Journal of Symbolic Computation 35 (2003) 465–485

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

A combined symbolic and numerical algorithm for the computation of zeros of orthogonal polynomials and special functions

Amparo Gil[a,c,*], Javier Segura[b,c,1]

[a] *Departamento de Matemáticas, Facultad Ciencias, Universidad Autónoma de Madrid, 28049-Madrid, Spain*
[b] *Departamento de Matemáticas, Universidad Carlos III de Madrid, 28911-Leganés, Madrid, Spain*
[c] *Universität GhK, FB 17 Mathematik-Informatik, 34132-Kassel, Germany*

Received 16 November 2001; accepted 16 April 2002

Abstract

A Maple algorithm for the computation of the zeros of orthogonal polynomials (OPs) and special functions (SFs) in a given interval $[x_1, x_2]$ is presented. The program combines symbolic and numerical calculations and it is based on fixed point iterations. The program uses as inputs the analytic expressions for the coefficients of the three-term recurrence relation and a difference-differential relation satisfied by the set of OPs or SFs. The performance of the method is illustrated with several examples: Hermite, Chebyshev, Legendre, Jacobi and Gegenbauer polynomials, Bessel, Coulomb and Conical functions.

Keywords: Zeros; Orthogonal polynomials; Special functions; Fixed point iterations

1. Introduction

The accurate and efficient computation of the zeros of orthogonal polynomials (OPs) is a relevant numerical issue; as is well known, the abscissas of Gaussian quadrature formulas are the roots of specific OPs. Also, the zeros of special functions (SFs) are important in a vast number of applications, in particular the zeros of Bessel functions.

The most general methods for the computation of the zeros of OPs are matrix methods, based on the Golub–Welsch (Golub and Welsch, 1969) algorithm which uses a result of Wilf (1962). Matrix methods are also available (Ikebe, 1975; Ikebe et al., 1991) for SFs which are minimal with respect to a 3-term recurrence relation. Methods based on

* Corresponding author.
E-mail addresses: amparo.gil@uam.es (A. Gil), jsegura@math.uc3m.es (J. Segura).
[1] Present address: Departamento de Matematicas, Estadistica y Computacion, Universidad de Cantabria, 39005 Santander, Spain.

0014-5793/03/$ - see front matter
doi:10.1016/S0747-7171(03)00013-0

specific approximations to the roots (mainly asymptotic expansions) are also available; for example, in Temme (1979) asymptotic approximations to the zeros of first and second kind Bessel functions, which are refined by a Newton method, are considered.

In this paper we present a Maple program for the evaluation of the zeros of OPs and SFs based on a globally convergent fixed point method. The program is available upon request from the authors.

The method applies for SFs and OPs, y_n, which are solutions of linear homogeneous second order ordinary differential equations (ODEs) and satisfy difference-differential equations (DDEs) of the type:

$$\begin{aligned} y_n'(x) &= a_n(x)y_n(x) + d_n(x)y_{n-1}(x) \\ y_{n-1}'(x) &= b_n(x)y_{n-1}(x) + e_n(x)y_n(x) \end{aligned}$$

where the coefficients $a_n(x)$, $b_n(x)$, $d_n(x)$ and $e_n(x)$ are continuous.

This method proves to be efficient and more general than matrix methods. Fixed point methods apply for arbitrary solutions of second order linear homogeneous ODEs and not only to polynomial solutions or minimal solutions of a three-term recurrence relation (TTRR) (which is, for instance, the case for regular Bessel and Coulomb functions). A further advantage of the fixed point method is that the interval for finding zeros can be arbitrarily chosen, contrary to matrix methods. Furthermore, the symbolic part of the algorithm provides analytical information concerning bounds on the distance between adjacent zeros.

2. Theory

The method for the computation of zeros in an interval I of solutions of second order ODEs

$$y_n'' + B_n(x)y_n' + A_n(x)y_n = 0 \tag{1}$$

with independent solutions $\{y_n^{(1)}, y_n^{(2)}\}$ is based on the existence of a contrast differential equation

$$y_{n-1}'' + B_{n-1}(x)y_{n-1}' + A_{n-1}(x)y_{n-1} = 0, \tag{2}$$

satisfied by independent solutions $\{y_{n-1}^{(1)}, y_{n-1}^{(2)}\}$ such that there exist first order DDEs:

$$\begin{aligned} y_n'(x) &= a_n(x)y_n(x) + d_n(x)y_{n-1}(x) \\ y_{n-1}'(x) &= b_n(x)y_{n-1}(x) + e_n(x)y_n(x), \end{aligned} \tag{3}$$

with continuous coefficients in I, satisfied both by $Y^{(1)} \equiv \{y_n^{(1)}, y_{n-1}^{(1)}\}$ and $Y^{(2)} \equiv \{y_n^{(2)}, y_{n-1}^{(2)}\}$. Such DDEs, satisfied simultaneously by two sets $Y^{(1)}$, $Y^{(2)}$, with $\{y_n^{(1)}, y_n^{(2)}\}$ and $\{y_{n-1}^{(1)}, y_{n-1}^{(2)}\}$ independent solutions of two different ODEs, will be called general. The indices n and $n-1$ will normally denote a parameter of the differential equations, but not necessarily: n and $n-1$ can be interpreted as labels referring to two different ODEs.

It can be shown that such general DDEs exist and are unique for a pair $\{Y^{(1)}, Y^{(2)}\}$ as described above. The restriction imposed on the contrast functions y_{n-1} is the continuity of the coefficients of the DDEs.

2.1. Properties of the solutions and the coefficients

Properties of the coefficients of the DDEs can be extracted using well-known properties of the solutions of ODEs. First, d_n and e_n can never cancel because $\{y_k^{(1)}, y_k^{(2)}\}$ ($k = n, n-1$) are independent solutions (see Segura, 2002, Lemma 2.1). Indeed, writing the first equation in (3) for $Y^{(1)}$ and $Y^{(2)}$:

$$y_n^{(1)'} = a_n(x)y_n^{(1)} + d_n(x)y_{n-1}^{(1)}$$
$$y_n^{(2)'} = a_n(x)y_n^{(2)} + d_n(x)y_{n-1}^{(2)}$$

and therefore $d_n(x) = W[y_n^{(1)}, y_n^{(2)}]/Z_n(x)$, with

$$Z_n(x) = \begin{vmatrix} y_n^{(1)} & y_{n-1}^{(1)} \\ y_n^{(2)} & y_{n-1}^{(2)} \end{vmatrix} = \det[Y^1, Y^2]. \tag{4}$$

Thus $d_n(x) \neq 0\,\forall x$ because the Wronskian of two independent solutions of a second order ODE never cancels. Similarly $e_n(x) = -W[y_{n-1}^{(1)}, y_{n-1}^{(2)}]/Z_n(x)$. These arguments, together with the fact that $Z_n(x)$ cannot be identically zero (Marx, 1953, Theorem 1), show that the DDEs (3) exist and are unique (Marx, 1953).

We also see that the continuity (and differentiability) of the coefficients is equivalent to the condition $Z_n(x) \neq 0\ \forall x \in I$.

