Article Submitted to Journal of Symbolic Computation

Symbolic computation of exact solutions expressible in hyperbolic and elliptic functions for nonlinear partial differential and differential-difference equations *

D. Baldwin[1], Ü. Göktaş[2], W. Hereman[1,3], L. Hong[4], R. S. Martino[5] and J. C. Miller[6]

[1] *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887, U.S.A.*

[2] *Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820, U.S.A.*

[3] *Department of Applied Mathematics, University of Stellenbosch, Private Bag X1, 7602 Matieland, South Africa*

[4] *Department of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, U.S.A.*

[5] *Center for Subsurface Modeling, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin, Austin TX 78712, U.S.A.*

[6] *Department of Applied Mathematics and Theoretical Physics, Churchill College, University of Cambridge, Cambridge CB3 0DS, England, U.K.*

Abstract

The *Mathematica* implementation of the tanh and sech-methods for computing exact travelling wave solutions of nonlinear partial differential equations (PDEs) is presented. These methods also apply to ordinary differential equations (ODEs).

New algorithms are given to compute polynomial solutions of ODEs and PDEs in terms of the Jacobi elliptic functions. An adaptation of the tanh-method to nonlinear differential-difference equations (DDEs) is also presented.

The new *Mathematica* packages, PDESpecialSolutions.m and DDESpecialSolutions.m, automatically compute closed-form solutions which are expressible as polynomials in the tanh, sech, sn and cn functions.

For systems of ODEs, PDEs, or DDEs with constant parameters, the software finds the conditions on the parameters so that the given differential equations admit solutions involving tanh, sech, both, sn or cn.

*Research supported in part by the National Science Foundation under Grants DMS-9732069, DMS-9912293 and CCR-9901929.

⋆ Correspondence per email: whereman@mines.edu

Several examples illustrate the capabilities of the software. New solutions are presented for some nonlinear PDEs and DDEs.

Use of the software, implementation issues, scope, limitations, and future extensions of the software are addressed.

1. Introduction

In the study of nonlinear wave phenomena, travelling wave solutions of partial differential equations (PDEs) and differential-difference equations (DDEs) have physical relevance. For example, the travelling wave solutions of the Korteweg-de Vries (KdV) and Boussinesq equations describe water waves. The travelling wave solution of the Toda lattice models the vibration of masses in a lattice with exponential interaction force. Often, bell shaped sech-solutions and kink shaped tanh-solutions model wave phenomena in fluid dynamics, plasmas, elastic media, electrical circuits, optical fibers, chemical reactions, bio-genetics, etc.

The knowledge of closed form solutions of nonlinear PDEs and ODEs facilitates the testing of numerical solvers, and aids in the stability analysis of solutions. Indeed, the real closed-form solutions given in this paper correspond to homoclinic and heteroclinic orbits in phase space, which are the separatrices of stable and unstable regions.

There is a variety of special methods to find solitary wave solutions of nonlinear ODEs and PDEs. Among the more algorithmic techniques are the Hirota method (see Hereman and Zhuang, 1995), the real exponential method (Hereman *et al.* 1986, Hereman and Takaoka 1990), and the singular manifold method (for references consult e.g. Estévez and Gordoa (1995, 1998), Estévez and Hernáez (2000), Hereman (1990), and Hereman and Nuseir (1997)). Some of these direct methods are straightforward to implement in any computer algebra system.

More comprehensive methods to find exact solutions of PDEs and ODEs are based on similarity reductions via Lie point symmetry methods (see e.g. Ludlow *et al.* (1999), and Hereman (1996) for references to publications and software).

Maple v. 7 offers a package *PDEtools*, which contains the function *pdesolve* to find exact solutions of classes of linear and non-linear PDEs. For more information consult Cheb-Terrab (1995, 2001ab). The methods presented in this paper are independent from these efforts.

