (12.60)) while in space of x variables is expressed by

$$\begin{aligned} d_k &= -\lambda_k D_k^{-1} D_k^{-T} g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1} \\ &= -\lambda_k Q_k \bar{C}_k^{-1} \bar{C}_k^{-T} Q_k^T g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1} \\ &= -\lambda_k Q_k \left(Q_k^T B_k Q_k\right)^{-1} Q_k^T g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1} \\ &= -\lambda_k Q_k Q_k^T B_k^{-1} Q_k Q_k^T g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1} \\ &= -\lambda_k B_k^{-1} g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1} \end{aligned}$$

(since $Q_k Q_k^T = I$) which shows that B_k is the preconditioning matrix.

In order to get a viable method we have to show that equations analogous to (8.5)–(8.6) are easy to solve. We have

$$Q_k \bar{C}_k \hat{g}_k = g_k \tag{12.62}$$

$$\bar{C}_k Q_k^T d_k = \hat{d}_k \tag{12.63}$$

and (12.62), due to Lemma 12.2, reduce to the equations

$$\bar{C}_k^T \hat{g}_k = \begin{bmatrix} Z_k^T g_k \\ 0 \end{bmatrix}, \tag{12.64}$$

where Z_k is such that $Q_k = [Z_k \; W_k]$ and $Z_k^T W_k = 0$. On the other hand, due to Lemma 12.3,

$$\begin{aligned} D_k d_{k-1} &= \bar{C}_k Q_k^T d_{k-1} \\ &= \bar{C}_k \begin{bmatrix} Z_k^T d_{k-1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} C_k Z_k^T d_{k-1} \\ 0 \end{bmatrix} \end{aligned} \tag{12.65}$$

and we can show that (12.63) can be restated as

$$d_k = Z_k C_k^{-1} \hat{d}_k^Z , \tag{12.66}$$

where $\hat{d}_k^Z$ is the part of $\hat{d}_k$ corresponding to Z_k (the other part is equal to zero).

The above analysis shows that (12.61) can be substituted by the equation

$$\hat{d}_k^Z = -\lambda_k C_k^{-1} C_k^{-T} \hat{g}_k^Z + (1 - \lambda_k) \hat{\beta}_k \hat{d}_{k-1}^Z$$

since the parts of all vectors appearing in (12.61) corresponding to the subspace spanned by columns of matrix W are zero. Furthermore, the direction d_k defined in (12.66) is the direction of descent which follows from relation (8.7) which is valid for any scaling nonsingular matrix D_k, in particular for D_k in (12.60).

It remains to specify the coefficient $\hat{\beta}_k$ in (12.61). We can assume that $\hat{\beta}_k = 1$ and then we have the Fletcher–Reeves version of the algorithm, or we evaluate $\hat{\beta}_k$ according to the rule

$$\hat{\beta}_k = \frac{\|\hat{g}_k\|^2}{\left|\hat{g}_k^T \left(\hat{g}_k - \hat{g}_{k-1}^c\right)\right|} , \tag{12.67}$$

where $\hat{g}_{k-1}^c$ is defined by

$$\bar{C}_k^T \hat{g}_{k-1}^c = \begin{bmatrix} Z_k^T g_{k-1} \\ 0 \end{bmatrix}. \tag{12.68}$$

to have the Polak–Ribière version. The main effort in solving (12.68) is associated with the calculation of $Z_k^T g_{k-1}$. Notice, however, that from the previous iteration we have at our disposal $Z_{k-1}^T g_{k-1}$ and that matrix Z_k differs from Z_{k-1} by last column so we need to evaluate only $z_k^T g_{k-1}$ at the cost of n multiplications and additions.

The evaluation of vector $\hat{d}_{k-1}^c$ in (12.61) requires the same computational effort – it is obtained, according to (12.65), by the formula

$$\hat{d}_{k-1}^c = \begin{bmatrix} C_k Z_k^T d_{k-1} \\ 0 \end{bmatrix},$$

which implies that $\hat{d}_{k-1}^Z = C_k Z_k^T d_{k-1}$. Since $Z_{k-1}^T d_{k-1}$ is calculated in iteration $k-1$, the main extra effort while evaluating $\hat{d}_{k-1}^c$ is associated with $z_k^T d_{k-1}$.

If the limited memory reduced-Hessian method is considered then computations required for getting $\hat{d}_{k-1}^c$ and $\hat{g}_{k-1}^c$ differ from that described above since Z_{k+1} is not simple enlargement of Z_k by adding a new column z_{k+1} and can be described by the formula

$$Z_{k+1} = Z_k \breve{Q}_k^T$$

(cf. (12.52) and (12.55)–(12.57)). $\breve{Q}_k$ is the product of m Givens rotations so the cost of its applying to a given vector is proportional to m. Thus, evaluating

$$\hat{d}_{k-1}^c = \bar{C}_k Z_k d_{k-1} = \bar{C}_k Z_{k-1} \breve{Q}_{k-1}^T d_{k-1} \tag{12.69}$$

bears cost proportional to mn, while $\hat{g}_{k-1}^c$ satisfying the equation

$$\bar{C}_k^T \hat{g}_{k-1}^c = \begin{bmatrix} \breve{Q}_k Z_{k-1}^T g_{k-1} \\ 0 \end{bmatrix} \tag{12.70}$$

can be determined at a cost proportional to m^2 since $Z_{k-1}^T g_{k-1}$ is known from iteration $k-1$.

We have all necessary ingredients to state the preconditioned conjugate gradient based reduced-Hessian algorithm.

Algorithm 12.3. (The preconditioned conjugate gradient based reduced-Hessian algorithm)

Parameters: $\mu,\ \eta \in (0,1)$, $\mu < \eta$, $\sigma_1 > 0$, positive integer number m.

1. Choose an arbitrary $x_1 \in \mathscr{R}^n$, set $h_1 = 1$, $D_1^b = [g_1]$, $Z_1 = [g_1/\|g_1\|]$, $S_1 = [\|g_1\|]$, $C_1 = [\sqrt{\sigma_1}]$ and $k = 1$.
2. If $k = 1$ substitute $-g_1$ for d_1, otherwise calculate d_k from $\hat{d}_k$ that satisfies (12.61) where $\hat{g}_k$ is obtained according to formula (12.64), $\hat{d}_{k-1}^c$ due to (12.69).
   If g_k is accepted then swap the last column of D_k^b with d_k getting $\bar{D}_k^b$ and $\bar{S}_k$ according to (12.51). Otherwise set $\bar{D}_k^b = D_k^b$, $\bar{S}_k = S_k$. Substitute C_k for $\bar{C}_k$.

3. Find α_k according to the modified Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.
4. If $\|g_{k+1}\| = 0$ then STOP.
   Otherwise do orthogonalization on the vectors from D_k^b and the gradient g_{k+1}.
   If g_{k+1} is accepted form $\breve{D}_k^b$ from $\bar{D}_k^b$ and $\breve{Z}_k$ from Z_k using (12.52) and $\breve{S}_k$ from $\bar{S}_k$ on the basis of (12.53). Set $\breve{h}_k = h_k + 1$.
   If g_{k+1} is rejected substitute $\bar{D}_k^b$, $\bar{Z}_k$, $\bar{S}_k$ for $\breve{D}_k^b$, $\breve{Z}_k$, $\breve{S}_k$. Set $\breve{h}_k = h_k$.
   Update $\bar{C}_k$ to $\breve{C}_k$ by using (12.42)–(12.45).
   Compute σ_{k+1} and do reinitialization by replacing σ_k with σ_{k+1}.
   If $\breve{h}_k = m+1$ then remove the first column of matrix $\breve{D}_k^b$ and create D_{k+1}^b from $\breve{D}_k^b$ by (12.58), S_{k+1} from $\breve{S}_k$ by (12.54), Z_{k+1} from $\breve{Z}_k$ by (12.56)–(12.57) and C_{k+1} from $\breve{C}_k$ by (12.59). Decrease $\breve{h}_k$ by one.
   If $\breve{h}_k < m+1$ substitute $\breve{D}_k$, $\breve{S}_k$, $\breve{Z}_k$ and $\breve{C}_k$ for D_{k+1}^b, S_{k+1}, Z_{k+1} and C_{k+1} respectively.
   Calculate $\hat{\beta}_{k+1}$ according to (12.67) using $\hat{g}_k^c$ defined by (12.70) (with $k-1$ replaced by k). Substitute $\breve{h}_k$ for h_{k+1}.
   Increase k by one and go to Step 2.

Algorithm 12.3 is the generalization of the L-RHR (Limited memory-Reduced-Hessian with Reinitialization) algorithm stated in [81]. Its formula for direction d_k contains also the previous direction d_{k-1} in the current reduced space. Potentially it should improve the performance of the L-RHR algorithm by taking simultaneously quasi-Newton and conjugate gradient iterations, especially when m is small. But this happens at the extra cost estimated at $mn + O(m^2)$. Therefore, an iteration of Algorithm 12.3 requires $3mn + 2n + O(m^2)$ operations if the matrix Z_k is not stored implicitly but D_k^b and products involving Z_k are computed as needed using D_k^b and S_k (cf. [81] and [195] for details). Although the cost of one iteration of Algorithm 12.3 is higher than that of L-HRH algorithm, it is still lower in comparison to the limited memory quasi-Newton method of Nocedal [144] which executes its iteration with the total number of operations equal approximately to $4mn$.

As far as the memory requirements are concerned one extra n dimensional vector has to be stored when compared to the L-RHR algorithm so if only matrices of dimension $m \times n$ are concerned it is approximately half the storage required for L-BFGS-B code.

In [81] the L-RHR algorithm is numerically compared to several limited memory algorithms including the implementation of the Siegel algorithm presented in [195] and L-BFGS-B method. The comparison shows the superiority of L-RHR in terms of CPU time and the number of function evaluations while requiring half the storage. Algorithm 12.3 is the extension of the L-HRH algorithm by giving more flexibility with the inclusion in the formula for d_k the scaled previous direction. Due to Theorem 4.5 one cannot expect the increase of efficiency of the Newton method by combining it with a conjugate gradient algorithm (see Algorithm 4.3) – in that case the modified quasi-Newton does not exhibit the quadratic rate of convergence. However, the limited memory quasi-Newton method discussed in this section has

likely linear rate of convergence (e.g. Theorem 5.2) thus its modification based on the conjugate gradient scheme can be beneficial.

12.7 Notes

The chapter presents a new scaling method which in contrast to that outlined in Chap. 3, and based on the work by Oren, Spedicato and Luenberger [148, 150, 152] gives marked improvement for quasi-Newton methods since it is not restricted to initial approximations of Hessian matrices.

The limited memory reduced-Hessian quasi-Newton algorithm which can be regarded as the most efficient version of a column-scaling algorithm, when applied to large scale problems, is the most efficient algorithm among limited memory methods. It could be extended to problems with box constraints along the lines the limited memory BFGS has been modified in Chap. 11 in order to cope with simple constraints. That remark applies also to Algorithm 12.3 presented in this chapter.

Appendix A
Elements of Topology and Analysis

A.1 The Topology of the Euclidean Space

We consider a linear space $\mathscr{X}$ with the mapping $\|\cdot\|$ which for every element of $\mathscr{X}$ assigns a nonnegative real number and satisfies:

1. $\|x\| > 0$ for $x \neq 0$ and $\|0\| = 0$.
2. $\|x+y\| \leqslant \|x\| + \|y\|$ for any $x, y \in \mathscr{X}$.
3. $\|\alpha x\| = |\alpha|\|x\|$ for any real number α and $x \in \mathscr{X}$.

We call the mapping $\|\cdot\|$ the *norm* of the space $\mathscr{X}$ and the space $\mathscr{X}$ the *normed space*.

If space $\mathscr{X}$ is the set of n dimensional vectors

$$x = \begin{bmatrix} (x)_1 \\ (x)_2 \\ \vdots \\ (x)_n \end{bmatrix}$$

with real components $(x)_i$, $i = 1, \ldots, n$ and the norm is defined by

$$\|x\| = \sqrt{\sum_{i=1}^{n} (x)_i^2}$$

then we call the space $\mathscr{X}$ the *Euclidean space* and denote it by $\mathscr{R}^n$. The norm $\|\cdot\|$ is then called the Euclidean and denoted by $\|\cdot\|_2$. One of the most interesting properties of the Euclidean norm is the Hölder inequality which states that

$$|x^T y| \leqslant \|x\|_2 \|y\|_2$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

and the equality holds if one of these vectors is the multiple of the other. Here, x, $y \in \mathscr{R}^n$ and

$$x^T y = \langle x, y \rangle = \sum_{i=1}^{n} (x)_i (y)_i$$

is the *scalar product*.

Suppose that $\{x_k\}$ is a sequence of points belonging to $\mathscr{R}^n$. We say that $\{x_k\}$ converges to $x \in \mathscr{R}^n$ and write $\lim_{k\to\infty} x_k = x$ if for any $\varepsilon > 0$ there exists an index K such that

$$\|x_k - x\| \leqslant \varepsilon \tag{A.1}$$

for any $k \geqslant K$. We assume that the norm in (A.1) is the Euclidean norm but for the notation simplicity we write $\|\cdot\|$ instead of $\|\cdot\|_2$.

Not every sequence is convergent, however when $\{x_k\}$ is convergent to the limit x then every its subsequence $\{x_{k_l}\}$ is convergent and has the same limit x. Here, $\{k_l\}$ is a sequence of natural numbers such that $k_{l+1} > k_l$. We say that $\{x_k\}$ has an accumulation point (or a limit point) x if there exists a subsequence $\{x_{k_l}\}$ such that $\lim_{l\to\infty} x_{k_l} = x$.

Consider now a sequence of real numbers $\{\alpha_k\}$ and all its accumulation points assuming that $+\infty$ and $-\infty$ are also limit points. Then we can define $\limsup \alpha_k$ as the largest limit point of $\{\alpha_k\}$ and $\liminf \alpha_k$ as the lowest limit point. According to the definition there exist α^i, α^s (possibly equal to $+\infty$, or $-\infty$) and subsequences of $\{\alpha_k\}$: $\{\alpha_{k_l}\}$ and $\{\alpha_{k_m}\}$ such that

$$\lim_{l\to\infty} \alpha_{k_l} = \alpha^i,$$
$$\lim_{m\to\infty} \alpha_{k_m} = \alpha^s.$$

Furthermore, for any other accumulation point α we do not have $\alpha > \alpha^s$, or $\alpha < \alpha^i$.

The set $A \subset \mathscr{R}^n$ is *bounded* if there exists a bounded number M such that

$$\|x\| \leqslant M$$

for all $x \in A$. The set $A \subset \mathscr{R}^n$ is *closed* if for any sequence $\{x_k\}$, such that $x_k \in A$ for any k, all accumulation points of $\{x_k\}$ belongs to A. If $A \subset \mathscr{R}^n$ is bounded and closed then it is *compact*. One of the most important properties of the Euclidean space is that if $\{x_k\}$ is the sequence of points belonging to the compact set A then there exists at least one limit point of the sequence belonging to A.