On the other hand, if the coefficients are continuous then necessarily the zeros of the solutions y_n and y_{n-1} are interlaced, and if one (non-trivial) solution of the DDEs has two or more zeros in I then necessarily $e_n d_n < 0$ (Segura, 2002, Lemma 2.4).

By differentiating the first DDE and using both DDEs to eliminate y_{n-1} and y'_{n-1}, we arrive at a second order ODE for y_n, satisfied by two independent solutions $y_n^{(1)}$, $y_n^{(2)}$. This gives the relations:

$$B_n = -a_n - b_n - \frac{d_n'}{d_n}; \qquad A_n = -a_n' - d_n e_n + a_n \frac{d_n'}{d_n} + a_n b_n \tag{5}$$

and similarly

$$B_{n-1} = -a_n - b_n - \frac{e_n'}{e_n}; \qquad A_{n-1} = -b_n' - d_n e_n + b_n \frac{e_n'}{e_n} + a_n b_n. \tag{6}$$

In Segura (2002), the case $B_n = B_{n-1} = B(x)$ was considered in detail. This is not a restriction, given that one can always consider a change of the dependent variables $y_k = \exp(-\frac{1}{2}\int B_k\,\mathrm{d}x)\tilde{y}_k$, $k = n, n-1$; the functions $\tilde{y}_k$, with the same zeros as y_k, satisfy ODEs without the first derivative term (ODEs in normal form).

Considering the first equations in (5) and (6) we see that the condition $B_n = B_{n-1}$ is equivalent to $d_n/e_n =$ constant. Therefore, an alternative way to consider the general case in which $B_n \neq B_{n-1}$ is to renormalize the solutions $y_k(x) = \nu_k(x)\tilde{y}_k(x)$ in such a way that the functions $\tilde{y}_k$ satisfy the DDEs (3) with $d_n/e_n =$ constant.

2.2. Transformation of the DDEs

Let us temporarily change ournotation for the DDEs, by denoting by y our "problem functions" and by w the "contrast functions":

$$\begin{aligned} y'(x) &= \alpha(x)y(x) + \delta(x)w(x) \\ w'(x) &= \beta(x)w(x) + \gamma(x)y(x). \end{aligned} \tag{7}$$

Next we describe the analytical transformations that are required to build globally convergent fixed point iterations (FPIs) for the computation of the zeros of the solutions y. We assume that the coefficients are differentiable, that δ and γ never cancel (Segura, 2002, Lemma 2.1) and that $\delta\gamma < 0$ (Segura, 2002, Lemma 2.4). As we have discussed, these are general properties for general DDEs having solutions with at least two zeros.

First, the functions are renormalized:

$$y(x) = \lambda_y(x)\bar{y}(x), \qquad w(x) = \lambda_w(x)\bar{w}(x) \tag{8}$$

with $\lambda_y(x),\ \lambda_w(x) \neq 0\ \forall x \in I$; then the DDEs for the renormalized functions are

$$\begin{aligned} \bar{y}' &= \bar{\alpha}\bar{y} + \bar{\delta}\bar{w} \\ \bar{w}' &= \bar{\beta}\bar{w} + \bar{\gamma}\bar{y} \end{aligned}$$

with

$$\bar{\alpha} = \left(\alpha - \frac{\lambda_y'}{\lambda_y}\right), \qquad \bar{\delta} = \delta\frac{\lambda_w}{\lambda_y}, \qquad \bar{\beta} = \left(\beta - \frac{\lambda_w'}{\lambda_w}\right), \qquad \bar{\gamma} = \gamma\frac{\lambda_y}{\lambda_w}. \tag{9}$$

The renormalizing functions are chosen in such a way that

$$\bar{\delta} = -\bar{\gamma}, \qquad \bar{\delta} > 0$$

that is

$$\lambda_y = \operatorname{sign}(\delta)\sqrt{-\frac{\delta}{\gamma}}\lambda_w.$$

With this selection and the change of variables $z(x) = \int \bar{\delta}(x)\,\mathrm{d}x$, we obtain

$$\begin{aligned} \dot{\bar{y}} &= \frac{\bar{\alpha}}{\bar{\delta}}\bar{y} + \bar{w} \\ \dot{\bar{w}} &= \frac{\bar{\beta}}{\bar{\delta}}\bar{w} - \bar{y} \end{aligned}$$

where dots denote derivation with respect to z.

Then, the ratio $H(z) = \bar{y}/\bar{w}$, with zeros and singularities interlaced (coinciding, up to a change of variables, with the zeros of y and w), satisfies:

$$\dot{H} = 1 + H^2 - 2\eta H \tag{10}$$

with

$$\eta = (\bar{\beta} - \bar{\alpha})/(2\bar{\delta}). \tag{11}$$

2.3. Global fixed point iterations

From Eq. (10), it can be shown (Segura, 2002) that the iteration

$$T(z) = z - \arctan(H(z)) \tag{12}$$

converges globally to the zeros of $y(x(z))$ in intervals where η does not change sign.

It is easy to relate the functions $H(z)$, $\eta(z)$ with the original functions y, w and the coefficients of the DDEs:

$$\begin{aligned} H(z) &= \operatorname{sign}(\delta)\sqrt{-\frac{\gamma}{\delta}}\frac{y(x(z))}{w(x(z))}, \\ z(x) &= \int \sqrt{-\gamma\delta}, \\ \eta(x) &= \frac{1}{2\sqrt{-\delta\gamma}}\left[\beta - \alpha + \frac{1}{2}\left(\frac{\delta'}{\delta} - \frac{\gamma'}{\gamma}\right)\right], \end{aligned} \tag{13}$$

where $\eta(z) = \eta(x(z))$.

Putting all these expressions together, the global iteration reads:

$$T(z) = z - \operatorname{sign}(\delta)\arctan\left(\sqrt{-\frac{\gamma}{\delta}}\frac{y(x(z))}{w(x(z))}\right). \tag{14}$$

As mentioned, $T(z)$ is a globally convergent iteration in intervals where η does not change sign. Let us denote the successive zeros of $y(x)$ and $w(x)$ by x_y^m, x_w^m; similarly, we denote the zeros of $y(x(z))$ and $w(x(z))$ by z_y^m and z_w^m, respectively. The zeros of $y(x(z))$ and $w(x(z))$ are interlaced; the indices m enumerate the zeros from the smallest to the largest ones in the interval I and they are chosen in such a way that:

$$\cdots < z_y^{m-1} < z_w^m < z_y^m < z_w^{m+1} < z_y^{m+1} < \cdots.$$

Given a value x_0 between two consecutive zeros of w, x_w^m and x_w^{m+1}, and taking as starting value $z_0 = z(x_0)$ we have Segura (2002)

$$\lim_{p\to\infty} T^{(p)}(z_0) = z_y^m,$$

where $x_y^m = x(z_y^m)$ is the zero of y_n between such two consecutive zeros of w.

The iteration $T(z)$ is quadratically convergent because $\dot{T}(z_y^m) = 0$ for any zero z_y^m of $y(x(z))$.

Observe that the application of the FPI requires that $z(x)$ is an invertible function (that is, that $z(x)$ is a change of variables). This is so, because the integrand is positive (Eq. (13)). Of course, the algorithm will be more efficient if the inversion of $z(x)$ can be obtained analytically; this is the case for all OPs and SFs we have considered so far.