Travelling wave solutions of many nonlinear PDEs, ODEs, and DDEs from soliton theory (and beyond) can be expressed as polynomials of the hyperbolic tangent and secant functions. One explanation is given in Hereman *et al.* (1986) and Hereman and Takaoka (1990). The tanh method provides a straightforward algorithm to compute such particular solutions for a large class of nonlinear PDEs. See Malfliet (1992), Malfliet and Hereman (1996), and Das and Sarma (1999) for a multitude of references to tanh-based techniques and various applications. The appeal and success of the method lies in the fact that one circumvents integration to get explicit solutions.

Here is how the tanh method works: For simplicity, let us consider a single polynomial PDE in $u(x,t)$. In a travelling frame of reference, $\xi = c_1 x + c_2 t + \Delta$, one transforms the PDE into an ordinary differential equation (ODE) in the new independent variable $T = \tanh \xi$. Doing so, one takes advantage of the property that the derivative of tanh is polynomial in tanh, i.e. $T' = 1 - T^2$. Consequently, all derivatives of T are polynomials of T. Via a chain rule, the PDE in $u(x,t)$ is thus transformed into a nonlinear ODE in $U(T)$, which has polynomial coefficients in T. One then seeks polynomial solutions of the ODE, thus generating a sub-class of a perhaps much larger class of solutions.

As a matter of fact, one encounters ODEs which are nonlinear, higher-order versions of the so-called ultraspherical differential equation,

$$(1 - x^2) y''(x) - (2\alpha + 1) x y'(x) + n(n + 2\alpha) y(x) = 0, \tag{1}$$

with integer $n \geq 0$ and α real, whose solutions are the Gegenbauer polynomials. Eq. (1) includes the Legendre equation ($\alpha = \frac{1}{2}$), satisfied by the Legendre polynomials, and the ODEs for Chebeyshev polynomials of type I ($\alpha = 0$) and type II ($\alpha = 1$).

As an aside, one encounters the associated Legendre equation,

$$(1 - x^2)^2 y''(x) - 2x(x^2 - 1) y'(x) + [n(n+1)(1 - x^2) - m^2] y(x) = 0, \tag{2}$$

with m and n non-negative integers, in solving the Sturm-Liouville problem for the KdV with a sech^2 potential (see Drazin and Johnson (1989) p. 45).

The origin of the tanh method has been subject to debate (see Verheest *et al.* (1999) and Das and Sarma (1999)). Variants of the method indeed appear in the mathematical physics, plasma physics, and fluid dynamics literature (see Das and Sarma (1999) for references). The tanh method applies to nonlinear polynomial systems of PDEs (in any number of independent variables) and has been successfully applied to many nonlinear problems (see e.g. Malfliet (1994), Fan (2000), and Gao and Tian (2001) for additional references). Viewed as a special case of PDEs, the method applies to ODEs.

In Parkes and Duffy (1996) the authors mention the difficulty of using the tanh method by hand for anything but simple PDEs. Therefore, they automated to some degree the tanh method using *Mathematica* (Wolfram, 1999). In a subsequent paper, Parkes *et al.* (1998) consider solutions to (odd-order generalized KdV) equations in even powers of sech. Their methods and software are not designed to find solutions involving odd powers of sech.

In this paper we present three flavors of tanh- and sech-methods as they apply to nonlinear polynomial systems of PDEs and ODEs. We also present the fully automated software package, **PDESpecialSolutions.m** in *Mathematica*, which implements these tanh- and sech-methods. In contrast to the code of Parkes and Duffy (1996), our software performs the computations from start to finish, and our key function PDESolve assumes a similar format as DSolve in *Mathematica*. Also, for PDEs, the number of independent variables x_i is no longer limited to one space variable x and time t. Our software can also handle any number of dependent variables.

Based on the strategy of the tanh-method, we also present new algorithms to compute polynomial solutions of ODEs and PDEs in terms of the Jacobi elliptic sine and cosine functions. Applied to the KdV equation, the so-called cnoidal solution (Drazin and Johnson (1989) p. 26-30) is obtained. For the Duffing equation (Lawden, 1989), we recovered known sn and cn-solutions which model vibrations of a nonlinear spring.