A subset $A \subset \mathscr{R}^n$ is *open* if for every $x \in A$ there exists $\varepsilon > 0$ such that

$$\{y \in \mathscr{R}^n : \ \|x - y\| \leqslant \varepsilon\} \subset A.$$

The concepts of open, closed and compact sets can be defined in more general spaces than normed spaces. Suppose that S is a set. Then a *topology*, or topological

structure on S is a collection of subsets of S, called open sets, satisfying:

1. The union of any number of open sets is open.
2. The intersection of any finite number of open sets is open.
3. The set S and the empty set $\emptyset$ are open.

A set S with a topology is called a *topological space*.

A subset A of a topological space is said to be closed if its complement $A^C = S \backslash A$ is open. One can show that the intersection of any number of closed sets is closed and the union of any finite number of closed sets is closed. In particular the whole space S and the empty set $\emptyset$ are closed. A subset A of a topological space is compact if from any union of open sets which cover A (A is a subset of the union) we can choose a finite union of sets covering A.

A *neighborhood* $\mathscr{N}$ of a point p of a topological space is any open set which contains p. In particular, if we consider space $\mathscr{X}$ with the norm then the open ball around p with radius $\varepsilon > 0$ is a neighborhood defined by

$$\mathscr{B}(p,\varepsilon) = \{y \in \mathscr{X} : \|y - p\| < \varepsilon\}.$$

One can build a topology in space $\mathscr{X}$ by taking all open balls of all points in $\mathscr{X}$.

If A is a subset of a topological space S then there exists a unique open set, denoted $\mathrm{int}(A)$ and called *interior* of A, which is subset of A and which contains any other open subset of A. In fact $\mathrm{int}(A)$ is the union of all open sets contained in A. If A is an open set then $\mathrm{int}(A) = A$. Likewise, there is a unique closed set, denoted $\mathrm{cl}(A)$ and called the *closure* of A, which contains A and which is a subset of any other closed set containing A. Actually, $\mathrm{cl}(A)$ is the intersection of all closed sets which contain A. If A is a closed set then $\mathrm{cl}(A) = A$.

We say that $A \subset S$ is *dense* in $B \subset S$, if $B \subset \mathrm{cl}(A)$ which means that every point of B is the limit point of some sequence of points belonging to A (this is equivalent to the condition that every neighborhood of every point $x \in B$ contains points of A). In particular, A is dense in S if $S = \mathrm{cl}(A)$.

Finally, we define the affine hull and relative interior of a set $A \subset \mathscr{R}^n$. To this end we introduce the *linear combination* of vectors $x_1, x_2, \ldots, x_m$ in A as a vector x defined by

$$x = \sum_{i=1}^{m} \alpha_i x_i \,,$$

where α_i, $i = 1, \ldots, m$ are real numbers. The *affine hull* of A, denoted as $\mathrm{aff}(A)$, is determined as

$$\mathrm{aff}(A) = \{x \in \mathscr{R}^n : \ x \text{ is a linear combination of vectors in } A\}.$$

The *relative interior* of A, denoted as $\mathrm{ri}(A)$, is defined then as its interior relative to $\mathrm{aff}(A)$. Thus, $x \in \mathrm{aff}(A)$ if there is an $\varepsilon > 0$ such that

$$(x + \varepsilon \mathscr{B}(0,1)) \bigcap \mathrm{aff}(A) \subset A. \tag{A.2}$$

As an example take

$$A = \left\{x \in \mathscr{R}^3 : \ (x)_1 \in [-1,1],\ (x)_2 \in (-1,1),\ (x)_3 = 0\right\}.$$

Then,

$$\begin{aligned} \text{aff}(A) &= \mathscr{R} \times \mathscr{R} \times \{0\}, \\ \text{ri}(A) &= \left\{x \in \mathscr{R}^3 : \ (x)_1 \in (-1,1),\ (x)_2 \in (-1,1),\ (x)_3 = 0\right\}. \end{aligned}$$

A.2 Continuity and Convexity

Suppose that f is a function which to a point $x \in \mathscr{R}^n$ assigns a real number: $f : \mathscr{R}^n \to \mathscr{R}$. By $\text{dom}(f)$ we denote all these points in $\mathscr{R}^n$ for which f is defined and different from $+\infty$:

$$\text{dom}(f) = \{x \in \mathscr{R}^n : \ f \text{ is defined at } x \text{ and } f(x) < +\infty\}.$$

We say that f has a *limit* g at the point $\bar{x}$ and write

$$\lim_{x \to \bar{x}} f(x) = g$$

if for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$0 < \|x - \bar{x}\| < \delta \quad \text{and} \quad x \in \text{dom}(f) \ \Rightarrow \ |f(x) - g| < \varepsilon.$$

Then the function f is *continuous* at $\bar{x} \in \text{dom}(f)$ if the limit of f at $\bar{x}$ exists and is equal to $f(\bar{x})$:

$$\lim_{x \to \bar{x}} f(x) = f(\bar{x}).$$

If f is defined on $\mathscr{R}$ ($n = 1$) then we can work with one-sided limits. We say that f has the *right-hand side limit* at $\bar{x}$, equal to g, and we write

$$\lim_{x \downarrow \bar{x}} f(x) = g,$$

if for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$0 < x - \bar{x} < \delta \quad \text{and} \quad x \in \text{dom}(f) \ \Rightarrow \ |f(x) - g| < \varepsilon.$$

Likewise we say that f has the *left-hand side limit* at $\bar{x}$, equal to g, and we write

$$\lim_{x \uparrow \bar{x}} f(x) = g,$$

if for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$0 < \bar{x} - x < \delta \quad \text{and} \quad x \in \text{dom}(f) \;\Rightarrow\; |f(x) - g| < \varepsilon.$$

Another useful concept is that of lower (upper) semicontinuity. A function f is *lower semicontinuous* at the point $\bar{x} \in \text{dom}(f)$ if

$$\liminf_{x \to \bar{x}} f(x) \geqslant f(\bar{x}),$$

and f is *upper semicontinuous* at the point $\bar{x} \in \text{dom}(f)$ if

$$\limsup_{x \to \bar{x}} f(x) \leqslant f(\bar{x}).$$

Convexity is very useful concept in optimization. A set $A \in \mathscr{R}^n$ is called *convex* if for any $x, y \in A$ and any $\lambda \in [0,1]$

$$\lambda x + (1-\lambda) y \in A.$$

In fact, if A is convex then any *convex combination* of points $x_i \in A$, $i = 1, \ldots, m - x$ defined as

$$x = \sum_{i=1}^{m} \lambda_i x_i, \quad \lambda_i \geqslant 0, \quad i = 1, \ldots, m, \quad \sum_{i=1}^{m} \lambda_i = 1$$

belongs to A: $x \in A$.

The *convex hull* of a set $A \in \mathscr{R}^n$, denoted $\text{co}(A)$, is the set of all convex combinations of points in A. $\text{co}(A)$ is the smallest convex set containing A. If A is convex then $\text{co}(A) = A$. Important property of a convex hull is described in the following lemma (called the Caratheodory's theorem, cf. e.g. [187]).

Theorem A.1. *If $A \in \mathscr{R}^n$ then $x \in \text{co}(A)$ if and only if x is expressible as a convex combination of $n+1$ (not necessarily different) points of A.*

Suppose that $\text{dom}(f) = \mathscr{R}^n$ then f is called a *convex function* if for any $\lambda \in [0,1]$

$$f(\lambda x + (1-\lambda) y) \leqslant \lambda f(x) + (1-\lambda) f(y) \tag{A.3}$$

for any $x, y \in \mathscr{R}^n$. We can define a convex function when $\text{dom}(f) \neq \mathscr{R}^n$ but then we require that $\text{dom}(f)$ is a convex subset of $\mathscr{R}^n$ and the condition (A.3) has to be fulfilled for any $x, y \in \text{dom}(f)$.

The concept of the convex function is closely related to convexity of a set. We define the *epigraph* of f as

$$\text{epi}(f) = \left\{ (x, y) \in \mathscr{R}^{n+1} : \; y \geqslant f(x) \right\},$$

and we can show that f is convex if and only if $\text{epi}(f)$ is a convex set.

We say that function f is *concave* if the function $(-f)(x) = -f(x)$ is convex. Suppose that $\alpha_i \geqslant 0$, $i = 1,\ldots,m$ and functions f_i, $i = 1,\ldots,m$ are convex, then the functions

$$\hat{f}(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$$
$$\tilde{f}(x) = \max_{i=1,\ldots,m} f_i(x)$$

are convex.

A function $f : \mathscr{R}^n \to \mathscr{R}$ is *strictly convex* if

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$

for any $\lambda \in (0,1)$ and $x \neq y \in \mathscr{R}^n$.

A function $f : \mathscr{R}^n \to \mathscr{R}$ is *locally uniformly convex* if there exists an increasing function $d : \mathscr{R}_+ \to \mathscr{R}_+$ (here, $\mathscr{R}_+ = \{x \in \mathscr{R} : \ x \geqslant 0\}$) such that

$$d(0) = 0, \qquad d(t) > 0 \quad \text{if } t > 0$$
$$f(\lambda x + (1-\lambda)y) \leqslant \lambda f(x) + (1-\lambda) f(y) - \lambda(1-\lambda) d(\|x-y\|)$$

for all $\lambda \in [0,1]$ and $x, y \in \mathrm{dom}(f)$.

In nondifferentiable optimization we deal with a closed, proper convex function f on $\mathscr{R}^n$, i.e. such that f is convex, lower semicontinuous and never assumes the value $-\infty$, although $+\infty$ is allowed. Under certain compactness conditions such functions are continuous (cf. e.g. [187]).

Theorem A.2. *Suppose that f is closed, proper convex function and for any $\alpha \in \mathscr{R}$ the set*

$$\{x \in \mathrm{dom}(f) : \ f(x) \leqslant \alpha\}$$

is compact. Then f is continuous on $\mathrm{dom}(f)$.

The stronger property than continuity is local Lipschitz continuity. A function $f : \mathscr{R}^n \to \mathscr{R}$ is said to be *locally Lipschitzian* if for each bounded subset A of $\mathscr{R}^n$ there exists a Lipschitz constant $L = L(A) < \infty$ such that

$$|f(x) - f(y)| \leqslant L\|x-y\| \qquad \text{for all} \quad x, y \in A. \tag{A.4}$$

Then, in particular f is continuous. Other examples of locally Lipschitzian functions are: continuously differentiable functions, convex functions, concave functions, any linear combination or pointwise maximum of a finite collection and superpositions of such functions.

A.3 Derivatives

Suppose that f is a locally Lipschitzian function, i.e. such that (A.4) is satisfied for any bounded $A \subset \mathscr{R}^n$. If $x \in \text{int}(A)$ then the *Clarke generalized directional derivative* of f at x in a direction $d \in \mathscr{R}^n$ defined by

$$f^{\circ}(x;d) = \limsup_{y \to x, t \downarrow 0} \frac{f(y+td)-f(y)}{t}$$

is a finite, convex function of d such that $f^{\circ}(x;d) \leqslant L\|d\|$. Therefore, the Dini upper directional derivative of f at x in a direction d:

$$f^{D}(x;d) = \limsup_{t \downarrow 0} \frac{f(x+td)-f(x)}{t}$$

exists for each $d \in \mathscr{R}^n$.

The directional derivative of a locally Lipschitzian function f at $x \in \text{int}(A)$ in a direction d

$$f'_{+}(x;d) = \lim_{t \downarrow 0} \frac{f(x+td)-f(x)}{t}$$

does not always exist. However, if it exists then we have

$$f'_{+}(x;d) \leqslant f^{D}(x;d).$$

On the other hand we always have

$$f^{D}(x;d) \leqslant f^{\circ}(x;d).$$

If for a function f at x and in a direction d there exists

$$f'_{-}(x;d) = \lim_{t \uparrow 0} \frac{f(x+td)-f(x)}{t}$$

and is equal to $f'_{+}(x;d)$ then f has *Gateaux derivative* at x in a direction d which we denote by $f'(x;d)$. If $f'(x;d)$ is a linear function in d then we say that f is Gateaux differentiable at x in a direction d. Then, the following holds

$$f'(x;d) = \langle g(x), d \rangle$$

for any $d \in \mathscr{R}^n$. Here, $g(x)$ is not dependent on d but is determined by x. $g(x)$ is called the *gradient* of f at x denoted by

$$g(x) = \nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} (g(x))_1 \\ (g(x))_2 \\ \vdots \\ (g(x))_n \end{bmatrix}$$

and the ith component of $\nabla f(x)$, $\frac{\partial f(x)}{\partial x_i}$, is the partial derivative of f at x.

Usually we work with a gradient ∇f which is related to the *Frechet derivative* – in that case $\nabla f(x)$ is such that

$$\lim_{p\to d, t\downarrow 0} \frac{f(x+tp)-f(x)}{t} = \langle \nabla f(x), d\rangle \quad \text{for all } d \in \mathscr{R}^n. \tag{A.5}$$

The relation (A.5) can be expressed equivalently as

$$f(x+d) = f(x) + \langle \nabla f(x), d\rangle + o(\|d\|) \quad \text{for all } d \in \mathscr{R}^n$$

with $o(t)/t \to 0$ as $t \downarrow 0$.

If in the limit in (A.5) p is equal to d but x is replaced by y such that $y \to x$, namely

$$\lim_{y\to x, t\downarrow 0} \frac{f(y+td)-f(x)}{t} = \langle \nabla f(x), d\rangle \quad \text{for all } d \in \mathscr{R}^n,$$

then f is strictly differentiable at x and we can show that ∇f is continuous at x relative to its domain

$$\mathrm{dom}(\nabla f) = \{y \in \mathscr{R}^n : \ f \text{ is differentiable at } y\}.$$

The second derivative of f is defined by applying the above reasoning to every component of the function ∇f, namely we say that f is twice differentiable at x if there exists the matrix $\nabla^2 f(x) \in \mathscr{R}^{n\times n}$ such that

$$\nabla f(x+d) = \nabla f(x) + \nabla^2 f(x)d + o(\|d\|) \quad \text{for all } d \in \mathscr{R}^n,$$

where $o(t)/t \to 0$ as $t \downarrow 0$. In particular, we write

$$\frac{\partial f(x+d)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{n} \frac{\partial^2 f(x)}{\partial x_j \partial x_i}(d)_j + o(\|d\|) \quad \text{for all } d \in \mathscr{R}^n.$$

If f is twice continuously differentiable at x then $\nabla^2 f(x)$ is a symmetric matrix since we can show that

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i} \tag{A.6}$$

for any $i, j = 1, \ldots, n$ and functions in (A.6) constitute the Hessian matrix

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_i^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

Suppose that $\mathscr{N}$ is an open set in $\mathscr{R}^n$. Then we say that f is continuously differentiable on $\mathscr{N}$, and write $f \in \mathscr{C}^1(\mathscr{N})$, if ∇f is defined and is continuous on $\mathscr{N}$. Furthermore, $f \in \mathscr{C}^2(\mathscr{N})$ if $\nabla^2 f$ exists and is continuous on $\mathscr{N}$.