2.4. Forward and backward iterative schemes

Next we describe the iterative scheme to compute all the zeros inside an interval where η does not change sign.

Let us, for instance, assume that $\eta < 0$ in a given interval. It can be shown (Segura, 2002) that $\pi/2 < z_y^m - z_w^m$ and $0 < z_w^{m+1} - z_y^m < \pi/2$ (and therefore

$z_y^{m+1} - z_y^m > \pi/2$) for all m such that the zeros are inside this interval. Hence, given that $z_w^{m+1} < z_y^m + \pi/2 < z_y^{m+1}$, the larger zeros subsequent to z_y^m can be computed iteratively: $z_y^{m+1} = \lim_{p\to\infty} T(z_y^m + \pi/2)$ (forward sweep). In contrast, when $\eta > 0$, a backward sweep is the way to find the zeros. If η changes sign, a combination of forward and backward sweeps is possible (Segura, 2002). As in Segura (2002), we will call expansive sweep the iterative computation of zeros starting from a zero z_y^m and evaluating the zeros smaller than z_y^m by a backward sweep and the zeros greater than z_y^m by a forward sweep; reciprocally, interchanging forward by backward sweep, we have a contractive sweep.

The current version of the algorithm assumes that only one change of sign in η can take place (which is a considerably general condition, see Segura, 2002). More general situations can be described; however, for the purpose of computing zeros of OPs and SFs, this is a general enough situation.

2.5. *The case of uniparametric families of functions*

A further restriction of the algorithm is that it has been implemented for the computation of zeros of families of functions depending on one parameter and functions which can be related to this kind of function (like Airy functions, which can be related to Bessel functions of order 1/3 (Segura, 1998)). This is not an intrinsic limitation of the method, as we previously discussed.

We consider uniparametric families of ODEs $y_k'' + B_k(x)y_k' + A_k(x)y_k = 0$, where k is the parameter, with independent solutions $\{y_k^{(1)}\}$, $\{y_k^{(2)}\}$ verifying

$$\begin{aligned} y_k' &= a_k y_k + d_k y_{k-1} \\ y_{k-1}' &= b_k y_{k-1} + e_k y_k \end{aligned}$$

with continuous coefficients.

The functions may depend on additional parameters, but only one parameter determines which is the contrast function appearing together with the problem function y_n in the DDEs.

If allowed by the range of k we can write two systems of DDEs for each problem function y_n, taking $k = n$ in:

$$\begin{cases} y_k' = a_k y_k + d_k y_{k-1} \\ y_{k-1}' = b_k y_{k-1} + e_k y_k; \end{cases} \qquad \begin{cases} y_k' = b_{k+1} y_k + e_{k+1} y_{k+1} \\ y_{k+1}' = a_{k+1} y_{k+1} + d_{k+1} y_k. \end{cases} \tag{15}$$

This means that a TTRR is verified:

$$y_{k+1} = r_k y_k + s_k y_{k-1}, \tag{16}$$

where

$$r_k = \frac{a_k - b_{k+1}}{e_{k+1}} \qquad \text{and} \qquad s_k = d_k/e_{k+1} \neq 0 \quad \forall x \in I. \tag{17}$$

If the coefficients of the DDEs are continuous then $\{y_k^{(1)}\}$, $\{y_k^{(2)}\}$ are necessarily independent solutions of the TTRR because $Z_k(x) \neq 0\, \forall x,\ k$ (Eq. (4)). TTRRs are a useful tool to compute SFs and OPs.

Another interesting fact is that we have two alternative FPIs by choosing as contrast functions $w = y_{n-1}$ or $w = y_{n+1}$. In one case we take (Eqs. (13) and (15)):

$$y = y_n, \qquad w = y_{n-1}, \qquad \alpha = a_n, \qquad \beta = b_n \qquad \delta = d_n, \qquad \gamma = e_n \tag{18}$$

and for the other one

$$\begin{array}{llll} y = y_n, & w = y_{n+1}, & \alpha = b_{n+1}, & \beta = a_{n+1} \\ \delta = e_{n+1}, & \gamma = d_{n+1}. & & \end{array} \tag{19}$$

More explicitly, denoting, as in Segura (2002):

$$n_i = \begin{cases} n+1, & i = +1 \\ n, & i = -1, \end{cases}$$

two different FPIs $T_i(z) = z - \arctan(H_i(z))$ can be considered to compute the zeros of y_n, where

$$\begin{aligned} H_i &= \frac{\bar{y}_n(z)}{\bar{y}_{n+i}(z)} = i\ \mathrm{sign}(d_{n_i}) K_i \frac{y_n(x(z_{n_i}))}{y_{n+i}(x(z_{n_i}))}, \\ K_i &= \left(-\frac{d_{n_i}}{e_{n_i}}\right)^{i/2}, \\ z_i &= \int \sqrt{-d_{n_i} e_{n_i}}\, \mathrm{d}x, \\ \eta_i &= i \frac{1}{2\sqrt{-d_{n_i} e_{n_i}}} \left(a_{n_i} - b_{n_i} + \frac{1}{2}\left(\frac{e'_{n_i}}{e_{n_i}} - \frac{d'_{n_i}}{d_{n_i}} \right) \right). \end{aligned} \tag{20}$$

2.6. *Improvement of the iteration step*

From Eq. (20) one can expect that, generally speaking, when $\eta_{+1} > 0$ then $\eta_{-1} < 0$ and vice-versa. This means that both forward and backward sweeps (or expansive and contractive) are generally available to compute the zeros of a function y_n.

The symbolic algorithm selects the most appropriate iteration depending on the monotonicity properties for the second order ODE (in normal form)

$$\ddot{\tilde{y}}(z) + \tilde{A}(z)\tilde{y}(z) = 0, \qquad \tilde{A}(z) = 1 - \dot{\eta} + \eta^2,$$

satisfied by

$$\tilde{y} = \exp\left(-\tfrac{1}{2} \int (\tilde{\alpha} + \tilde{\beta})\, \mathrm{d}z \right) \bar{y},$$

where $\tilde{\alpha} = \bar{\alpha}/\bar{\delta}$, $\tilde{\beta} = \bar{\beta}/\bar{\delta}$ (see Section 2.2). Given the continuity of the coefficients of the DDEs, the functions $\tilde{y}_n(z)$ have the same zeros as $\bar{y}$.

If, for instance, $\dot{\tilde{A}}(z) < 0$ in a given interval, then the spacing between the zeros of $\tilde{y}(z)$, and hence of $y(x(z))$, increases for increasing z, that is:

$$z_y^m - z_y^{m-1} < z_y^{m+1} - z_y^m$$

and if an iteration with $\eta < 0$ has been chosen: $z_y^m - z_w^m > \pi/2$, $z_w^m - z_y^{m-1} < \pi/2$; then it is not difficult to see that

$$z_w^{m+1} < z_y^m + (z_y^m - z_y^{m-1}) < z_y^{m+1} < z_w^{m+2}.$$

This means that the starting value $z_y^m + \Delta z^m$, with $\Delta z^m = (z_y^m - z_y^{m-1})$, provides convergence to z_y^{m+1} (which is the zero between z_w^{m+1} and z_w^{m+2}) and that the step Δz^m improves the default step $\pi/2$ because $\Delta z^m > \pi/2$; therefore it provides a closer (under)estimation of z_y^{m+1}.