Without intervention from the user, our software computes travelling wave solutions as polynomials in either $T = \tanh \xi$, $S = \operatorname{sech} \xi$, both, $\mathrm{CN} = \operatorname{cn}(\xi; m)$, or $\mathrm{SN} = \operatorname{sn}(\xi; m)$ with $\xi = c_1 x + c_2 y + c_3 z + \ldots + c_n t + \Delta = \sum_{j=0}^{N} c_j x_j + \Delta$. The c_j have physical meaning: the spatial coordinate coefficients are the components of the wave vector and the time coefficient is the angular frequency of the wave. The wave travels in the direction of the wave vector; its plane wave front is perpendicular to that wave vector. Δ denotes the constant phase of the wave.

An adaptation of the tanh-method to nonlinear differential-difference equations (DDEs) is also presented and implemented. A separate package, called **DDESpecialSolutions.m**, solves polynomial systems of semi-discrete equations. These systems arise, for example, from PDEs for $u(x, t)$ where the space variable x is discretized and time t is kept continuous. To avoid confusion with fully discretized lattice equations, we refer to such equations as differential-difference equations (DDEs for short).

Finally, for systems of PDEs or DDEs involving constant parameters, the software packages automatically determine the conditions on the parameters so that the given equations might admit solutions involving tanh, sech, combinations of both, sn or cn.

In Sections 2 and 3, we outline the steps of the tanh- and sech-methods for nonlinear polynomial PDEs. We restrict ourselves to solutions which are polynomial in either tanh or sech. We use the Boussinesq equation and a 3 dimensional modified KdV equation to illustrate the steps of both methods. In Section 4 we consider a more general class of polynomial solutions involving both tanh and sech. The sech-tanh method is illustrated by solving a system of three coupled evolution equations. In Section 5 we show how modifying the chain rule allows us to find polynomial solutions in the Jacobi elliptic functions sn and cn. Section 6 shows how the tanh method can be applied to polynomial DDEs, using the Toda lattice as a example. In Sections 7 and 8, we present exact solutions for a variety of nonlinear polynomial ODEs, PDEs and DDEs. Some (presumably) new solutions are given. Section 9 addresses other perspectives and possible extensions of the algorithms. The use of the packages is shown in Section 10. In Section 11 we discuss implementation issues, scope and limitations of our software. We draw some conclusions in Section 12.

2. Tanh method for nonlinear PDEs

In this section, we outline the steps of the tanh method (Malfliet, 1996) and illustrate each step for the Boussinesq equation.

Given is a system of M polynomial PDEs of order m and with constant coefficients,

$$\boldsymbol{\Delta}(\mathbf{u}(\mathbf{x}), \mathbf{u}'(\mathbf{x}), \mathbf{u}''(\mathbf{x}), \cdots, \mathbf{u}^{(k)}(\mathbf{x}), \cdots, \mathbf{u}^{(m)}(\mathbf{x})) = \mathbf{0}, \tag{3}$$

where the dependent variable $\mathbf{u}$ has M components u_i and the independent variable $\mathbf{x}$ has N components x_j. Furthermore, $\mathbf{u}^{(k)}(\mathbf{x})$ denotes the collection of mixed derivative terms of order k. Any constant parameters in (3) will be denoted by lower Greek letters.

For simplicity of notation, in the examples in Section 7 we will denote the dependent variables by u, v, w, etc. and the independent variables by x, y, z, and t.

Example: The Boussinesq equation,

$$u_{tt} - u_{xx} + 3uu_{xx} + 3u_x^2 + \alpha u_{xxxx} = 0, \tag{4}$$

with real parameter α, was proposed by Boussinesq to describe surface water waves whose horizontal scale is much larger than the depth of the water (Ablowitz and Clarkson, 1991).