In optimization very often we refer to the function approximation based on its first and second order derivatives. These approximations can be regarded as the results following the mean value theorems stated under the assumption that $f \in \mathscr{C}^1(\mathscr{N})$, or $f \in \mathscr{C}^2(\mathscr{N})$ for some open set $\mathscr{N}$. Below we present some versions of the mean value theorem.

Theorem A.3. *Suppose that $f \in \mathscr{C}^1(\mathscr{N})$ and $x.y \in \mathscr{N}$ then*

$$f(y) = f(x) + \int_0^1 \nabla f(x+\alpha(y-x))^T (y-x) d\alpha$$

and

$$f(y) = f(x) + \nabla f(x+\alpha(y-x))^T (y-x)$$

for some $\alpha \in (0,1)$. Furthermore, if $f \in \mathscr{C}^2(\mathscr{N})$ then

$$\begin{aligned} f(y) = {} & f(x) + \nabla f(x)^T (y-x) \\ & + \int_0^1 \left(\int_0^\xi (y-x)^T \nabla^2 f(x+\alpha(y-x))(y-x) d\alpha \right) d\xi \end{aligned}$$

and

$$\begin{aligned} f(y) = {} & f(x) + \nabla f(x)^T (y-x) \\ & + \frac{1}{2}(y-x)^T \nabla^2 f(x+\alpha(y-x))(y-x) \end{aligned}$$

for some $\alpha \in (0,1)$.

The mean value theorem is widely used in optimization to build linear or quadratic approximations of functions defining the problem. These approximations enable us to handle efficiently the direction finding subproblems.

When an optimization problem is not described by functions from the class $\mathscr{C}^1$ (or $\mathscr{C}^2$) we cannot refer to gradients and Hessians to estimate perturbations around a current point. However, if a function f is locally Lipschtzian we are still equipped with analytical tools which give us possibility to work with linear, or quadratic approximations instead of less tractable general nonlinear functions. Suppose that a set $A \subset \mathscr{R}^n$ is as referred in the condition (A.4). Then

$$\langle \nabla f(y), d \rangle = f'(y;d) = \lim_{t \to 0} \frac{f(y+td) - f(y)}{t} \leqslant L\|d\| \tag{A.7}$$

at a point y at which f is differentiable. One can show that the set of points at which (A.7) holds is dense in A. Therefore, for all $y \in A \bigcap \mathrm{dom}(\nabla f)$ we have

$$\|\nabla f(y)\| \leqslant L \tag{A.8}$$

(notice that L is dependent on A). Moreover, for every $x \in A$ there exists a sequence $\{y_k\}$ such that $y_k \to x$, $y_k \in \mathrm{dom}(\nabla f)$ and the corresponding sequence of gradients $\{\nabla f(y_k)\}$, due to (A.8), is bounded. Thus, the sequence $\{\nabla f(y_k)\}$ has accumulation points. This leads us to the observation that the set

$$S(x) = \{z \in \mathscr{R}^n : \nabla f(y_k) \to z, \text{ for some sequence } y_k \to x, \\ y_k \in \mathrm{dom}(\nabla f)\}$$

is not empty, bounded and closed. Based on $S(x)$ we define

$$\partial f(x) = \mathrm{co}(S(x)),$$

which in addition is convex.

The set $\partial f(x)$ is called the *subdifferential* (or *generalized gradient*) of f at x and any of its element $g \in \partial f(x)$ a *subgradient* of f at x. The subdifferential, according to the above analysis, can be restated as

$$\partial f(x) = \mathrm{co}\left(\{z \in \mathscr{R}^n : \nabla f(y_k) \to z, \text{ for some sequence } y_k \to x, \\ y_k \in \mathrm{dom}(\nabla f)\}\right).$$

The important properties of the subdifferential are listed in the following theorem. To state some of the properties of ∂f we need the definition of the locally bounded point-to-set mapping. We say that the mapping $g : S \to 2^S$ is locally bounded if for any bounded $A \subset S$ the set $\{z \in g(y) : y \in A\}$ is bounded. Here, 2^S denotes all subsets of S. Furthermore, $g : S \to 2^S$ is upper semicontinuous if for any sequence $\{x_k\}$ converging to x each accumulation point g of the sequence $\{g_k\}$, where $g_k \in g(x_k)$, belongs to $g(x)$.

Theorem A.4. *Suppose that $f : \mathscr{R}^n \to \mathscr{R}$ is locally Lipschtzian. Then:*

(i) $\partial f(x)$ is a nonempty convex compact set for any $x \in \mathscr{R}^n$.
(ii) The point-to-set mapping $\partial f(\cdot) : \mathscr{R}^n \to 2^{\mathscr{R}^n}$ is locally bounded.
(iii) The mapping $\partial f(\cdot)$ is upper semicontinuous.

The set, where locally Lipchitzian function is differentiable, is dense. Therefore, the natural question is when a subdifferential reduces to a single vector.

Theorem A.5. *Suppose that $f : \mathscr{R}^n \to \mathscr{R}$ is locally Lipschitzian. Then the following are equivalent:*

(i) $\partial f(x)$ consists of a single vector.
(ii) $\nabla f(x)$ exists and is continuous at x relative to $\mathrm{dom}(\nabla f)$.
(iii) f is strictly differentiable at x.

If one of the conditions in Theorem A.5 holds then we assume that $\partial f(x) = \{\nabla f(x)\}$.

In general, $f'(x;d) < f^\circ(x;d)$ for a locally Lipschitz function $f : \mathscr{R}^n \to \mathscr{R}$. When

$$f'(x;d) = f^\circ(x;d) \tag{A.9}$$

for all $d \in \mathscr{R}^n$ we say that f is subdifferentially regular. Below we give two important cases when (A.9) holds.

Theorem A.6. *Suppose that f is convex, then*

$$f'(x;d) = \max\left\{g^T d : \ g \in \partial f(x)\right\} \quad \text{for all } \ x,d \in \mathscr{R}^n \tag{A.10}$$

and f is subdifferentially regular.

Theorem A.7. *Assume that functions $f_i : \mathscr{R}^n \to \mathscr{R}$ are continuously differentiable where $i \in I$ and I is a finite set. Then*

$$f(x) = \max_{i \in I} f_i(x)$$

is subdifferentially regular and

$$\begin{aligned} f'(x;d) &= \max_{i \in I(x)} \langle \nabla f_i(x), d \rangle \quad \text{for all } \ x,d \in \mathscr{R}^n \,, \\ \partial f(x) &= \operatorname{co}\left(\{\nabla f_i(x) : \ i \in I(x)\}\right) \quad \text{for all } \ x \in \mathscr{R}^n \,, \end{aligned}$$

where $I(x) = \{i \in I : \ f(x) = f_i(x)\}$.

Appendix B
Elements of Linear Algebra

B.1 Vector and Matrix Norms

The Euclidean space of n dimensional vectors we have denoted by $\mathscr{R}^n$. By analogy, a matrix with n rows and m columns belongs to the space we denote as $\mathscr{R}^{n\times m}$.

If $A \in \mathscr{R}^{n\times m}$, then by $A^T \in \mathscr{R}^{m\times n}$ we mean the matrix transposed to A obtained by replacing columns of A by its rows. We say that a matrix $A \in \mathscr{R}^{n\times m}$ is *symmetric* if $A^T = A$. If $A \in \mathscr{R}^{n\times n}$ then A defines the quadratic form

$$q(x) = x^T Ax, \quad x \in \mathscr{R}^n.$$

A quadratic form q is *positive definite* (in that case we also say that A is positive definite) if

$$x^T Ax > 0 \quad \text{for all} \;\; x \neq 0, \tag{B.1}$$

and *positive semidefinite* if $x^T Ax \geqslant 0$ for all $x \in \mathscr{R}^n$. On the other hand, q is *negative* definite provided that

$$x^T Ax < 0 \quad \text{for all} \;\; x \neq 0,$$

and *negative semidefinite* if $x^T Ax \leqslant 0$ for all $x \in \mathscr{R}^n$. In any other case we say that q is *indefinite*.

Besides the Euclidean norm the following norms are frequently used for a vector with n components:

$$\|x\|_1 = \sum_{i=1}^{n} |(x)_i|,$$
$$\|x\|_\infty = \max_{i=1,\ldots,n} |(x)_i|.$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

The norm $\|x\|_\infty$ is called the *Chebyshev norm*. Any of these norms, including the Euclidean norm, can be used to define the matrix p-norms for a matrix $A \in \mathscr{R}^{n\times m}$:

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}\,, \tag{B.2}$$

where $p = 1, 2$, or $p = \infty$. In general a matrix norm $\|\cdot\| : \mathscr{R}^{n\times m} \to \mathscr{R}$ must satisfy the following conditions:

1. $\|A\| \geqslant 0$ for all $A \in \mathscr{R}^{n\times m}$ and $\|A\| = 0$ if and only if $A = 0$.
2. $\|A + B\| \leqslant \|A\| + \|B\|$ for all $A, B \in \mathscr{R}^{n\times m}$.
3. $\|\alpha A\| = |\alpha|\|A\|$ for all $A \in \mathscr{R}^{n\times m}$ and $\alpha \in \mathscr{R}$.

One can show that (B.2) and the formula

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2}\,,$$

which defines the *Frobenius norm*, fulfill the conditions (1)–(3).

One of the most important properties of the p-norm is

$$\|Ax\|_p \leqslant \|A\|_p \|x\|_p\,,$$

the other one holds for any matrix norm

$$\|AB\| \leqslant \|A\|\|B\|, \quad A \in \mathscr{R}^{n\times m},\ B \in \mathscr{R}^{m\times q}.$$

Furthermore, for $\|A\|_1$ and $\|A\|_\infty$ we have analytically tractable formulae:

$$\|A\|_1 = \max_{j=1,\ldots,m} \sum_{i=1}^{n} |a_{ij}|,$$
$$\|A\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|.$$

The norm $\|A\|_2$ is related to eigenvalues of A and is discussed in the next section.

B.2 Spectral Decomposition

A number λ is an *eigenvalue* of $A \in \mathscr{R}^{n\times n}$ if there exists a nonzero vector v such that

$$Av = \lambda v.$$

Vector v is called an *eigenvector* of A corresponding to λ. In general, eigenvalues of a matrix $A \in \mathscr{R}^{n\times n}$ can be complex numbers, then also eigenvectors can have

complex components. However, if a matrix $A \in \mathscr{R}^{n\times n}$ is symmetric then we have the following theorem.

Theorem B.1. *Suppose that a matrix $A \in \mathscr{R}^{n\times n}$ is symmetric:*

(i) If λ is an eigenvalue of A with the corresponding vector v, then $\lambda \in \mathscr{R}$ and $v \in \mathscr{R}^n$.
(ii) If λ and $\tilde{\lambda}$ are eigenvalues of A with corresponding eigenvectors v and $\tilde{v}$ respectively, then $\lambda \neq \tilde{\lambda}$ implies $v^T\tilde{v} = 0$ (vectors v and $\tilde{v}$ are orthogonal).

We say that a matrix $Q \in \mathscr{R}^{n\times n}$ is *orthogonal*, if

$$Q^TQ = I \quad \text{and} \quad QQ^T = I \tag{B.3}$$

(in other words, when columns of A are vectors mutually orthogonal and have the unit Euclidean norm). From Theorem B.1 we can derive the following theorem.

Theorem B.2. *Suppose that $A \in \mathscr{R}^{n\times n}$ is symmetric. Then, there exists the orthogonal matrix $Q \in \mathscr{R}^{n\times n}$ such that*

$$Q^TAQ = \Lambda, \tag{B.4}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

and λ_i, $i = 1,\ldots,n$ are eigenvalues of A.

The immediate consequence of Theorem B.2 is a *spectral decomposition* of a symmetric matrix $A \in \mathscr{R}^{n\times n}$:

$$A = \sum_{i=1}^{n} \lambda_i q_i q_i^T, \tag{B.5}$$

where q_i, $i = 1,\ldots,n$ are columns of the matrix Q appearing in (B.4). In order to derive (B.5) it is sufficient to take into account (B.3) and rewrite (B.4) as

$$A = Q\Lambda Q^T. \tag{B.6}$$

If $A \in \mathscr{R}^{n\times m}$ is not a square matrix then we can refer to its SVD (*Singular Value Decomposition*).

Theorem B.3. *Suppose that $A \in \mathscr{R}^{n\times m}$, then there exist orthogonal matrices $\in \mathscr{R}^{n\times n}$ and $V \in \mathscr{R}^{m\times m}$ such that*

$$U^TAV = \Sigma, \tag{B.7}$$

where

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & 0 \cdots 0 \\ 0 & \sigma_2 & & & 0 \cdots 0 \\ & & \ddots & & \vdots \quad \vdots \\ & & & \sigma_l & 0 \cdots 0 \\ 0 & 0 & \cdots & 0 & 0 \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots \quad \vdots \\ 0 & 0 & \cdots & 0 & 0 \cdots 0 \end{bmatrix},$$

$l = \min[n,m]$*, the left upper submatrix of dimension* l *is diagonal and*

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_l \geqslant 0.$$

Real numbers $\sigma_1, \ldots, \sigma_l$ are called *singular values* of A. If $l = n$ then bottom zero block matrices disappear in the representation of Σ, while $l = m$ rules out the right zero block matrices. Equation (B.7) can be restated as

$$A = U\Sigma V^T = \sum_{i=1}^{l} \sigma_i u_i v_i^T \ ,$$

where u_i, v_i, $i = 1, \ldots, l$ are the first columns of the matrices U and V respectively. The singular value decomposition carries a lot of information about a matrix A but to specify it we need new definitions.