This suggests that using the difference of two previously evaluated zeros as iteration steps instead of $\pm\pi/2$ will improve the performance if an iteration ($i = +1$ or -1) can be chosen such that $\dot{\tilde{A}}(z)\eta > 0$.

For instance, for the case of OPs we observe that $\tilde{A}(z)$ has a maximum and then we should choose an expansive iteration.

When possible, the algorithm will consider this improvement of the iteration step.

3. Description of the program

As previously described, the algorithm is based on the existence of two linear first order DDEs. However, we choose as symbolic input the coefficients of a TTRR and of one of the DDEs. Of course, both choices are equivalent (Eqs. (16) and (17)). The reason for giving the TTRR and one DDE is that this option is more often available in books (see Abramowitz and Stegun, 1972).

The algorithm has several modes of operation. The user can select one of the already implemented cases for the computation of zeros or introduce other recurrence relations. The functions which are implemented in the present version of the program are: general Bessel functions $\cos\alpha J_\nu(x) - \sin\alpha Y_\nu(x)$ (we denote $n = \nu$), regular Coulomb wave functions $F_L(\eta, \rho)$ (notation: $n = L$, $\gamma = \eta$, $x = \rho$), first kind Conical functions $P^n_{-1/2+i\tau}(x)$ ($x > 1$), where i is the imaginary unit, Hermite polynomials $H_n(x)$, Legendre polynomials $P_n(x)$, generalized Laguerre polynomials $L_n^{(\alpha)}(x)$, Gegenbauer polynomials $C_n^{(\alpha)}(x)$ and Jacobi polynomials $P_n^{(\alpha,\beta)}(x)$.

If one of these functions is considered, we do not need to introduce the coefficients of a TTRR and a DDE.

The algorithm produces both symbolic and numerical results. The symbolic results can be useful to implement algorithms for the computation of SFs in compilable languages like Fortran. This is relevant when the speed of the algorithms is of concern or when the zeros for different sets of parameters of a same function are needed; a Fortran template will also be made available in the near future. Also, as described in Segura (2003), this information can be used to study interlacing properties of the zeros.

The numerical results are the zeros of the function under consideration in the selected interval. One can select performing only the symbolic task or both the symbolic and numerical tasks.

3.1. Inputs

We are prompted by the program to provide the following inputs:

1. Number of digits for the calculation.
2. Selection of one of the implemented functions or introduction of coefficients.
3. If we choose to introduce a function not yet implemented, the program asks for:
   (a) Interval of definition of the function.
   (b) Number of parameters of the function (the first to be introduced is the parameter n that is incremented in the TTRR and the DDE).
   (c) Name of the parameters and permissible range of the parameters.
   (d) Explicit expressions of the coefficients in the TTRR $y_{n+1} = r_n y_n + s_n y_{n-1}$ and the DDE $y'_n = a_n y_n + d_n y_{n-1}$. n is one of the parameters that have been defined before.
4. Values of the parameters: We must fix one particular set of values for the parameters. Even if only the symbolic computation is required we must give a set of representative values. The reason for this is that the iteration to be chosen ($i = +1$ or -1) may depend on the actual values considered and even if this is not so, the discussion of the monotonicity properties of $\tilde{A}(z)$ when free parameters are present becomes a difficult task.
5. Selection of strictly symbolic or symbolic/numerical computations.
6. If the symbolic/numerical option is chosen:
   (a) State the interval (contained in the interval of definition of the functions) where the zeros are sought.
   (b) Choose the method of computation of the ratios $y_n/y_{n\pm1}$. Three possibilities are given:
      (ii.a) Finite continued fraction: This method applies for dominant solutions of the corresponding TTRR and it corresponds to the forward evaluation of the recurrence relation. Also, when the TTRR has no dominant solutions, this option can be generally used when a moderate number of iterations are needed. In this case, the ratio of starting values y_0/y_1 must be given from which, by forward recursion, the values $y_n/y_{n\pm1}$ can be computed. This method is applied for the OPs already implemented in the algorithm.
      (ii.b) Infinite continued fraction: This can be the method of choice when minimal solutions of the corresponding TTRR are considered. This is the case, for instance, for the (already implemented) functions: Bessel function $J_n(x)$, regular Coulomb wavefunction and Conical function $P^n_{-1/2+i\tau}$ (i is the imaginary unit). The continued fraction is computed using the modified Lenz–Thompson algorithm (Press et al., 1992).
      (ii.c) Direct computation: When the functions under consideration are implemented in Maple, Maple can directly compute the ratio y_n/y_{n+1}. The program then asks us to provide the function y_n in the form of the corresponding Maple command; if the function y_n depends on additional

parameters in addition to n, the values of these extra parameters must be fixed when the function is provided. The method of direct computation can be applied to all the already implemented functions in the algorithm except for the Coulomb functions and Conical functions (not yet implemented in Maple). However, the algorithms work more efficiently if one of the two previous approaches can be used. For the already implemented functions, the option of direct computation is considered for Bessel functions $\mathcal{C}_\nu(x) = \cos\alpha J_\nu(x) - \sin\alpha Y_\nu(x)$ when $\cos\alpha \neq 0$.

(c) Relative precision required, which should never be smaller than $10^{-\text{Digits}}$ (the number of digits is the first input described above).
(d) Maximum number of iterations allowed.

In the Appendix, we provide an example of a session with the algorithm *zeros.mpl*.

3.2. Basic tasks

The program *zeros.mpl* performs the following tasks:

- Calculation of the coefficients b_n and e_n of the difference relation:

  $$y'_{n-1} = b_n y_{n-1} + e_n y_n$$

  using the coefficients of the TTRR $y_{n+1} = r_n y_n + s_n y_{n-1}$ and the differential relation $y'_n = a_n y_n + d_n y_{n-1}$ (Eq. (17)).
- Check the continuity of the coefficients of the DDEs and the inequality $e_n d_n < 0$. Check also the consistency of Eqs. (5) and (6); A_n and A_{n-1} are computed and it is verified that they are identical with a shift in the parameter n; the same is done for B_n. In case the checks fail, an error message appears and the code stops its execution.
- Normalization of the solutions; calculation of: change of variable $z(x)$, η_i of the normalization K_i and $\tilde{A}(z)$, as described in Section 2.5 (initially assuming that the iteration is $i = -1$).
- Computation of the extrema of $\tilde{A}(z)$ and selection of the appropriated iteration ($i = +1, -1$) depending on the monotonicity properties of $\tilde{A}(z)$ (Section 2.6).
- Re-evaluation (if the selected iteration turns out to be $i = +1$) of the basic functions of the algorithm (change of variable, η_i, etc).
- Selection and initialization of the sweep (forward, backward, contractive or expansive) depending on the location of the value x_η (the zero of $\eta_i(x)$) if it exists and the sign of η_i. For instance, if η_i is negative in the interval under consideration a forward sweep is the option.