We say that vectors $a_i \in \mathscr{R}^n$, $i = 1, \ldots, m$ are *linearly independent* if the equation

$$\sum_{i=1}^{m} \alpha_i a_i = 0$$

holds only when $\alpha_i = 0$, $i = 1, \ldots, m$. The *rank* of a matrix $A \in \mathscr{R}^{n \times m}$, denoted as $\mathrm{rank}(A)$, is the maximum set of linearly independent columns of A. By $\mathrm{null}(A)$ we mean the *null space* defined as

$$\mathrm{null}(A) = \{v \in \mathscr{R}^m : \ Av = 0\}. \tag{B.8}$$

The *range space* generated by a matrix $A \in \mathscr{R}^{n \times m}$ is denoted by $\mathrm{range}(A)$ and is defined by

$$\mathrm{range}(A) = \{v \in \mathscr{R}^n : \ v = Aw \ \text{ for some } \ w \in \mathscr{R}^m\}. \tag{B.9}$$

The null and range spaces of A are subspaces of the Euclidean spaces $\mathscr{R}^m$ and $\mathscr{R}^n$ respectively. In general, we say that the subset $\mathscr{S} \subset \mathscr{R}^n$ is the *subspace* of $\mathscr{R}^n$ if for any two elements $x, y \in \mathscr{S}$ we have

$$\alpha x + \beta y \in \mathscr{S} \quad \text{for all } \ \alpha, \beta \in \mathscr{R}.$$

Any subspace $\mathscr{S}$ of $\mathscr{R}^n$ is defined by a finite number of vectors in $\mathscr{R}^n$. Suppose that $s_i \in \mathscr{R}^n$, $i = 1, \ldots, m$, then these vectors *span* the subspace $\mathscr{S}$ if

$$\begin{aligned} \mathscr{S} &= \left\{ s \in \mathscr{R}^n : \; s = \sum_{i=1}^{m} \alpha_i s_i \;\text{ where }\; \alpha_i \in \mathscr{R},\; i = 1, \ldots, m \right\} \\ &= \text{span}\{s_1, s_2, \ldots, s_m\}. \end{aligned} \tag{B.10}$$

The set $\{s_i\}_1^m$ of linearly independent vectors which span the subspace $\mathscr{S}$ is the subspace *basis*. In this case, m, the number of vectors in the basis, is called the subspace *dimension* and is denoted by $\dim(A)$.

Now suppose that $A \in \mathscr{R}^{n\times m}$ has the singular value decomposition: $U^T AV = \Sigma$ and

$$\sigma_r > \sigma_{r+1} = \cdots = \sigma_l = 0.$$

then one can show that

$$\begin{aligned} \text{rank}(A) &= r \\ \text{null}(A) &= \text{span}\{v_{r+1}, v_{r+2}, \ldots, v_m\} \\ \text{range}(A) &= \text{span}\{u_1, u_2, \ldots, u_r\} \end{aligned}$$

Here, u_i are columns of the matrix U and v_i columns of the matrix V.

Singular values can also be used to evaluate certain norm matrices. To show that we first refer to interesting properties of orthogonal matrices:

$$\|QAZ\|_F = \|A\|_F \tag{B.11}$$

$$\|QAZ\|_2 = \|A\|_2 \tag{B.12}$$

for any $A \in \mathscr{R}^{n\times m}$ where orthogonal matrices Q and Z have appropriate dimensions. From the singular value decomposition we have $U^T AV = \Sigma$, thus according to (B.11)–(B.12) we can write $\|\Sigma\|_2 = \|A\|_2$ and $\|\Sigma\|_F = \|A\|_F$ which imply that for any $A \in \mathscr{R}^{n\times m}$

$$\|A\|_F = \sqrt{\sum_{i=1}^{l} \sigma_i^2} \tag{B.13}$$

$$\|A\|_2 = \sigma_1 \,. \tag{B.14}$$

Notice that when $A \in \mathscr{R}^{n\times n}$ is symmetric and positive semidefinite then we can show that all eigenvalues of S are nonnegative and singular values and eigenvalues of A are identical. If A is not positive semidefinite then we can refer to the matrix $A^T A$ to define $\|A\|_2$ in terms of eigenvalues of $A^T A$. Since eigenvalues of $A^T A$ coincide with its singular values, thus we have $\|A^T A\|_2 = \sigma_1(A^T A)$ and $\lambda_{\max}(A^T A) = \sigma_1(A^T A)$,

where $\sigma_1(B)$ is the largest singular value and $\lambda_{\max}(B)$ the largest eigenvalue of a matrix B. Since

$$A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma\Sigma V^T,$$

thus $\lambda_{\max}(A^T A) = (\sigma_(A))^2$ and, from (B.14), we have

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}. \tag{B.15}$$

Due to (B.15) the matrix norm $\|\cdot\|_2$ is called *spectral*.

Singular values or eigenvalues can also be used to determine the norm of the matrix inverse. Suppose that $A \in \mathscr{R}^{n\times n}$ is such that there exists the matrix $A^{-1} \in \mathscr{R}^{n\times n}$ satisfying

$$A^{-1}A = I, \quad AA^{-1} = I, \tag{B.16}$$

then the matrix A^{-1} is the *matrix inverse* of the matrix A.

Several matrix inverse properties play important role in matrix computations, in particular

$$\begin{aligned}(AB)^{-1} &= B^{-1}A^{-1}\\ \left(A^{-1}\right)^T &= \left(A^T\right)^{-1} = A^{-T}.\end{aligned}$$

Frequently in optimization we refer to *Sherman–Morrison–Woodbury formula* which gives a convenient representation of the inverse of $(A+ST^T)^{-1}$ where $A \in \mathscr{R}^{n\times n}$ and $S,T \in \mathscr{R}^{n\times k}$:

$$\left(A+ST^T\right)^{-1} = A^{-1} - A^{-1}S\left(I+T^T A^{-1}S\right)^{-1}T^T A^{-1}. \tag{B.17}$$

If $A \in \mathscr{R}^{n\times n}$ is symmetric positive definite and invertible then

$$\left\|A^{-1}\right\|_2 = \frac{1}{\sigma_n(A)},$$

where $\sigma_n(A)$ is the smallest singular value (eigenvalue) of A. We can use this relation together with (B.14) to bound from below and above values of positive definite quadratic forms:

$$\sigma_n(A)\|x\|_2^2 = \frac{\|x\|_2^2}{\|A^{-1}\|_2} \leqslant x^T Ax \leqslant \|A\|_2\|x\|_2^2 = \sigma_1(A)\|x\|_2^2.$$

If $Q \in \mathscr{R}^{n\times n}$ is an orthogonal matrix then its matrix inverse is given by $Q^{-1} = Q^T$ which follows directly from (B.3) and (B.16). Furthermore, we have

$$\|Qx\|_2 = \|x\|_2$$

for any $x \in \mathscr{R}^n$ and any orthogonal matrix $Q \in \mathscr{R}^{n\times n}$. It follows from the evaluation: $\|Qx\|_2^2 = x^T Q^T Q x = x^T x = \|x\|_2^2$.

A subspace of $\mathscr{R}^n$ can be specified by listing vectors spanning it as in (B.10), by given a matrix whose columns are used to define a subspace as in (B.8)–(B.9). Alternatively we can state *normals* of a subspace as a set of linearly independent vectors $q_1, q_2, \ldots, q_{n-k}$. Then the subspace is defined by a set of linear equations:

$$q_i^T x = \rho_i, \quad i = 1, \ldots, n-k, \tag{B.18}$$

where $\rho_i \in \mathscr{R}$. We call the subspace of x satisfying (B.18) a *k-plane*. We denote a k-plane by $\mathscr{P}_k$. Notice that if $x_1 \in \mathscr{P}_k$ then any $x \in \mathscr{P}_k$ satisfies

$$q_i^T (x - x_1) = 0, \quad i = 1, \ldots, n-k$$

and that $u = x - x_1 \in \mathscr{P}_k$, called a *vector* of the $\mathscr{P}_k$ in contrast to x fulfilling (B.18) which we call a *point* of the $\mathscr{P}_k$ plane, has to be a solution of the equations:

$$q_i^T u = 0, \quad i = 1, \ldots, n-k. \tag{B.19}$$

One can show that the subspace $\{u \in \mathscr{R}^n : \ u \text{ satisfies (B.19)}\}$ has a basis consisting of vectors $u_1, \ldots, u_k$, thus any vector u of $\mathscr{P}_k$ can be represented by

$$u = \sum_{i=1}^{k} \alpha_i u_i, \quad \alpha_i \in \mathscr{R},\ i = 1, \ldots, k,$$

and any point of $\mathscr{P}_k$ by

$$x = x_1 + \sum_{i=1}^{k} \alpha_i u_i, \quad \alpha_i \in \mathscr{R},\ i = 1, \ldots, k,$$

where x_1 is some point of $\mathscr{P}_k$.

The subspace $\mathscr{U} = \text{span}\{u_1, \ldots, u_k\}$ has the property that every its vector is orthogonal to every vector of the subspace $\mathscr{Q} = \text{span}\{q_1, \ldots, q_{n-k}\}$, namely, if $u \in \mathscr{U}$ and $q \in \mathscr{Q}$ then $u^T q = 0$. Thus we can write

$$\mathscr{U} = \left\{u \in \mathscr{R}^n : \ u^T q = 0 \ \text{ for all } \ q \in \mathscr{Q}\right\}.$$

We call such a subspace the *orthogonal complement* of the subspace $\mathscr{Q}$ and denote by $\mathscr{Q}^\perp$. Thus we have $\mathscr{U} = \mathscr{Q}^\perp$.

B.3 Determinant and Trace

Suppose that $A \in \mathscr{R}^{n\times n}$ and that its coefficients are given by a_{ij}, $i, j = 1, \ldots, n$ where a_{ij} is the element of A in the ith row and the jth column. The *determinant* of A, denoted by $\det(A)$, is defined by the recursive formula

$$\det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \det(A_{1j}).$$

By A_{1j} we mean the $(n-1)$ by $(n-1)$ matrix obtained by deleting the first row and the jth column of A. The determinant of a scalar $a \in \mathscr{R}^{1\times 1}$ is defined by $\det(a) = a$. The most important properties of the determinant are listed in the following theorem.

Theorem B.4. *Suppose that $A, B \in \mathscr{R}^{n\times n}$. Then,*

$$\det(AB) = \det(A)\det(B) \tag{B.20}$$
$$\det(A^T) = \det(A) \tag{B.21}$$
$$\det(cA) = c^n \det(A) \quad \text{for all } c \in \mathscr{R} \tag{B.22}$$
$$\det(A) \neq 0 \Leftrightarrow A \quad \text{is nonsingular} \tag{B.23}$$
$$\det(A^{-1}) = \frac{1}{\det(A)}. \tag{B.24}$$

Furthermore, if matrix $\tilde{A}$ is obtained from matrix A by substituting some of its column a_i by ca_i, where $c \in \mathscr{R}$, then

$$\det(\tilde{A}) = c\det(A). \tag{B.25}$$

If $Q \in \mathscr{R}^{n\times n}$ is orthogonal then we have $Q^T Q = I$ and from (B.20), $\det(Q)^2 = 1$ since $\det(I) = 1$. This implies that $\det(Q) = \pm 1$.

If $A \in \mathscr{R}^{n\times n}$ is symmetric positive definite then from (B.6) $A = Q\Lambda Q^T$ where Q is orthogonal and Λ diagonal with eigenvalues of A on its diagonal. Then $\det(A) = \det(Q)^2 \det(\Lambda) = \det(\Lambda)$. From (B.25) we have

$$\det(A) = \det(\Lambda) = \prod_{i=1}^{n} \lambda_i \,, \tag{B.26}$$

where λ_i, $i = 1, \ldots, n$ are eigenvalues of A. In fact (B.26) holds for any matrix $A \in \mathscr{R}^{n\times n}$.

Suppose that a matrix $A \in \mathscr{R}^{n\times n}$ has elements a_{ij}, $i, j = 1, \ldots, n$. The *trace* of A, denoted by $\mathrm{trace}(A)$, is defined as

$$\mathrm{trace}(A) = \sum_{i=1}^{n} a_{ii}.$$

One can show that for any $A \in \mathscr{R}^{n\times n}$ the following holds

$$\mathrm{trace}(A) = \sum_{i=1}^{n} \lambda_i \,,$$

where λ_i, $i = 1, \ldots, n$ are eigenvalues of A.

Appendix C
Elements of Numerical Linear Algebra

C.1 Condition Number and Linear Equations

The central problem in numerical algebra is that of solving a nonsingular set of linear equations: having a nonsingular square matrix $A \in \mathscr{R}^{n\times n}$ find a $x \in \mathscr{R}^n$ which is a solution of the equations

$$Ax = b, \tag{C.1}$$

where $b \in \mathscr{R}^n$. Since $\det(A) \neq 0$ there is the unique solution to (C.1) which we denote by $x(0)$. Since the mapping of real data into their floating point arithmetic counterparts introduces errors we are also interested in the perturbed equations

$$(A + \varepsilon F)x = b + \varepsilon f\,, \tag{C.2}$$

where ε assume small positive values. The perturbation of input data A, b is measured by relative errors

$$r(A,\varepsilon) = |\varepsilon|\frac{\|F\|}{\|A\|}, \quad r(b,\varepsilon) = |\varepsilon|\frac{\|f\|}{\|b\|},$$

where the matrix norm is consistent with the vector norm according to (B.2). The solution to (C.2) we denote by $x(\varepsilon)$ provided that we consider small enough values of ε for which $A + \varepsilon F$ are nonsingular.

The essential result associated with (C.2) states that (cf. e.g. [87])

$$\frac{\|x(\varepsilon) - x(0)\|}{\|x(0)\|} \leqslant \kappa(A)\,(r(A,\varepsilon) + r(b,\varepsilon)) + o(\varepsilon),$$

where $o(t)/t \to 0$ when $t \downarrow 0$. Here, $\kappa(A)$ is the *condition number* of a matrix A which is defined by

$$\kappa(A) = \|A\|\|A^{-1}\|. \tag{C.3}$$

R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization,*
Nonconvex Optimization and Its Applications.

The condition number depends on the matrix norm used in (C.3). If we use the spectral norm then $\kappa_2(A)$ can be evaluated as

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_1(A)}{\sigma_n(A)}, \tag{C.4}$$

where we remind that $\sigma_1(A)$ and $\sigma_n(A)$ are the largest and smallest singular values of A. However, we can show that any two condition numbers $\kappa_\alpha(A)$ and $\kappa_\beta(A)$ are equivalent in the following sense

$$c_1 \kappa_\beta(A) \leqslant \kappa_\alpha(A) \leqslant c_2 \kappa_\beta(A)$$

for some positive constants c_1, c_2 and for any $A \in \mathscr{R}^{n\times n}$.

We say that a matrix A is *ill-conditioned* if its condition number $\kappa(A)$ is large. On the other hand matrices with small condition numbers are *well-conditioned*. One can show that for any $p = 1, 2, \infty$ we always have $\kappa_p(A) \geqslant 1$ and if $p = 2$ then $\kappa_2(Q) = 1$ if Q is an orthogonal matrix (it follows directly from (C.4) since $\sigma_1(Q) = \sigma_n(Q) = 1$).

Conditionality cannot be confused with singularity. Following [87] consider the matrix $B \in \mathscr{R}^{n\times n}$:

$$B_n = \begin{bmatrix} 1 & -1 & \cdots & -1 \\ 0 & 1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

which has $\kappa_\infty(B_n) = n2^{n-1}$, so it is ill-conditioned, but $\det(B_n) = 1$. The other example concerns the matrix $D \in \mathscr{R}^{n\times n}$ which is near singular but at the same time is well-conditioned:

$$D_n = \begin{bmatrix} \frac{1}{10} & & & \\ & \frac{1}{10} & & \\ & & \ddots & \\ & & & \frac{1}{10} \end{bmatrix}.$$

We have $\det(D_n) = 10^{-n}$, so for large n the determinant is close to zero, but at the same time $\kappa_2(D_n) = 1$. The condition number measures how big is relative error of the solution to (C.2) in comparison to relative errors of the problem data.

C.2 The LU and Cholesky Factorizations

Equations (C.1) are solved using matrix factorizations. The most commonly applied is the LU factorization which stipulates the matrix representation as the product of an upper triangular matrix U and a lower triangular matrix L:

$$A = LU,$$

where $A, L, U \in \mathscr{R}^{n\times n}$. Then (C.1) can be stated as

$$Ly = b \tag{C.5}$$

$$Ux = y, \tag{C.6}$$

and first equations (C.5) are solved by *forward substitution* (i.e. y_1 is calculated from the first equation, then, having y_1, y_2 from the second and so on) and (C.6) by *backward substitution* (i.e. first x_n is evaluated from the last equation, then, having x_n, x_{n-1} from the $(n-1)$th equation and so on). The cost of solving (C.5)–(C.6) is of the same order $O(n^2)$ as the cost of matrix–vector product [87], while the LU factorization requires $O(n^3)$ floating point operations. (We say that the number of operations is of order $O(n^p)$ if there exists a positive number $c < \infty$ such that $O(n^p) \leqslant cn^p$.)

When $A \in \mathscr{R}^{n\times n}$ is symmetric and positive definite then there exists the representation of A in the form

$$A = C^T C,$$

where C is upper triangular. This is the Cholesky factorization (and C is the Cholesky factor of A) which can be used in the same way as the LU factorization to solve (C.1) (it is sufficient to replace (C.5) by $C^T y = b$ and assume that $U = C$ in (C.6)).

The Cholesky factorization can be accomplished at the half cost of the LU factorization. However, the main benefit of using the Cholesky factorization instead of the LU decomposition are better numerical properties of the former. The LU factorization is achieved by building a sequence of the Gauss transformations M_k (cf. [87] for details), $k = 1, \ldots, n-1$. The matrix M_k is applied to the matrix

$$N_k = M_{k-1} M_{k-2} \cdots M_1 A$$

in order to introduce zeros in the kth column of N_k beginning from row $k+1$. The structure of M_k guarantees that zeros which were introduced to the previous columns (up to column $k-1$) are not effected by M_k. At the end we have $U = N_{n-1}$ and the lower triangular matrix is equal to

$$L = (M_{n-1} M_{n-2} \cdots M_1)^{-1}.$$

The main drawback of the scheme is the step in which we construct the Gaussian matrix M_k which requires the division of $n_{1k}, \ldots, n_{k-1,k}$ by n_{kk} which is called the *pivot*. One can show that small values of pivots can significantly enhance inaccuracies inherent in floating point operations. To overcome that drawback the matrix M_k is built after the rows of N_k are permuted in such a way that the largest element from $n_{1k}, \ldots, n_{k-1,k}, n_{kk}$ is placed in the kth position of the kth column. The LU decomposition algorithm that applies the Gauss transformations mixed with the permutation matrices (which are identity matrices with rows interchanged) is the LU factorization algorithm with pivoting (or partial pivoting since there exists more numerically stable version which applies complete pivoting).

The most widely used algorithms for the Cholesky factorization: *Outer product Cholesky*, *Gaxpy Cholesky algorithm* do not require any form of permutation but at the same time are numerically stable. Furthermore, there are versions which can introduce modifications to the factorized matrix to guarantee that the modified matrix is positive definite in the case it is singular.

Outer product Cholesky algorithm is based on the recursive formula. Suppose that $A \in \mathscr{R}^{n\times n}$ is symmetric and positive definite. Then, A can be stated as

$$\begin{aligned} A &= \begin{bmatrix} B & v \\ v^T & \alpha \end{bmatrix} \\ &= \begin{bmatrix} I_{n-1} & v/\beta \\ 0 & \beta \end{bmatrix} \begin{bmatrix} B - vv^T/\alpha & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{n-1} & 0 \\ v^T/\beta & \beta \end{bmatrix}. \end{aligned}$$

Here, $\beta = \sqrt{\alpha}$ and $\alpha > 0$ since A is positive definite. Important observation is that the matrix $B - vv^T/\alpha$ is positive definite. Indeed, if we introduce the nonsingular matrix X defined as

$$X = \begin{bmatrix} I_{n-1} & 0 \\ -v^T/\alpha & 1 \end{bmatrix},$$

then

$$X^T A X = \begin{bmatrix} B - vv^T/\alpha & 0 \\ 0 & \alpha \end{bmatrix} \tag{C.7}$$

and since $X^T AX$ is positive definite $B - vv^T/\alpha$, as a principal minor in (C.7), must be too. If C_1 is the Cholesky factor of $B - vv^T/\alpha$ then

$$C = \begin{bmatrix} C_1 & v/\beta \\ 0 & \beta \end{bmatrix}$$

is the Cholesky factor of A.

C.3 The QR Factorization

Another matrix decomposition widely used in optimization is the QR factorization. Suppose that $A \in \mathscr{R}^{n\times m}$, then there exist matrices Q and R such that

$$A = QR,$$

where $Q \in \mathscr{R}^{n\times n}$ is an orthogonal matrix, $R \in \mathscr{R}^{n\times m}$ is of the following form

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{C.8}$$

with $R_1 \in \mathscr{R}^{m\times m}$ being upper triangular provided that $m < n$, otherwise

$$\begin{aligned} A &= RQ, \\ R &= [R_1\ 0], \end{aligned}$$

with $Q \in \mathscr{R}^{m\times m}$ and $R_1 \in \mathscr{R}^{n\times n}$ being lower triangular.

The QR factorization always exists and the proof of that is provided in the next sections by presenting algorithms accomplishing the decomposition. The QR factorization of A provides valuable insight into subspaces generated by columns of A.

Theorem C.1. *Suppose that $A = QR$ is a QR factorization of a full column rank $A \in \mathscr{R}^{n\times m}$ and $A = [a_1\ a_2\ \ldots\ a_m]$, $Q = [q_1\ q_2\ \ldots\ q_n]$, then*

$$\begin{aligned} \operatorname{span}\{a_1, a_2, \ldots, a_k\} &= \operatorname{span}\{q_1, q_2, \ldots, q_k\}, \quad k = 1, \ldots, m, \\ \operatorname{range}(A) &= \operatorname{range}(Q_1), \\ \operatorname{range}(A)^{\perp} &= \operatorname{range}(Q_2), \end{aligned}$$

where $Q = [Q_1\ Q_2]$ and $Q_1 \in \mathscr{R}^{n\times m}$.

Here, $\operatorname{range}(A)^{\perp} = \left\{x \in \mathscr{R}^n :\ x^T y = 0,\ \forall y \in \operatorname{range}(A)\right\}$. The proof is given in [87] and is based on the relation

$$a_k = \sum_{i=1}^{k} r_{ik} q_i \in \operatorname{span}\{q_1, \ldots, q_k\}, \tag{C.9}$$

which follows from the representation $A = QR$. The relation (C.9) is important since it can be used to derive an algorithm for the *thin QR factorization*: $A = Q_1 R_1$ which according to the next theorem is unique [87].

Theorem C.2. *Suppose that $A \in \mathscr{R}^{n\times m}$ has full column rank. The thin QR factorization*

$$A = Q_1 R_1$$

is unique with $Q_1 \in \mathscr{R}^{n\times m}$ having orthogonal columns and R_1 being upper triangular with positive elements on its diagonal. Furthermore, $R_1 = C$ where C is the Cholesky factor of the matrix $A^T A$.

There are several algorithms to compute the QR factorization of a given matrix and also we have the choice of algorithms for the thin QR factorization. In many applications the thin QR factorization and the associated algorithms offer the computational advantage over the standard QR factorization since the part Q_2 of Q does not have to be evaluated which saves both the memory storage and floating point operations needed in the case of full Q. Not surprising then that algorithms for the standard and thin QR are significantly different.

C.4 Householder QR

The Householder QR algorithm (for evaluating the QR decomposition) is based on a *Householder reflection* [105] which is an matrix $H \in \mathscr{R}^{n\times n}$ defined by

$$H = I - \frac{2}{v^T v} v v^T ,$$

where v is some vector in $\mathscr{R}^n$. The idea behind a Householder reflection is the same as in the case of a Gauss transformation – find a matrix which transforms a given vector to a vector which has all elements, except one, equal to zero.

For $x \in \mathscr{R}^n$ we have

$$Hx = \left(I - \frac{2vv^T}{v^T v} \right) x = x - \frac{2v^T x}{v^T v} v$$

and if we take $v = x \pm \|x\|_2 e_1$, then (cf. [87] for details)

$$Hx = \mp \|x\|_2 e_1 ,$$

where e_1 is the unit vector. If Householder transformations are applied to a matrix $A \in \mathscr{R}^{n\times m}$ with the aim of making the matrix upper triangular, then the sequence of matrices $H_1, \ldots, H_m$ is constructed and the matrix H_i corresponds to the ith column of A.

Suppose that after the first two steps of the Householder procedure we have the matrix

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} ,$$

where by "$\times$" we denote any real number. In the third step the Householder transform H_3 is constructed of the form

$$H_3 = \begin{bmatrix} I_2 & \\ & \hat{H}_3 \end{bmatrix}$$

with

$$\hat{H}_3 = I_3 - \frac{2}{v_3^T v_3} v_3 v_3^T ,$$

where v_3 is chosen in order to introduce zero elements into the last two positions in the third column of $H_2 H_1 A$. Thus, after three steps we have

$$H_3H_2H_1A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

and the next step will complete the transformation of the matrix A into the form given in (C.8). After m steps we end up with

$$H_mH_{m-1}\ldots H_1A = R$$

and the orthogonal matrix Q is given by

$$Q = H_1^T \ldots H_{m-1}^T H_m^T$$

since a Householder matrix H is orthogonal and the product of orthogonal matrices is also orthogonal. Each Householder transformation H_i is uniquely defined by the corresponding vector v_i. If we restate H_i by

$$H = I - \beta v_i v_i^T$$

with $\beta = 2/v_i^T v_i$ then in order to store H_i it is sufficient to store the *essential part* of $\hat{v}_i$:

$$\hat{v}_i = \begin{bmatrix} 0\,0\,\ldots\,1\;(\hat{v}_i)_{i+1}\;\;(\hat{v}_i)_{i+2}\;\ldots\;(\hat{v}_i)_n \end{bmatrix}^T.$$

Vector $\hat{v}_i$ is characterized by its ith component normalized to 1, so only components of $\hat{v}_i$ from the $(i+1)$th position has to be stored. Conveniently it can be put into the ith subcolumn of $H_iH_{i-1}\ldots H_1A$ which contains zero elements introduced by H_i.

If a Householder matrix $H \in \mathscr{R}^{n\times n}$ is applied to a matrix $A \in \mathscr{R}^{n\times m}$ the cost associated with it is equal to $4mn$ floating point operations. Therefore, the cost of the QR factorization based on Householder matrices is estimated at $2m^2(n-m/3)$ floating point operations. It is higher than the cost of the Gauss elimination method (which uses the LU factorization) which is equal to $2n^3/3$ provided that both methods are applied to a square matrix. If $A \in \mathscr{R}^{n\times n}$ is a symmetric positive definite matrix then the most efficient factorization method, from all the discussed so far, is that calculating the Cholesky factor and described in the previous section – it needs $n^3/3$ floating point operations.

On the other hand, the Householder QR algorithm has much more favorable roundoff properties in comparison to the LU, or the Cholesky methods [87, 102].

C.5 Givens QR

The Givens QR rotation matrices have the following form

$$G(i,k,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}.$$

Elements c are in the intersection of ith row and ith column and kth row and kth column, while element s position is determined by ith row and kth column and element $-s$ position by the kth row and the ith column. Furthermore, we assume that $k > i$ and

$$c = \cos(\theta), \quad s = \sin(\theta).$$

Givens rotations $G(i,k,\theta) \in \mathscr{R}^{n\times n}$ are obviously orthogonal matrices. An angle θ defines a rotation in the (i,k) coordinate plane. Suppose that $x \in \mathscr{R}^n$ and $y = G(i,k,\theta)^T x$, then

$$y_j = \begin{cases} cx_i - sx_k & j = i, \\ sx_i + cx_k & j = k, \\ x_j & j \neq i,k, \end{cases}$$

and if the angle θ is chosen in such a way that

$$c = \frac{x_i}{\sqrt{x_i^2 + x_k^2}}, \quad s = \frac{-x_k}{\sqrt{x_i^2 + x_k^2}},$$

we have $y_k = 0$. This shows that using Givens rotations we can insert zero in any position of a given vector.

Suppose now that a matrix $A \in \mathscr{R}^{n\times m}$ has to be transformed to the form (C.8). To this end we build a sequence of $(n-1)(n-2)\cdots(n-m)$ Givens rotations where each matrix introduces zero to A. To illustrate the procedure take the matrix with five rows and four columns, then the steps are as follows:

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{(2,3)}$$

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{(1,2)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{(3,4)}$$

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{(3,4)}$$

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Assume that we applied t Givens rotations to obtain

$$G_t^T G_{t-1}^T \cdots G_1^T A = R.$$

Since the product of orthogonal matrices $G_1, \ldots, G_t$ is orthogonal we have $A = QR$, where

$$Q = G_1 \cdots G_{t-1} G_t.$$

Each Givens matrix G_i is fully described by one real number c, or s (which are in the relation $c^2 + s^2 = 1$) and two integer numbers i - indicating position in the column which is supposed to be zeroed and the column k in which the element is zeroed. Since the indices i and k indicate the position of an element to which zero is inserted it is sufficient to place the real number defined by c in the zeroed element. Therefore, no additional storage is required when the QR factorization based on Givens matrices is used.

The procedure has similar to the Householder QR algorithm numerical properties but its number of floating point operations is higher at $3m^2(n-m/3)$. For that reason the Householder QR method is more often used than the Givens QR algorithm. On the other hand the Givens rotation, due to its feature of inserting zero to any element of a matrix, is an ideal tool to perform updates of a QR factorization.