3.3. Output

The numerical outputs of the code are the zeros computed in the interval and the number of iterations applied to compute each zero.

The symbolic output is displayed before entering the numerical procedures and it provides the information needed to implement the fixed point method for the problem function, namely:

1. The change of variables $z(x)$ applied.
2. The coefficient $\tilde{A}(z)$ for the second order ODE in normal form in the z variable (for brevity, it is expressed in terms of x, which is a function of z).
3. The parameter $\eta_i(x(z))$, also expressed in terms of x.
4. The normalization factor K_i.
5. Sign of the d_{n_i} coefficient of the DDE.
6. Type of FPI (index $i = 1$ or -1).

Outputs 1, 4, 5 and 6 are all that is needed to implement the numerical method.

The type of sweep (forward/backward, contractive/expansive) depends, as discussed in more detail in Segura (2002), on the sign of η_i.

4. Examples of symbolic output

Table 1 summarizes the symbolic outputs for the implemented cases. For brevity, $\tilde{A}(z)$ is not shown, where $M_{n,\lambda} = \sqrt{(n+1)(n+1+\lambda)}$, $N_{n,\lambda} = \sqrt{(n+1+\lambda)/(n+1)}$, $R_{n,\lambda} = 2(n+1)+\lambda$ and $\gamma_{n,\tau} = ((n-1/2)^2+\tau^2)^{-1/2}$. Of course the same results hold for a general solution of the system of DDEs. For instance, the same results apply to general Bessel functions $\cos\alpha J_n(x) - \sin\alpha Y_n(x)$, for combinations of the regular and irregular Coulomb functions, and so on.

Table 1
Examples of symbolic outputs provided by the code

$y_n(x)$	$z(x)$	$\eta(x)$	K_i	sign(d)	i
$H_n(x)$ (Hermite)	$\sqrt{2(n+1)}x$	$-\frac{x}{2}\sqrt{\frac{2}{n+1}}$	$\sqrt{2n+2}$	+	+1
$P_n(x)$ (Legendre)	$(n+1)\tanh^{-1}x$	$-x$	1	+	+1
$L_n^{(\alpha)}(x)$ (Laguerre)	$M_{n,\alpha}\log(x)$	$\frac{2n+2+\alpha-x}{2\sqrt{(n+1+\alpha)(n+1)}}$	$N_{n,\alpha}$	−	+1
$C_n^{(\alpha)}(x)$ (Gegenbauer)	$\sqrt{(n+2\alpha)(n+1)}\tanh^{-1}x$	$-\frac{x(2n+1+\alpha)}{\sqrt{(n+2\alpha)(n+1)}}$	$\sqrt{\frac{n+2\alpha}{n+1}}$	+	+1
$P_n^{(\alpha,\beta)}(x)$ (Jacobi)	$\frac{2M_{n,\alpha}M_{n+\beta,\alpha}}{R_{n,\alpha+\beta}}\tanh^{-1}x$	$-\frac{x(R_{n,\alpha+\beta})^2-\beta^2+\alpha^2}{4M_{n,\alpha}M_{n+\beta,\alpha}}$	$\frac{N_{n,\alpha}}{N_{n+\beta,\alpha}}$	+	+1
$P^n_{-1/2+i\tau}(x)$ (Conical)	$\frac{1}{\gamma_{n,\tau}}\cosh^{-1}x$	$\frac{(n-1/2)x}{\sqrt{x^2-1}}\gamma_{n,\tau}$	$\gamma_{n,\tau}$	−	−1
$J_n(x)$ (Bessel)	x	$\frac{(2n-1)}{2x}$	1	+	−1
$F_n(\gamma,x)$ (Coulomb)	$\frac{x\sqrt{n^2+\gamma^2}}{n}$	$\frac{n^2+\gamma x}{x\sqrt{n^2+\gamma^2}}$	1	+	−1

All these cases have in common that the normalization factor K_i does not depend on x. As discussed, this reflects the fact that the coefficient $B_n(x)$ of the ODE does not depend on n and, consequently, the d_n and e_n coefficients are such that d_n/e_n is a constant. However, the algorithm is able to deal with the more general situation in which $B_n(x)$ depends on n.

For example, the Kummer function $M(a, c; x)$ satisfies the differential equation

$$xy'' + (c - x)y' - ay = 0$$

and therefore the differential equations for the family of functions $y_n(x) = M(a + n, b + n; x)$ have coefficients $B_n(x)$ depending on n. In this case, the program gives the following symbolic output:

$$z(x) = 2\sqrt{x(1 - n - a)}; \qquad \eta(x) = -\frac{1}{4}\frac{3 - 2b - 2n + 2x}{\sqrt{x(1 - n - a)}}$$
$$K_i = \frac{\sqrt{x(1 - n - a)}}{|b + n - 1|}; \qquad \mathrm{sign}(d) = -1; \qquad i = -1$$

and we see that, as expected, K_i depends on x.

The type of sweep for each family of functions is easily read from the obtained expression of η_i, where i has been chosen, as described, to improve the iteration step by using the monotonicity properties of the function $\tilde{A}(z)$. For instance, for OPs there is a transition point (x_η for which $\eta_i(x_\eta) = 0$) inside their support and η_i is positive on the right of the interval. This means that expansive sweeps are the option, which is coherent with the fact that $\tilde{A}(z)$ has a maximum inside the interval of definition. For Bessel ($x > 0$) and Conical functions ($x > 1$), a forward (backward) sweep would be considered for $n < 1/2$ ($n > 1/2$). On the other hand, for Coulomb functions ($x > 0$) there is the possibility of backward or expansive sweeps depending on the values of the parameters. In all cases described with a transition point (x_η), the expansive sweep will be replaced by a forward or backward sweep for certain selections of the interval; for instance, if we choose an interval $[x_1, x_2]$ such that $x_1 > x_\eta$ with x_η the transition point, then the expansive sweep is replaced by a forward sweep.

It is interesting to note that the change of variables for OPs maps the support of the polynomials to $(-\infty, +\infty)$. This means that the change of variables becomes singular at the ends of the support, which go to infinity. Therefore, it is expected that the zeros approaching the ends of the support interval are computationally the most demanding. This effect becomes more apparent as the order increases and the rest of the parameters approach singular values (for example, $\alpha \to -1^+$ for Laguerre, $\alpha \to -1/2^+$ for Gegenbauer and $\alpha \to -1^+$, $\beta \to -1^+$ for Jacobi). To deal with the computation of the extreme zeros of OPs for extreme values of the parameters, it is convenient to fix the maximum number of iterations to higher values than the recommended 50 iterations.

On the other hand, the difference equations over the order n are not necessarily the most efficient option for the computation of zeros of OPs. For instance, by taking into account the relation of Laguerre polynomials with confluent hypergeometric functions:

$$\frac{n!}{(n + \alpha)_n} L_n^\alpha = M(-n, \alpha + 1, x)$$

the problem can be translated into finding the zeros of $M(a, b, x)$ for the particular values of the parameters corresponding to a Laguerre polynomial. We have many possibilities of defining our sequences of functions y_n, an option is $y_n(x) = M(a + n, b + n, x)$. It turns out, as we have already shown, that the associated change of variables for this option is not singular as $x \to 0^+$ and that it behaves like $z(x) \sim \sqrt{x}$. This selection behaves much better than the already implemented version for the smallest zero. In a future publication (Gil et al., in preparation), we will study in detail the use of different sequences of functions to compute zeros of hypergeometric and confluent hypergeometric functions depending on the values of the parameters and the range of the zeros.