For example, suppose that we have the QR factorization of a matrix $B \in \mathscr{R}^{n\times m}$: $A = QR$ and rank-one update is applied to B. We need modification of Q and R in order to get Q_1 and R_1 such that

$$A + uv^T = Q_1 R_1.$$

Here, $u \in \mathscr{R}^n$ and $v \in \mathscr{R}^m$ are arbitrary vectors. First, following [79] (cf. also [87]), we incorporate the rank-one update into R yielding

$$A + uv^T = Q\left(R + wv^T\right)$$

with $w = Q^T u$. Givens rotations $G_1, \ldots, G_{n-1}$ are applied to vector w transforming it into a multiple of e_1:

$$G_1^T G_2^T \cdots G_{n-1}^T w = \pm \|w\|_2 e_1,$$

and this is possible since the product of Givens matrices is an orthogonal matrix, thus it preserves the Euclidean norm. The same Givens rotations transform the matrix R into the upper Hessenberg matrix H (i.e. a matrix whose $m \times m$ principal submatrix differs from an upper triangular matrix by nonzero elements lying just below the diagonal). Thus, in the case of $n = 5$, $m = 4$, we have

$$G_1^T G_2^T \cdots G_{n-1}^T w = \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$G_1^T G_2^T \cdots G_{n-1}^T R = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

In the next step the QR factorization of the matrix

$$G_1^T G_2^T \cdots G_{n-1}^T \left(R \pm \|w\|_2 e_1 v^T\right) = H_1$$

(which is also an upper Hessenberg matrix) is computed with the help of the Givens rotations $\bar{G}_1, \ldots, \bar{G}_m$ yielding

$$\bar{G}_m^T \cdots \bar{G}_1^T H = R_1.$$

Notice now that the product of matrices $G_{n-1} G_{n-2} \cdots G_1 \bar{G}_1 \bar{G}_2 \cdots \bar{G}_m$ is an orthogonal matrix so is the matrix Q_1 defined by

$$Q_1 = Q G_{n-1} G_{n-2} \cdots G_1 \bar{G}_1 \bar{G}_2 \cdots \bar{G}_m.$$

It remains to check that Q_1 and R_1 are the QR factors of $A + uv^T$. Indeed, we have

$$\begin{aligned} A + uv^T &= Q\left(R + wv^T\right) \\ &= Q G_{n-1} G_{n-2} \cdots G_1 \bar{G}_1 \bar{G}_2 \cdots \bar{G}_m \bar{G}_m^T \cdots \bar{G}_1^T \\ &\quad \times G_1^T G_2^T \cdots G_{n-1}^T \left(R + wv^T\right) \\ &= Q_1 R_1 \end{aligned}$$

since Givens matrices are orthogonal. One can show that the described procedure can be accomplished in $14n^2 + 12m^2$ floating point operations.

In the second example we consider the updating procedure for the QR factorization when a column is deleted from a matrix $A \in \mathscr{R}^{n\times m}$. Assuming that

$$A = [a_1\ a_2\ \ldots\ a_k\ \ldots\ a_m]$$

and that a_k is deleted we look for R_1 and Q_1 such that $A_1 = Q_1 R_1$ where A_1 differs from A by the kth column. Initially

$$R = \begin{bmatrix} R_{11} & v & R_{13} \\ & r_{kk} & R_{23} \\ & & R_{33} \end{bmatrix}$$

and when column a_k is removed it is reflected in deleting the kth column from R. The matrix

$$\bar{R} = \begin{bmatrix} R_{11} & R_{13} \\ & R_{23} \\ & R_{33} \end{bmatrix}$$

is upper Hessenberg and can be transformed into an upper triangular matrix by a sequence of Givens rotations G_k, $G_{k+1},\ldots,G_m$ – each G_i introduces zero element into the ith column just below the diagonal. As a result we obtain the matrix R_1:

$$G_m^T \cdots G_{k+1}^T G_k^T \bar{R} = R_1.$$

The corresponding matrix Q_1 is defined by

$$Q_1 = Q G_k G_{k+1} \cdots G_m$$

since we have

$$G_m^T \cdots G_{k+1}^T G_k^T Q^T A_1 = G_m^T \cdots G_{k+1}^T G_k^T Q^T \bar{R} = R_1.$$

C.6 Gram–Schmidt QR

The Gram–Schmidt procedure is aimed at the thin QR factorization. Assuming that $A \in \mathscr{R}^{n\times m}$ we look for matrices $Q_1 \in \mathscr{R}^{n\times m}$ and $R_1 \in \mathscr{R}^{m\times m}$ such that $A = Q_1 R_1$, the columns of Q_1 are mutually orthogonal vectors and R_1 is upper triangular. The basis for the Gram–Schmidt algorithm is (C.9). When solved for q_k it gives

$$q_k = \frac{a_k - \sum_{i=1}^{k-1} r_{ik} q_i}{r_{kk}}. \tag{C.10}$$

If we treat r_{kk} as the normalizing factor then the other coefficients r_{ik} of R_1 are found by stipulating that direction

$$a_k - \sum_{i=1}^{k-1} r_{ik} q_i$$

is orthogonal to the previously determined directions q_i, $i = 1, \ldots, k-1$ which are mutually orthogonal. It means that r_{ik} are given by

$$r_{ik} = a_k^T q_i, \quad i = 1, \ldots, k-1.$$

The kth step of the Gram–Schmidt procedure is completed by setting

$$r_{kk} = \left\| a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right\|_2 .$$

The presented classical Gram–Schmidt algorithm has poor numerical properties which are reflected by the loss of orthogonality of vectors $q_1, \ldots, q_m$. Fortunately, there exists a modified version which needs more floating point operations but, as far as numerical stability is concerned, is significant improvement over the classical version. The modified Gram–Schmidt procedure is based on the following relation which holds under the condition that Q_1 and R_1 are the factors of the thin QR decomposition of A:

$$\begin{aligned} A - \sum_{i=1}^{k-1} q_i r_i^T &= \sum_{i=k}^{m} q_i r_i^T \\ &= [q_k \; q_{k+1} \; \cdots \; q_m] \begin{bmatrix} r_k^T \\ r_{k+1}^T \\ \vdots \\ r_m^T \end{bmatrix} \\ &= [q_k \; q_{k+1} \; \cdots \; q_m] \begin{bmatrix} 0 \ldots 0 & r_{kk} & r_{k,k+1} & \cdots & r_{km} \\ 0 \ldots 0 & & r_{k+1,k+1} & \cdots & r_{k+1,m} \\ \vdots \quad \vdots & & & \ddots & \vdots \\ 0 \ldots 0 & & & & r_{mm} \end{bmatrix}. \end{aligned}$$

Here, r_i^T is the ith row of the matrix R_1. One can show that

$$\sum_{i=k}^{m} q_i r_i^T = \left[0 \; z^{(k)} \; B^{(k)} \right], \tag{C.11}$$

where $B^{(k)} \in \mathscr{R}^{n \times (m-k)}$ and $z^{(k)} \in \mathscr{R}^n$. At the kth step of the modified Gram–Schmidt algorithm vectors $q_i, \ldots, q_{k-1}$ and $r_1^T, \ldots, r_{k-1}^T$ are already available thus the matrix $B^{(k)}$ and vector $z^{(k)}$ can be evaluated. The relation (C.10) allows to assume that

$$r_{kk} = \left\| z^{(k)} \right\|_2, \quad q_k = \frac{1}{r_{kk}} z^{(k)}.$$

On the other hand (C.11) and the fact that q_k is orthogonal to $q_{k+1}, \ldots, q_m$ enable us to take

$$r_k^T = q_k^T \left[0 \; z^{(k)} \; B^{(k)} \right]$$

which together with q_k we need to proceed to the next step.

The computational cost of the modified Gram–Schmidt algorithm is equal to $2nm^2$ while the cost of the Householder QR procedure is almost twice as much, when the procedure computes also explicitly the Q_1 part of the matrix Q, since it is estimated at $4nm^2 - 4m^3/3$. Therefore, if columns of a matrix A are fairly linearly independent the modified Gram–Schmidt QR is preferable over the Householder QR.

References

1. ADACHI, N. 1973 On the uniqnuess of search directions in variable–metric algorithms, *J. Optim. Theory Applications*, Vol. 11, 590–604.
2. AL–BAALI, M. 1985 Decent property and global convergence of the Fletcher-Reeves Method with inexact line search. *IMA J. Numer. Anal.* Vol. 5, 121–124.
3. AL–BAALI, M. & FLETCHER, R. 1984 An efficient line search for nonlinear least squares, *J. Optim. Theory and Applications*, Vol. 48, 359–377.
4. AL–BAALI, M. & FLETCHER, R. 1996 On the order of convergence of preconditioned nonlinear conjugate gradient methods, *SIAM J. Sci. Comput.*, Vol. 17, 658–665.
5. ALTMAN, M. 1959 On the convergence of the conjugate gradient method for non–bounded linear operators in Hilbert spaces, in *Approximation methods in functional analysis*, Lecture Notes, California Institute of Technology, 33–36.
6. BEALE, E.M.L. 1972 A derivation of conjugate gradients, in *Numerical methods for nonlinear optimization*, F.A. Lootsma ed., Academic Press, London, 39–43.
7. BERTSEKAS, D.P. 1974 Partial conjugate gradient methods for a class of optimal control problems, *IEEE Trans. on Auto. Control*, Vol. 19, 209–217.
8. BERTSEKAS, D.P. 1976 On the Goldstein–Levitin–Polyak gradient projection method, *IEEE Trans. Auto. Control*, Vol. 21, 174–184.
9. BERTSEKAS, D.P. 1982 *Constrained optimization and lagrange multipliers methods*, Academic Press, NY.
10. BERTSEKAS, D.P. 1982 Projected Newton methods for optimization problems with simple constraints, *SIAM J. Control and Optimization*, Vol. 20, 221–245.
11. BERTSEKAS, D.P. & MITTER, S.K. 1973 A descent numerical method for optimization problems with nondifferentiable cost functionals, *SIAM J. Control*, Vol. 11, 637–652.
12. BIHAIN, A. 1984 Optimization of upper semidifferentiable functions, *J. Optimization Theory Applications*, Vol. 44, 545–568.
13. BONGARTZ, I. CONN, A.R., GOULD, N.I.M. & TOINT, PH.L. 1994 *CUTE: Constrained and unconstrained testing environment*, Research Report RC 18860, IBM T.J. Watson Research Center, Yorktown, USA.
14. BRODLIE, K.W., GOURLAY, A.R. & GREENSTADT, J. 1973 Rank-one and rank-two corrections to positive definite matrices expressed in product form, *IMA J Appl Math.*, Vol. 11, 73–82.
15. BROYDEN, C.G. 1965 A class of methods for solving nonlinear simultaneous equations, *Math. of Comp.*, Vol. 19, 577–593.
16. BROYDEN, C.G. 1967 Quasi-Newton methods and their application to function minimization, *Math. of Comp.*, Vol. 21, 368–381.
17. BROYDEN, C.G. 1970 The convergence of a class of double–rank minimization algorithms, *J. Inst. Math. Applics.*, Vol. 6, 76–90.

18. BUCKLEY, A. 1978 A combined conjugate–gradient quasi-Newton minimization algorithm, *Math. Program.*, Vol. 15, 200–210.
19. BUCKLEY, A. 1978 Extending the relationship between the conjugate gradient and BFGS algorithms, *Math. Program.*, Vol. 15, 343–348.
20. BUCKLEY, A. & LENIR, A. 1983 QN–like variable storage conjugate gradients, *Math. Program.*, Vol. 27, 155–175.
21. BUCKLEY, A. & LENIR, A. 1985 BBVSG–a variable storage algorithm for function minimization, *ACM Transactions on Mathematical Software*, Vol. 11, 103–119.
22. BURKE, J.V. 1990 On the identification of active constraints II: the nonconvex case, *SIAM J. Numer. Anal.*, Vol. 27, 1081–1102.
23. BURKE, J.V. & J.J. MORÉ, J.J. 1988 On the identification of active constraints, *SIAM J. Numer. Anal.*, Vol. 25, 1197–1211.
24. BURKE, J.V. & MORÉ, J.J. 1994 Exposing constraints, *SIAM J. Optimization*, Vol. 4, 573–595.
25. BURKE, J.V. MORÉ, J.J. & TORALDO, G. 1990 Convergence properties of trust region methods for linear and convex constraints, *Mathematical Programming*, Vol. 47, 305–336.
26. BYRD, R.H., LU, P., NOCEDAL, J. & ZHU, C. 1995 A limited memory algorithm for bound constrained optimization, *SIAM J. Scientific Computing*, Vol. 16, 1190–2108.
27. BYRD, R.H. & NOCEDAL, J. 1989 A tool for the analysis of quasi-Newton methods with application to unconstrained minimization, *SIAM J. Numer. Anal.*, Vol. 26, 727–739.
28. BYRD, R.H., NOCEDAL, J. & SCHNABEL, R.B. 1996 Representations of quasi-Newton matrices and their use in the limited memory methods, Technical Report NAM–03, Northwestern University.
29. BYRD, R.H., NOCEDAL, J. & YUAN, Y. 1987 Global convergence of a class of quasi-Newton methods on convex problems, *SIAM J. Numer. Anal.*, Vol. 24, 1171–1190.
30. CALAMAI, P.H. & MORÉ, J.J. 1987 Projected gradient methods for linearly constrained problems, *Math. Program.*, Vol. 39, 93–116.
31. CHENEY, C.C. 1966 *Introduction to Approximation Theory*, McGraw Hill, NY.
32. CLARKE, F.H. 1975 Generalized gradients and applications, *Trans. Am. Math. Soc.*, Vol. 205, 247–262.
33. COHEN, A.I. 1972 Rate of convergence of several conjugate gradient algorithms, *SIAM J. Numer. Anal.*, Vol. 9, 248–259.
34. COLVILLE, A.R. 1968 A Comparative study of nonlinear programming codes, *IBM New York Scientific Centre Report*, 320–2949.
35. CONN, A.R., GOULD, N.I.M. & TOINT, PH.L. 1988 Global convergence of a class of trust region algorithms for optimization with simple bounds, *SIAM J. Numer. Anal.*, Vol. 28, 433–460.
36. CONN, A.R., GOULD, N.I.M. & TOINT, PH.L. 1988 Testing a class of methods for solving minimization problems with simple bounds on the variables, *Mathematics of Computation*, Vol. 50, 399–430.
37. CONN, A.R., GOULD, N.I.M. & TOINT, PH.L. 1992 *LANCELOT: a FORTRAN package for large–scale nonlinear optimization (Release A)*, Number 17 in Springer Series in Computational Mathematics, Springer–Verlag, New York.
38. CROWDER, H. & WOLFE, P. 1972 Linear convergence of the conjugate gradient method, *IBM Journal of Research and Development*, Vol. 16, 431–433.
39. CURTIS, J.H. 1954 A generalization of the method of conjugate gradients for solving systems of linear algebraic equations, *Math. Tables and Aids to Comp.*, Vol. 8, 189–193.
40. DAI, Y.H. 1997 Analysis of conjugate gradient methods, PhD thesis, Institute of Computational mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences.
41. DAI, Y.H. & LIAO, L.Z. 2001 New conjugate conditions and related nonlinear conjugate gradient methods, *Appl. Math. Optim.*, Vol. 43, 87–101.
42. DAI, Y.H. & YUAN, Y. 1996 Convergence properties of the Fletcher–Reeves method, *IMA J. Numer. Anal.*, Vol. 16, 155–164.
43. DAI, Y.H. & YUAN, Y. 1999 Global convergence of the method of shortest residuals, *Numer. Math.*, Vol. 83, 581–598.

44. DAI, Y.H. & YUAN, Y. 2000 *Nonlinear conjugate gradient methods*, Shanghai Science and Technology Publisher, Beijing.
45. DAI, Y.H. & YUAN, Y. 2001 A three–parameter family of hybrid conjugate gradient method, *Math. Comp.*, Vol. 70, 1155–1167.
46. DANIEL, J.W. 1967 The conjugate gradient method for linear and nonlinear operator equations, *SIAM J. Numer. Anal.*, Vol. 4, 10–26.
47. DANIEL, J.W. 1967 Convergence of the conjugate gradient method with computationally convenient modifications, *Numer. Math.*, Vol. 10, 125–131.
48. DANIEL, J.W. 1970 *The approximate minimization of functionals*, Prentice–Hall, Englewood Cliffs, New Jersey.
49. DANIEL, J.W. 1970 A correction concerning the convergence rate for the conjugate gradient method, *SIAM J. Numer. Anal.*, Vol. 7, 277–280.
50. DAVIDON, W.C. 1959 variable metric method for minimization, Report ANL–5990, Argonne National Laboratory, Argonne, Illinois, finally published *SIAM J. Optimization*, Vol. 1, 1991, 1–17.
51. DAVIDON, W.C. 1975 Optimally conditioned optimization algorithms without line search, *Math. Program.*, Vol. 9, 1–30.
52. DEMYANOV, V.F. 1968 Algorithms for some minimax problems, *Journal of Computer and Systems Science*, Vol. 2, 342–380.
53. DENNIS, J.E. & MORÉ, J.J. 1977 quasi-Newton methods: motivation and theory, *SIAM Review*, Vol. 19, 46–89.
54. DENNIS, J.E. & SCHNABEL, R.B. 1981 A new derivation of symmetric positive definite secant updates, in *Nonlinear Programming 4*, O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds., Academic Press, London, New York, 167–199.
55. DENNIS, J.E. & SCHNABEL, R.B. 1983 *Numerical methods for unconstrained minimization*, Prentice–Hall, Englewood Cliffs, NJ, 1983. Reprinted by SIAM Publications.
56. DIXON, L.C. 1972 Quasi-Newton algorithms generate identical points, *Math. Program.*, Vol. 2, 383–387.
57. DOLAN, E.D. & MORÉ, J.J. 2002 Benchmarking optimization software with performance profiles, *Math. Program.*, Vol. 91, 201–213.
58. DUNN, J.C. 1987 On the convergence of projected gradient processes to singular critical points, *J. of Optim. Theory and Applications*, Vol. 55, 203–216.
59. DUNN, J.C. 1988 Gradient projection methods for systems optimization problems, *Control and Dynamic Systems*, Vol. 29, 135–195.
60. ENGELI, M., GINSBURG, TH., RUTISHAUSER, H. & STIEFEL, E. 1959 *Refined iterative methods for computation of the solution and the eigenvalues of self–adjoint boundary value problems*, Birkhauser Verlag, Basel/Stuttgart.
61. FACCHINEI, F, FISCHER, A. & KANZOW, C. 1998 On the accurate identification of active cosntraints, *SIAM J. Optimization*, Vol. 9, 14–32.
62. FACCHINEI, F, JUDICE, J. & SOARES, J. 1998 An active set Newton algorithm for large–scale nonlinear programs with box constraints, *SIAM. J. Optimization*, Vol. 8, 158–186.
63. FACCHINEI, F, LUCIDI, S. & PALAGI, L. 2002 A truncated Newton algorithm for large scale box constrained optimization, *SIAM J. Optimization*, Vol. 12, 1100–1125.
64. FEDER, D.P. 1962 Automatic lens design with a high–speed computer, *Journal of the Optical Society of America*, Vol. 52, 177–183.
65. FENELON, M.C. 1981 Preconditioned conjugate–gradient–type algorithm for large–scale unconstrained optimization, PhD thesis, Department of Operations Research, Stanford University, Stanford, CA.
66. FISCHBACH, J.W. 1956 Some applications of gradient methods, *Proceedings of the Sixth Symposium in Applied Mathematics 1953*, McGraw–Hill, New York, 59–72.
67. FLETCHER, R. 1965 Function minimization without evaluating derivatives–a review, *Computer J.* Vol. 8, 33–41.
68. FLETCHER, R. 1970 A new approach to variable metric algorithms, *Computer J.*, Vol. 13, 317–322.

69. FLETCHER, R. 1976 Conjugate gradient methods for indefinite systems, in *Numerical Analysis Dundee 1975*, ed. G.A. Watson, Springer–Verlag, New York, 73–89.
70. FLETCHER, R. 1980 *Practical optimization. Vol. 1, unconstrained optimization*, J. Wiley, Chichester.
71. FLETCHER, R. 1987 *Practical Methods of Optimization*, J. Wiley, Chichester.
72. FLETCHER, R. & POWELL, M.J.D. 1963 A rapidly convergent descent method for minimization, *Computer J.*, 163–168.
73. FLETCHER, R. & REEVES, C.M. 1964 Function minimization by conjugate gradients, *Comp. J.*, Vol. 7, 149–154.
74. FORSYTHE, G.E., HESTENES, M.R. & ROSSER, J.B. 1951 Iterative methods for solving linear equations, *Bull. Amer. Math. Soc.*, Vol. 57, 480.
75. FOX, L., HUSKEY, H.D. & WILKINSON, J.H. 1948 Notes on the solution of algebraic linear simultaneous equations, *Quart. J. of Mech. and Appl. Math.*, Vol. 1, 149–173.
76. GAFNI, E.M. & BERTSEKAS, D.P. 1984 Two-metric projection methods for constrained optimization, *SIAM J. Control and Optimization*, Vol. 22, 936–964.
77. GILBERT, J. CH. & LEMARÉCHAL, C. 1989 Some numerical experiments with variable–storage quasi-Newton algorithms, *Math. Program.*, Vol. 45, 407–436.
78. GILBERT, J.CH. & NOCEDAL, J. 1992 Global convergence properties of conjugate gradient methods for optimization, *SIAM J. Optimization*, Vol. 2, 21–42.
79. GILL, P.E., GOLUB, G.H., MURRAY, W. & SAUNDERS, N.M. 1974 Methods for modifying matrix factorizations, *Mathematics of Computations*, Vol. 28, 505–535.
80. GILL, P.E. & LEONARD, M.W. 2001 Reduced-Hessian quasi-Newton methods for unconstrained optimization, *SIAM J. Optimization*, Vol. 12, 209–237.
81. GILL, P.E. & LEONARD, M.W. 2003 Limited–memory reduced-Hessian methods for large–scale unconstrained optimization, *SIAM J. Optimization*, Vol. 14, 380–401.
82. GILL, P.E., MURRAY, W., SAUNDERS, M.A. & WRIGHT, M.H. 1986 User's guide for NPSOL (version 4.0): A Fortran package for nonlinear programming, Report SOL86-2, Department of Operations Research, Stanford University, Stanford, CA.
83. GOLDFARB, D. 1970 A family of variable metric methods derived by variational means, *Math. Computation*, Vol. 24, 23–26.
84. GOLDFARB, D. 1976 Factorized variable metric methods for unconstrained optimization, *Mathematics of Computations*, Vol. 30, 796–811.
85. GOLDSTEIN, A.A. 1964 Convex programming in Hilbert spaces, *Bull. Amer. Math. Soc.*, Vol. 70, 709–710.
86. GOLUB, G. & O'LEARY, D.P. 1989 Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, Vol. 31, 50–102.
87. GOLUB, G. & VAN LOAN, CH.F. 1996 *Matrix computations*, The Johns Hopkins University Press, Baltimore.
88. GRIPPO, L. & LUCIDI, S. 1997 A globally convergent version of the Polak–Ribière conjugate gradient method, *Math. Program.*, Vol. 78, 375–391.
89. GRIPPO, L. & LUCIDI, S. 2003 Convergence conditions, line search algorithms and trust implementations for the Polak–Ribière conjugate gradient method, Technical Report 25–03, Dipartimento di Informatica e Sistematica, Universitèa di Roma "La Sapienza".
90. HAGER, W.W. 1988 *Applied linear numerical algebra*, Prentice–Hall, Englewood Cliffs, NJ.
91. HAGER, W.W. & ZHANG, H. 2005 A new conjugate gradient method with guaranted descent and an efficient line search, *SIAM J. Optimization*, Vol. 16, pp. 170–192.
92. HAGER, W.W. & ZHANG, H. 2006 A survey of nonlinear conjugate gradient methods, *System modeling and optimization*, eds. F. Ceragioli et al, Proceedings of the 22nd IFIP Conference, Turin, 2006, Springer–Verlag, 67–82.
93. HAGER, W.W. & ZHANG, H. 2006 A new active set algorithm for box constrained optimization problems, *SIAM J. Optimization*, Vol. 17, 526–557.
94. HELD, M., WOLFE, P. & CROWDER, H.P. 1974 Validation of subgradient optimization, *Math. Program.*, Vol. 6, 62–88.

95. HESTENES, M.R. 1951 Iterative methods for solving linear equations, NAML Report 52–9, July 2, 1951, National Bureau of Standards, Los Angeles, California, also *J. Optim. Theory and Applications*, Vol. 11, 323–334.
96. HESTENES, M.R. 1955 Iterative computational methods, *Communications on Pure and Applied Mathematics*, Vol. 8, 85–96.
97. HESTENES, M.R. 1956 The conjugate–gradient method for solving linear systems, *Proceedings of the Sixth Symposium in Applied Mathematics 1953*, McGraw–Hill, New York, 83–102.
98. HESTENES, M.R. 1956 Hilbert space methods in variational theory and numerical analysis, *Proceedings of the International Congress of Mathematicians 1954*, North–Holland, Amsterdam, 229–236.
99. HESTENES, M.R. 1969 Multiplier and gradient methods, *J. Optim. Theory and Applications*, Vol. 4, 303–320.
100. HESTENES, M. R. 1980 *Conjugate direction methods in optimization*, Springer–Verlag, New York, NY.
101. HESTENES, M.R. & STIEFEL, E. 1952 Methods of conjugate gradients for solving linear systems, *Journal of Research of the National Bureau of Standards*, Vol. 29, 409–439.
102. HIGHAM, N.J. 1996 *Accuracy and stability of numerical algorithms*, SIAM Publications, Philadelphia, PA.
103. HIMMEBLAU, D.H. 1972 *Applied nonlinear programming.* McGraw Hill, New York.
104. HIRIART–URRUTY, J.–B. & LEMARÉCHAL, C. 1993 *Convex analysis and minimization algorithms*, Springer–Verlag, Berlin, New York.
105. HOUSEHOLDER, A.S. 1964 *The theory of matrices in numerical analysis*, Blaisdell Publishing Co., New York.
106. HUANG, H.Y. 1970 Unified approach to quadratically convergent algorithms for function minimization, *J. Optim. Theory and Applications*, Vol. 5, 405–423.
107. HUANG, H.Y. 1974 Method of dual matrices for function minimization, *J. Optim. Theory and Applications*, Vol. 13, 519–537.
108. KELLEY, J.E. 1960 The cutting plane method for solving convex problems, *Journal of the Society for Industrial and Applied Mathematics*, Vol. 8, 703–712.
109. KIWIEL, K. 1983 An Aggregate subgradient method for nonsmooth convex minimization. *Math. Program.*, Vol. 27, 320–341.
110. KIWIEL, K. 1985 *Methods of descent for nondifferentiable optimization.* Lecture Notes in Mathematics 1133, Springer-Verlag: Berlin.
111. KOLDA, T.G., O'LEARY, D.P. & NAZARETH, L. 1998 BFGS with update skipping and varying memory, *SIAM J. Optimization*, Vol. 8, 1060–1083.
112. LALEE, M. & NOCEDAL, J. 1993 Automatic column scaling strategies for quasi-Newton methods, *SIAM J. Optimization*, Vol. 3, 637–653.
113. LANCZOS, C. 1950 An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bur. Standards*, Vol. 45, 252–282.
114. LANCZOS, C. 1952 Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Standards*, Vol. 49, 33–53.
115. LANCZOS, C. 1955 Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Standards*, Vol. 49, 33–53.
116. LASDON, L.S., MITTER, S.K. & WAREN, A.D. 1967 The conjugate gradient method for optimal control problems, *IEEE Trans. on Auto. Control*, Vol. 12, 132–138.
117. LE, D. 1985 A fast and robust unconstrained optimization method requiring minimum storage. *Math. Program.*, Vol. 32, 41–68.
118. LEMARÉCHAL, C. 1975 An extension of Davidon methods to nondifferentiable Problem, in *Mathematical Programming Study*, Vol. 3 , M.L. Balinski and P. Wolfe eds., North–Holland: Amsterdam, 95–109.
119. LEMARÉCHAL, C. 1978 Nonsmooth optimization and descent methods. RR-78-4, International Institute for Applied Systems Analysis (Laxenburg, Austria, March).