5. Performance of the code

Let us discuss the performance of the code for the group of polynomials and SFs already implemented in the algorithm.

The main advantage of Maple algorithms compared to fixed precision programming languages is that the number of digits in the computation can be chosen at will. This feature can be used to check the accuracy and efficiency of our algorithm. We have computed the zeros of OPs with 10^{-100} relative precision and checked their accuracy by computing the polynomials at the zeros, $p(x_z)$, and comparing with the estimated value $p(x_z(1 + \epsilon)) \simeq p(x_z) + \epsilon x_z p'(x_z) = \epsilon x_z p'(x_z)$ (where ϵ is the relative error in the computation of the zero x_z). Typically, the algorithm needs around 10 iterations on average for each zero, although slower convergence is observed in the extreme cases described in the previous section. We obtain full agreement within the precision considered in all cases when enough digits (120–130) are considered for orders up to 100. For even higher orders and higher precision we expect that the algorithm will also work accurately. The same check was considered for Bessel functions. Conical functions and Coulomb wavefunctions are not yet implemented in Maple and this direct check cannot be performed.

Next, we will show the number of required iterations for the computation of the zeros for a certain selection of parameters. In all cases, the precision demanded for the calculation of the zeros is 10^{-12}; however, the convergence is so fast that the number of iterations required increases by few units when much higher precisions are required (for instance 1–2 extra iterations are required for 10^{-40} relative precision). By using asymptotic expansions as initial guesses for the zeros, the performance could be improved in some cases. However, the algorithm is intended to work under very general circumstances and for arbitrary solutions of the corresponding ODEs while asymptotic approximations are specific for particular solutions. For a fast algorithm for the zeros of first and second kind Bessel functions based on asymptotic expansions see Temme (1979). As we will next show, our algorithm, being quite general, is also very efficient (see, for instance, Fig. 6).

Figs. 1–8 show the number of iterations needed in order to compute the zeros of Hermite (Fig. 1), Legendre (Fig. 2), generalized Laguerre (Fig. 3), Gegenbauer (Fig. 4) and Jacobi (Fig. 5) polynomials, for selected values of the parameters, as a function of the position of the zeros. Figs. 6–8 correspond to Bessel ($J_\nu(x)$), Conical ($P^n_{-1/2+i\tau}(x)$) and Coulomb wavefunctions ($F_n(\eta, x)$) respectively. All figures correspond to 10^{-12} relative precision.

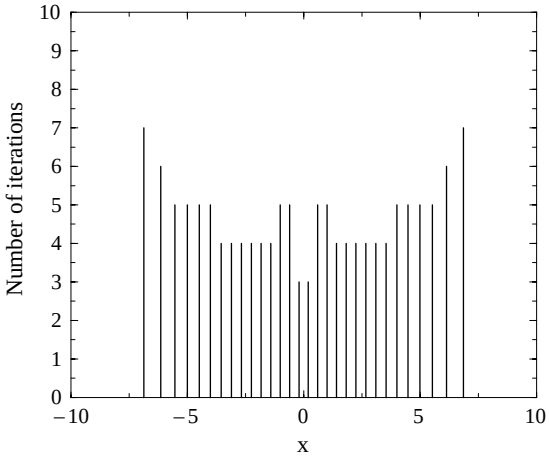

Fig. 1. Number of iterations for computing the zeros of $H_{30}(x)$.

Figs. 1, 2, 4 and 5 show a similar pattern in the distribution of iterations needed to compute each zero. In all four cases η is zero at $x = 0$ and expansive sweeps are considered starting from this point. If we had chosen $\alpha \neq \beta$, Fig. 5 would not be symmetric; the transition point would move apart from the origin. The method is expected to converge better as η is smaller; in fact, if $\eta = 0$ the ratio H_i is a tangent function in z (see Eqs. (10) and (20)) as, for instance, happens for Bessel functions of order $n = 1/2$. In this case the algorithm gives the correct answer in one iteration. This explains why the smallest zeros in modulus are the fastest to compute. As we move from $x = 0$ the number of iterations increases, and the largest zeros in modulus are the most demanding ones, as we previously discussed. The fact that close to $x = 0$ small peaks appear in the figures is an effect of the improvement of the iteration step, which becomes effective after the zeros corresponding to these peaks have been evaluated.

For the Laguerre case (Fig. 3) the transition point is at a positive value of x and the increase in the number of iterations is faster as we move to the left from this transition point.

The pattern in Fig. 6 (Bessel functions) and 7 (Conical functions) is similar with regard to the variation in the number of iterations and corresponds to a backward sweep, with a slight increase in the number of iterations as x becomes smaller. Of course, the distribution of zeros is quite different in each of these two cases, as could be expected given the different change of variables $z(x)$ associated to each case. The largest zeros require more iterations because the improved iteration step starts to operate after these initial zeros have been computed. Finally, Fig. 8 (Coulomb functions) corresponds again to an expansive sweep.

Our numerical implementation of the fixed point method proves to be an efficient algorithm for the evaluation of the zeros of SFs and it appears to be at least a factor 2 faster than the intrinsic Maple procedures for computing the zeros of Bessel functions. Besides, it is important to point out that external programs do not have the same privilege as intrinsic

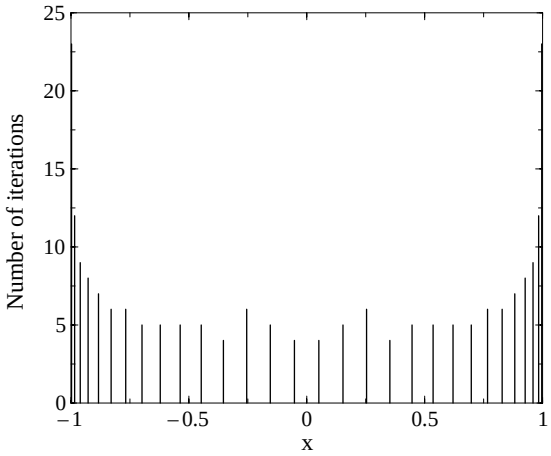

Fig. 2. Number of iterations for computing the zeros of $P_{30}(x)$.

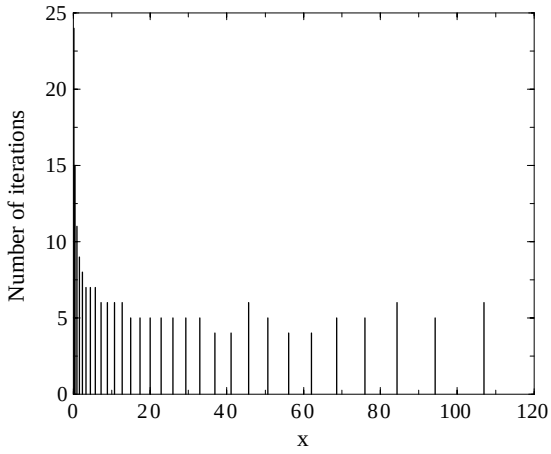

Fig. 3. Number of iterations needed for computing the zeros of $L_{30}^{(3/2)}(x)$.

Maple procedures. Also, the comparison of similar methods (Segura and Gil, 1999) (implemented in Fortran) with other available codes (Vrahatis et al., 1995), was favourable.

As for OPs, the Maple procedure *fsolve* is an interesting tool to compute such zeros. For moderately large orders we observe that our code tends to be faster, in spite of the fact that, as commented, it is difficult to compare the speed of external algorithms with intrinsic procedures. For instance, our algorithm is faster than *fsolve* (using Maple 7) for the computation of the zeros of Laguerre polynomials of moderately large order and negative parameter, like $L_{50}^{(-1/6)}(x)$.

The fixed point method converges with certainty even for extreme cases like, for instance, when computing the zeros of $P_{100}^{(-0.99,-0.99)}(x)$. For this polynomial, the

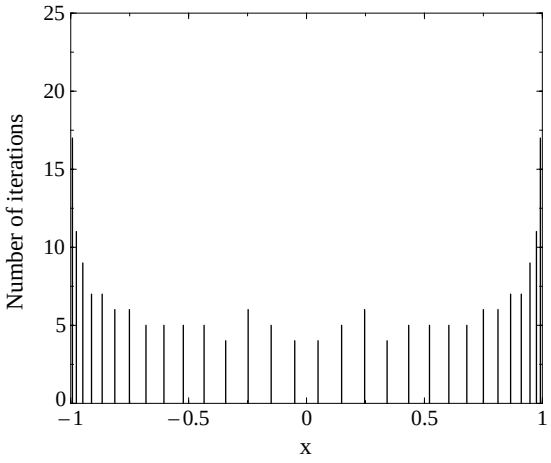

Fig. 4. Number of iterations needed for computing the zeros of $C_{30}^{(3/2)}(x)$.

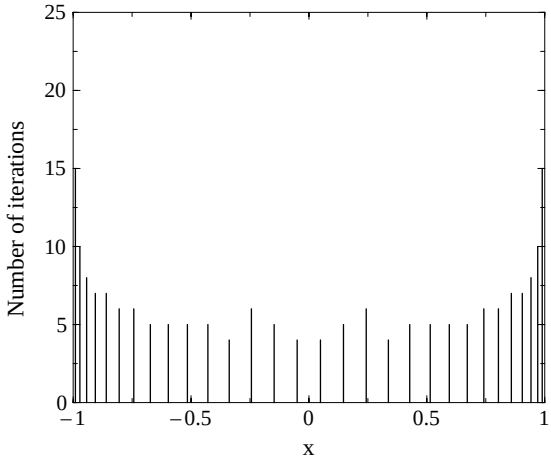

Fig. 5. Number of iterations needed for computing the zeros of $P_{30}^{(3/2,3/2)}(x)$.

convergence of *fsolve* is uncertain if a moderate number of digits is chosen (20 digits), producing zeros outside the interval $[-1, 1]$. For a larger number of digits (50 digits), this problem disappears but the convergence rate becomes very slow. This shows that *fsolve* (Maple 7) becomes quite unstable for these extreme cases (and for Maple V, *fsolve* fails to converge for more than 16 digits), losing too many significant digits. These problems are not observed for the fixed point method.

Furthermore, we expect to improve the performance for the computation of the extreme zeros in a forthcoming publication (Gil et al., in preparation). As discussed for Laguerre polynomials regarding the calculation of the smallest zeros, alternative sequences of

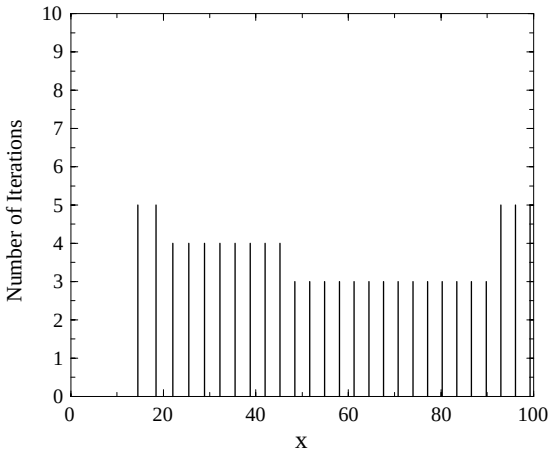

Fig. 6. Number of iterations needed for finding the zeros of $J_{10}(x)$ in the interval [1, 100].

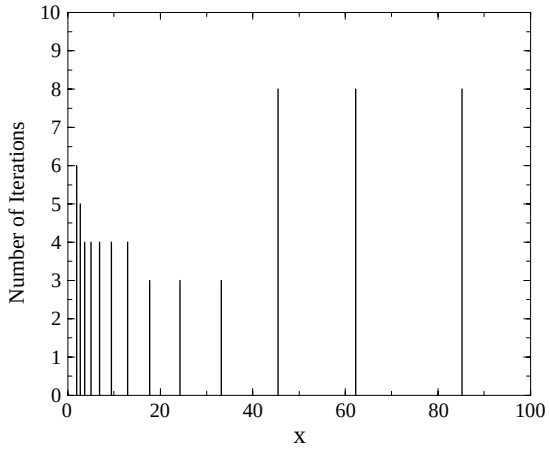

Fig. 7. Number of iterations needed for finding the zeros of $P^{10}_{-1/2+i10}(x)$ in the interval [1.1, 100].

functions solve the relatively slow convergence for the extreme zeros when extreme values of the parameters are considered. The comparison will be even more favourable then.

6. Conclusions

We have described a combined symbolic/numerical algorithm to compute the zeros of families of functions satisfying a TTRR together with a first order linear DDE. These conditions are satisfied by a broad family of SFs and OPs. Therefore, the method is applicable for the computation of the zeros of classical OPss and non-polynomial solutions of the same ODEs. The algorithm is also successful in computing the zeros of other SFs

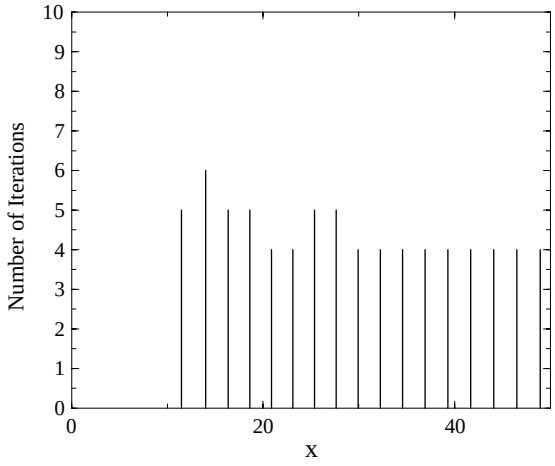

Fig. 8. Number of iterations needed for finding the zeros of $F_{20}(-20, x)$ in the interval [1, 50].

like, for instance, arbitrary solutions of the Bessel equation, the Coulomb wave equation, Hermite equation (parabolic cylinder functions), Legendre equation with complex degree (Conical functions) and Airy functions (whose zeros can be computed via the connection of Airy functions with Bessel functions (Segura, 1998)), among others.