120. LEMARÉCHAL, C. 1981 A view of line searches, in *Optimization and optimal control, Lecture Notes in Control and Information Science 30*, (A. Auslander, W. Oettli and J. Stoer, eds), Springer–Verlag, Berlin, 59–78.
121. LEONARD, M.W. 1995 Reduced Hessian quasi-Newton methods for optimization, PhD theis, Department of Mathematics, University of California, San Diego, CA.
122. LEVITIN, E.S. & POLYAK, B.T. 1966 Constrained minimization methods, *Z. Vycisl. Mat. i Mat. Fiz.*, Vol. 6, 787–823 (in Russian).
123. LIN, C-J. & MORÉ, J.J. 1999 Newton's method for large bound–constrained optimization problems, *SIAM J. Optimization*, Vol. 9, 1100–1127.
124. LIU, Y. & STOREY, C. 1991 Efficient generalized conjugate gradient algorithms, Part I: Theory, *J. Optim. Theory Applications*, Vol. 69, 129–137.
125. LIUA, D.C. & NOCEDAL, J. 1989 On the limited memory BFGS method for large scale optimization problems, *Math. Program.*, Vol. 45, 503–528.
126. LUENBERGER, D.G. 1969 Hyperbolic pairs in the method of conjugate gradients, *SIAM J. Appl. Math.*, Vol. 17, 1263–1267.
127. LUENBERGER, D.G. 1970 The conjugate residual method for constrained minimization problems, *SIAM J. Numer. Anal.*, Vol. 7, 390–398.
128. LUENBERGER, D.G. 1973 *Introduction to linear and nonlinear programming*, Addison Wesley, Reading, MA.
129. MATTHIES, H. & STRANG, G. 1979 The solution of nonlinear finite element equations, *International Journal of Numerical Methods in Engineering* 14, 1613–1626.
130. MCCORMICK, G.P. & RITTER, K. 1972 Methods of conjugate directions vs. quasi-Newton methods, *Math. Program.*, Vol. 3, 101–116.
131. MCCORMICK, G.P. & RITTER, K. 1974 Alternate proofs of the convergence properties of the conjugate–gradient method, *J. Optim. Theory and Applications*, Vol. 13, 497–518.
132. MCCORMICK, G.P. & TAPIA, R.A. 1972 The gradient projection method under mild differentiability conditions, *SIAM J. Control*, Vol. 10, 93–98.
133. MIFFLIN, R. 1977 An algorithm for constrained optimization with semismooth functions, *Math. Oper. Res.*, Vol. 2, 191–207.
134. MIFFLIN, R. 1982 A modification and an extension of Lemaréchal's algorithm for nonsmooth minimization, in D.C. Sorensen and R.J-B. Wets , eds, Nondifferential and variational techniques in optimization, *Mathematical Programming Study*, Vol. 17, 77–90.
135. MORÉ, J. & SORENSEN, D.C. 1984 Newton's method, in *Studies in Numerical Analysis* (G.H. Golub, ed.), The Mathematical Association of America, Providence, RI., 29–82.
136. MORÉ, J.J. & THUENTE, D.J. 1994 Line search algorithms with guaranteed sufficient decrease, *ACM Transactions on Mathematical Software*, Vol. 20, 286–307.
137. MORÉ, J.J. & TORALDO, G. 1989 Algorithms for bound constrained quadratic programming problems, *Numer. Math.*, Vol. 55, 377–400.
138. MORÉ, J.J. & TORALDO, G. 1991 On the solution of large quadratic programming problems with bound constraints, *SIAM J. Optimization*, Vol. 1, 93–113.
139. NAZARETH, J.L. 1977 A relationship between BFGS and conjugate gradient algorithms, AMD Tech. Memo 282, Argonne National Laboratory.
140. NAZARETH, J.L. 1979 A relationship between BFGS and conjugate gradient algorithms and its implications for new algorithms, *SIAM J. Numer. Anal.*, Vol. 16, 794–800.
141. NAZARETH, J.L. 1986 Conjugate gradient methods less dependent on conjugacy, *SIAM Review*, Vol. 28, 501–511.
142. NAZARETH, J.L. 1986 The method of succesive affine reduction for nonlinear minimization, *Math. Program.*, Vol. 35, 97–109.
143. NAZARETH, J.L. 1999 Conjugate–gradient methods, *Encyclopedia of Optimization*, C. Floudas and P. Pardalos, eds., Kluwer Academic Publishers, Boston.
144. NOCEDAL, J. 1980 Updating quasi-Newton matrices with limited storage, *Mathematics of Computation*, Vol. 35, 773–782.
145. NOCEDAL, J. 1992 Theory of algorithms for unconstrained optimization, *Acta Numerica*, 199–241.

146. NOCEDAL, J. & WRIGHT, S.J. 1999 *Numerical optimization*, Springer–Verlag, New York.
147. OBERLIN, C. & WRIGHT, S.J. 2006 Active set identification in nonlinear programming, *SIAM J. Optimization*, Vol. 17, 577–605.
148. OREN, S.S. 1974 Self–scaling variable metric algorithm, Part II, *Management Sci.*, Vol. 20, 863–874.
149. OREN, S.S. 1982 Perspectives on self–scaling variable metric algorithms, *J. Optim. Theory and Applications*, Vol. 37, 137–147.
150. OREN, S.S. & LUENBERGER, D.G. 1974 Self–scaling variable metric algorithms, *Mathematical Program.*, Vol. 10, 70–90.
151. OREN, S.S. & LUENBERGER, D.G. 1974 Self–scaling variable metric (SVM) algorithms I: criteria and sufficient conditions for scaling a class of algorithms, *Management Sci.*, Vol. 20, 845–862.
152. OREN, S.S. & SPEDICATO, E. 1976 Optimal conditioning of self–scaling variable metric algorithm, Part II, *Management Sci.*, Vol. 20, 863–874.
153. ORTEGA, J.M. & RHEINBOLDT, W.C. 1970 *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, New York.
154. OSTROVSKI, A.M. 1970 *Solution of equations in Euclidean and Banach spaces*, Academic Press, New York, New York.
155. PAGUREK, B. & WOODSIDE, C.M. 1968 The conjugate gradient method for optimal control problems with bounded variables, *Automatica*, Vol. 4, 337–349.
156. PAIGE, C.C. & SAUNDERS, M.A. 1975 Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.*, Vol. 12, 617–629.
157. PEARSON, J.D. 1969 Variable metric methods of minimization, *Comp. J.*, Vol. 12, 171–178.
158. PERRY, A. 1976 A modified conjugate gradient algorithm , Discussion Paper No. 229, Center for Mathematical Studies in Economics and Management Science, Nortwestern University.
159. POLAK, E. 1971 *Computational methods in optimization: a unified approach*, Academic Press, New York.
160. POLAK, E., MAYNE, D.Q. & WARDI, Y. 1983 On the extension of constrained optimization algorithms from differentiable to nondifferentiable problems. *SIAM J. on Control Optimization*, Vol. 21, 179–203.
161. POLAK, E. & RIBIÈRE, G. 1969 Note sur la convergence de méthodes de directiones conjuguédes, *Rev. Française Informat. Recherche Opérationnelle*, Vol. 3, 35–43.
162. POLYAK, T. 1969 The conjugate gradient method in extremem problems, *USSR Comp. Math. Math. Phys.*, Vol. 9, 94–112.
163. POTRA, F.A. 1989 On *Q*–order and *R*–order of convergence, *J. Optim. Theory and Applications*, Vol. 63, 415–431.
164. POWELL, M.J.D. 1962 An iterative method for finding stationary values of a function of several variables. *Comp. J.*, Vol. 5, 147–151.
165. POWELL, M.J.D. 1964 An efficient method for finding the minimum of a function of several variables without calculating derivatives, *Comp. J.*, Vol. 7, 155–162.
166. POWELL, M.J.D. 1976 Some global convergence properties of a variable metric algorithm for minimization without exact line searches, in *Nonlinear Programming SIAM–AMS Proceedings, Vol. IX*, (R.W. Cottle and C.E. Lemke, eds), SIAM Publications, Philadelphia.
167. POWELL, M.J.D. 1976 Some convergence properties of the conjugate gradient method, *Math. Program.*, Vol. 11, 42–49.
168. POWELL, M.J.D. 1977 Restart procedures for the conjugate gradient method, *Math. Program.*, Vol. 12, 241–254.
169. POWELL, M.J.D. 1984 Nonconvex minimization calculations and the conjugate gradient method, *Numerical Analysis, Dundee, 1983*, Lecture Notes in Mathematics, Vol. 1066, Springer–Verlag, Berlin, 122–141.
170. POWELL, M.J.D. 1987 Updating conjugate directions by the BFGS formula, *Math. Program.*, Vol. 38, 29–46.
171. PSHENICHNYI B.N. & DANILIN, YU.M. 1978 *Numerical methods in extremal problems.* Mir Publishers, Moscow.

172. PYTLAK, R. 1986 Minimax optimal control problems with discrete time: solutions properties and algorithms, PhD thesis, Warsaw University of Technology (in Polish).
173. PYTLAK, R. 1989 Numerical experiment with new conjugate direction methods for non-differentiable optimization, in *Proceedings of the 28th IEEE CDC Conference*, 12–15 December, Tampa, Florida, 2457–2462.
174. PYTLAK, R. 1994 On the convergence of conjugate gradient algorithms, *IMA J. Numer. Anal.*, Vol. 14, 443–460.
175. PYTLAK, R. 1998 An efficient algorithm for large scale problems with simple bound on the variables, *SIAM J. on Optimization*, Vol. 8, 632–560.
176. PYTLAK, R. 1999 *Numerical procedures for optimal control problems with state constraints*, Lecture Notes in Mathematics 1707, Springer–Verlag, Heidelberg.
177. PYTLAK, R. 2002 Comments on: global convergence of the method of shortest residuals by Y. Dai and Y. Yuan, *Numer. Math.*, Vol. 91, 319–321.
178. PYTLAK, R. & MALINOWSKI, K. 1987 Numerical methods for minimax optimal control problems with discrete time, *Proceedings of IFIP Conference on Analysis and Optimization of Systems*, Lecture Notes on Control and Information Sciences, 1986, Vol. 83, eds. A. Bensoussan and J.L. Lions, Springer-Verlag, Berlin, 167–178.
179. PYTLAK, R. & TARNAWSKI, T. 2003 The method of shortest residuals for large scale nonlinear problems, *Proceedings of 2003 American Control Conference*, June 4–6, Denver, Colorado, USA, 4742–4747.
180. PYTLAK, R. & TARNAWSKI, T. 2004 Preconditioned Conjugate Gradient Algorithms for Nonconvex Problems, Research Report RR/ISI/WCY/WAT/01/2004, Military University of Technology, Warsaw, also in *Proceedings of the 43rd IEEE CDC*, December, Bahamas, 3191–3196.
181. PYTLAK, R. & TARNAWSKI, T. 2006 Preconditioned Conjugate Gradient Algorithms for Nonconvex Problems, *Pacific Journal of Optimization*, Vol. 2, 81–104.
182. PYTLAK, R. & TARNAWSKI, T. 2008 Preconditioned conjugate gradient algorithms for non-convex problems with box constraints, Research Report RR/ISI/WCY/WAT/04/06, Military University of Technology, version 10th of February 2008.
183. PYTLAK, R. & TARNAWSKI, T. 2007 On the method of shortest residuals for unconstrained optimization, *J. Optim. Theory and Applications*, Vol. 133, 99–110.
184. REID, J.K. 1971 On the method of conjugate gradients for the solution of large sparse systems of linear equations, in *Large sparse sets of linear equations*, Academic Press, New York, 231–254.
185. RITTER, K. 1980 On the rate of superlinear convergence of a class of variable metric methods, *Numer. Math.*, Vol. 35, 293–313.
186. ROBINSON, S.M. 1984 Local structure of feasible sets in nonlinear programming, *Mathematical Programming Study*, Vol. 22, 217–230.
187. ROCKAFELLAR, R.T. 1970 *Convex analysis*, Princeton University Press, Princeton, New Jersey.
188. ROSENBROCK, H.H. 1960 An automatic method for finding the greatest or least value of a function. *The Computer Journal*, Vol. 3, 175–184.
189. SAAD, Y. 1996 *Iterative methods for sparse linear systems*, PWS Publishing Company.
190. SHANNO, D.F. 1970 Conditioning of quasi-Newton methods for function minimization, *Math. Computation*, Vol. 24, 647–657.
191. SHANNO, D.F. 1978 Conjugate gradient methods with inexact searches, *Math. Oper. Res.*, Vol. 3, 244–256.
192. SHANNO, D.F. 1978 On the convergence of a new conjugate gradient algorithm, *SIAM J. on Numer. Anal.*, Vol. 15, 1247–1257.
193. SHANNO, D.F. & PHUA, K–H. 1978 Matrix conditioning and nonlinear optimization, *Math. Program.*, Vol. 14, 149–160.
194. SHANNO, D.F. & PHUA, K–H. 1980 Remark on algorithm 500: minimization of unconstrained multivariate functions, *ACM Trans. on Math. Software*, Vol. 6, 618–622.

195. SIEGEL, D. 1992 Implementing and modifying Broyden class updates for large scale optimization, Report DAMPT/1992/NA12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.
196. SIEGEL, D. 1992 The use of conjugate direction matrices in quasi-Newton methods for nonlinear optimization, PhD thesis, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.
197. SIEGEL, D. 1993 Modifying the BFGS update by a new column scaling technique, *Math. Program.*, Vol. 66, 45–78.
198. SINNOTT, J.F. & LUENBERGER, D.G. 1967 Solution of optimal control problems by the method of conjugate gradients, in *1967 Joint Automatic Control Conference, preprints of papers*, Lewis Winner, New York, 566–574.
199. SPEDICATO, E. 1978 Computational experience with quasi-Newton algorithms for minimization problems of moderately large size, in L.C.W. Dixon and G.P. Szego eds, *Towards Global Optimization 2*. North-Holland, Amsterdam, 209–219.
200. STIEFEL, E. 1952 Über einige methoden der relaxationsrechnung, *Zeitschrift für Angewandte Mathematik und Physik*, Vol. 3, 1–33.
201. STIEFEL, E. 1953 Some special methods of relaxation technique, in *Simultaneous linear equations and the determination of eigenvalues*, ed. L.J. Paige and O. Tausky, Applied Mathematics Series 29, National Bureau of Standards, U.S. Goverment Printing Office, Washington, D.C., 43–48.
202. STIEFEL, E. 1957 Recent developments in relaxation techniques, *Proceedings of the International Congress of Mathematicians 1954*, North–Holland, Amsterdam, 348–391.
203. STIEFEL, E. 1958 Kernel polynomials in linear algebra and their numerical applications, in *Further contributions to the solution of simultaneous linear equations and the determination of eigenvalues*, Applied Mathematics Series 49, National Bureau of Standards, U.S. Goverment Printing Office, Washington, D.C., 1–22.
204. STOER, J. & WITZGALL, G. 1970 *Convexity and optimization in finite dimensions*, Springer–Verlag, New York.
205. TOINT, PH.L. 1988 Global convergence of a class of trust region methods for nonconvex minimization in Hilbert space, *IMA J. Numer. Anal.*, Vol. 8, 231–252.
206. VOIGT, R.G. 1971 Orders of convergence for iterative procedures, *SIAM J. Numer. Anal.*, Vol. 8, 222–243.
207. WOLFE, P. 1969 Convergence conditions for ascent methods, *SIAM Review*, Vol. 11, 226–235.
208. WOLFE, P. 1969 Convergence conditions for ascent methods, II some corrections, *SIAM Review*, Vol. 13, 185–188.
209. WOLFE, P. 1975 A method of conjugate subgradients for minimizing nondifferentiable functions, in *Mathematical Programming Study*, Vol. 3, M.L. Balinski, P. Wolfe, eds., North–Holland, Amsterdam, 145–173.
210. WOLFE, P. 1976 Finding the nearest point in a polytope, *Math. Program.*, Vol. 11, 128–149.
211. ZANGWILL, W.I. 1967 Minimizing a function without calculating derivatives, *Computer J.*, Vol. 10, 293–296.
212. ZARANTONELLO, E.H. 1971 Projections on convex sets in Hilbert space and spectral theory, in *Contributions to Nonlinear Functional Analysis*, E.H. Zarantonello ed., Academic Press, New York.
213. ZHU, C. BYRD, R.H., LU, P. & NOCEDAL, J. 1994 L–BFGS–B—FORTRAN subroutines for large–scale bound constrained optimization, Research Report, Northwestern University, Department of Electrical Engineering and Computer Science.
214. ZOUTENDIJK, G. 1970 Nonlinear programming, computational methods, in *Integer and Nonlinear Programming*, (J. Abadie, ed.), North–Holland, Amsterdam, 37–86.

Index