In the literature, there exist several algorithms for the computation of zeros of the Bessel functions J_ν and Y_ν (Piessens, 1990; Segura and Gil, 1999; Temme, 1979; Vrahatis et al., 1995), but no methods and algorithms are available for other SFs, although matrix methods have been described to solve the problem for regular Coulomb wavefunctions (Ikebe, 1975; Ikebe et al., 1991) and a global Newton method was described in Segura (2001) to compute the zeros of general solutions of the Bessel equation. The algorithm here described is able to compute all these zeros efficiently.

As a bonus, the method, being based on analytical properties, provides analytical information on the spacing between the zeros of a given sequence of functions. It also provides all the information needed to implement the algorithms in compilable languages like Fortran. A Fortran template will also be provided to build specific programs for the computation of the zeros of families of functions within this (vast) set of holonomic functions.

Fixed point methods to compute the turning points of this type of function (Gil and Segura, submitted) are also available and will be implemented in the next update of the algorithm.

Acknowledgements

A. Gil acknowledges support from the A. von Humboldt foundation. J. Segura acknowledges support from DAAD. We thank the editors and the anonymous referees for valuable suggestions.

Appendix

```
> read ‘zeros.mpl‘;
```

This package computes the zeros of Special Functions
and Orthogonal Polynomials
(c) Amparo Gil (Dept. Matematicas, Universidad Autonoma de Madrid, Spain)
and
Javier Segura (Dept. Matematicas, Universidad Carlos III de Madrid, Spain)
Introduce number of digits for the computation

```
> 20;
```

Is your function one of the following:
1. Bessel functions
$c\,J_n(x) - \sqrt{1-c^2}Y_n(x)$, c in the interval $[-1, 1]$
2. Conical functions
$P_{n,-1/2+i,\tau}(x)$
3. Coulomb functions
$F_{n,\beta}(x)$
4. Gegenbauer polynomials
$C_{\alpha,n}(x)$
5. Generalized Laguerre polynomials
$L_{\alpha,n}(x)$
6. Hermite polynomials
$H_n(x)$
7. Jacobi polynomials
$P_{\alpha,\beta,n}(x)$
8. Legendre polynomials
$P_n(x)$
YES $-->$ *1*
NO $-->$ *2*

```
> 1;
```

Which is your function. Select 1, 2, . . . , 8

```
> 4;
```

Values of the parameters of the functions
Parameter n

```
> 10;
```

Parameter α

```
> .5;
```

The user must now choose performing only the symbolic tasks
or both symbolic and numerical tasks
0 $-->$ SYMBOLIC
1 $-->$ SYMBOLIC + NUMERIC

```
> 1;
```

Provide the following information
a) Interval for the calculation of zeros [xa, xb]
Remember that [xa, xb] must be in the interval $]-1, 1[$

```
> -.9;
> .9;
```

c) Precision required for the computation of the zeros

```
10^(-12);
```

d) Maximum number of iterations allowed (recommended: 50)

```
> 50;
```

First check ok
Second check ok
A has a maximum

1. Change of variables $z(x) := \sqrt{(n+2\alpha)(n+1)}\ \mathrm{arctanh}(x)$

2. $A := -\frac{1}{4}\frac{-4n^2 - 8n\alpha + 4x^2n^2 + 8x^2n\alpha - x^2 + 4x^2\alpha^2 + 2 - 4\alpha}{(n+2\alpha)(n+1)}$

3. Parameter $eta(x) := -\frac{1}{2}\frac{x(2n+1+2\alpha)}{\sqrt{n+2\alpha}\sqrt{n+1}}$

4. Normalization factor $Ki := \sqrt{\frac{n+2\alpha}{n+1}}$

5. Iteration 1

6. Sign of dn_i, 1

NUMERICAL OUTPUTS:

1) TYPE OF SWEEP:
Expansive

2) ZEROS:

```
xcero[  1  ]:=  -.86506336668898451072
xcero[  2  ]:=  -.67940956829902440623
xcero[  3  ]:=  -.43339539412924719080
xcero[  4  ]:=  -.14887433898163121089
xcero[  5  ]:=  .14887433898163121089
xcero[  6  ]:=  .43339539412924719080
xcero[  7  ]:=  .67940956829902440623
xcero[  8  ]:=  .86506336668898451072
```

Number of iterations needed for each zero:

```
it[  1  ]:=  8
it[  2  ]:=  8
it[  3  ]:=  6
it[  4  ]:=  4
it[  5  ]:=  4
it[  6  ]:=  6
it[  7  ]:=  8
it[  8  ]:=  8
```

Number of zeros found := 8

References

Abramowitz, M., Stegun, I.A. (Eds.), 1972. Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables. Reprint of the 1972 ed. A Wiley–Interscience Publication. Selected Government Publications. John Wiley & Sons, Inc, New York; National Bureau of Standards, Washington, D.C.

Gil, A., Koepf, W., Segura, J., Numerical algorithms for the computation of zeros of hypergeometric functions (in preparation).

Gil, A., Segura, J., Computing zeros and turning points of special functions from fixed point methods (submitted for publication).

Golub, G.H., Welsch, J.H., 1969. Calculation of Gauss quadrature rules. Math. Comp. 23, 221–230.

Ikebe, Y., 1975. The zeros of regular Coulomb wave functions and of their derivatives. Math. Comp. 29, 878–887.

Ikebe, Y., Kikuchi, Y., Fujishiro, I., 1991. Computing zeros and orders of Bessel functions. J. Comput. Appl. Math. 38, 169–184.

Marx, I., 1953. On the structure of recurrence relations II. Michigan Math. J. 2, 99–103.

Piessens, R., 1990. On the computation of zeros and turning points of Bessel functions. Bull. Greek Math. Soc. 31, 117–122.

Press, W.H., Teukolski, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in Fortran. Cambridge University Press, Cambridge.

Segura, J., 1998. A global Newton method for the zeros of cylinder functions. Numer. Algorithms 18, 259–276.

Segura, J., 2001. Bounds on differences of adjacent zeros of Bessel functions and iterative relations between consecutive zeros. Math. Comp. 70, 1205–1220.

Segura, J., 2002. The zeros of special functions from a fixed point method. SIAM J. Numer. Anal. 40 (1), 114–133.

Segura, J., 2003. On the zeros and turning points of special functions. J. Comput. Appl. Math. (in press).

Segura, J., Gil, A., 1999. ELF and GNOME: two tiny codes to evaluate the real zeros of the Bessel functions of the first kind for real orders. Comput. Phys. Comm. 117, 250–262.

Temme, N.M., 1979. An algorithm with ALGOL 60 program for the computation of the zeros of ordinary Bessel functions and those of their derivatives. J. Comput. Phys. 32, 270–279.

Vrahatis, M.N., Ragos, O., Skiniotis, T., Zafiropoulos, F.A., Grapsa, T.N., 1995. RFSFNS: a portable package for the numerical determination of the number and the calculation of roots of Bessel functions. Comput. Phys. Comm. 92, 252–266.

Wilf, H.S., 1962. Mathematics for the Physical Sciences, Wiley, New York (Problem 9).