One could apply the tanh method directly to (4). However, to show the method for a simple system of PDEs, we recast (4) as a first order system in time,

$$\begin{aligned} &u_{1,x_2} + u_{2,x_1} = 0, \\ &u_{2,x_2} + u_{1,x_1} - 3u_1u_{1,x_1} - \alpha u_{1,3x_1} = 0, \end{aligned} \tag{5}$$

where $x_1 = x$, $x_2 = t$, and $u_1(x_1, x_2) = u(x, t)$. We also introduced the auxiliary variable $u_2(x_1, x_2)$, using the notation

$$u_{i,kx_j} \stackrel{\text{def}}{=} \frac{\partial^k u_i}{\partial x_j^k}, \quad u_{i,px_j\,rx_k\,sx_\ell} \stackrel{\text{def}}{=} \frac{\partial^{p+r+s} u_i}{\partial x_j^p \partial x_k^r \partial x_\ell^s}, \text{ etc.} \tag{6}$$

Step T1: Transform the PDE into a nonlinear ODE

Seek solutions in a travelling frame of reference,

$$\xi = \sum_{j=1}^{N} c_j x_j + \Delta, \tag{7}$$

where the components c_j of the wave vector and the phase Δ are constant.

The tanh method seeks polynomial solutions expressible in the hyperbolic tangent, $T = \tanh\xi$. The first and, consequently, all higher-order derivatives are polynomials in T. Based on the identity

$$\cosh^2\xi - \sinh^2\xi = 1, \tag{8}$$

one computes

$$\begin{aligned} \tanh'\xi &= \mathrm{sech}^2\xi = 1-\tanh^2\xi, \\ \tanh''\xi &= -2\tanh\xi + 2\tanh^3\xi, \quad \text{etc.} \end{aligned} \tag{9}$$

Since $T' = 1 - T^2$, repeatedly applying the chain rule,

$$\frac{\partial \bullet}{\partial x_j} = \frac{d\bullet}{dT}\frac{dT}{d\xi}\frac{\partial \xi}{\partial x_j} = c_j(1-T^2)\frac{d\bullet}{dT}, \tag{10}$$

transforms the given system of PDEs into a coupled system of nonlinear ODEs,

$$\boldsymbol{\Delta}(T, \mathbf{U}(T), \mathbf{U}'(T), \mathbf{U}''(T), \ldots, \mathbf{U}^{(m)}(T)) = \mathbf{0}, \tag{11}$$

for $\mathbf{U}(T) = \mathbf{u}(\mathbf{x})$, and where each component of $\boldsymbol{\Delta}$ is a nonlinear ODE with polynomial coefficients in T.

Example: Substituting

$$\begin{aligned} u_{i,x_j} &= c_j(1-T^2)U_i', \\ u_{i,2x_j} &= c_j^2(1-T^2)\left[(1-T^2)U_i'\right]' = c_j^2(1-T^2)[-2TU_i' + (1-T^2)U_i''], \\ u_{i,3x_j} &= c_j^3(1-T^2)\left[-2T(1-T^2)U_i' + (1-T^2)^2U_i''\right]' \\ &= c_j^3(1-T^2)\left[-2(1-3T^2)U_i' - 6T(1-T^2)U_i'' + (1-T^2)^2U_i'''\right], \end{aligned} \tag{12}$$

into (5), and cancelling common factors $(1-T^2)$, yields

$$\begin{aligned} & c_2U_1' + c_1U_2' = 0, \\ & c_2U_2' + c_1U_1' - 3c_1U_1U_1' + \alpha c_1^3\left[2(1-3T^2)U_1' + 6T(1-T^2)U_1'' - (1-T^2)^2U_1'''\right] = 0, \end{aligned} \tag{13}$$

with $U_1(T) = u_1(x_1, x_2)$ and $U_2(T) = u_2(x_1, x_2)$.

Step T2: Determine the degree of the polynomial solutions

We seek polynomial solutions of the form

$$U_i(T) = \sum_{j=0}^{M_i} a_{ij}T^j. \tag{14}$$

Before the a_{ij} can be computed, the leading exponents M_i must be determined. To avoid trivial solutions, we assume that $M_i \geq 1$. Substituting U_i into (11), the coefficients of every power of T in every equation must vanish. In particular, the highest degree terms must vanish. Since the highest degree terms depend only on T^{M_i} in (14), substituting $U_i(T) = T^{M_i}$ into the left hand side of (11) is sufficient. This results in a polynomial system $\mathbf{P}(T)$. Equating every two possible highest exponents in every component P_i gives the linear system that determines the M_i. The linear system is then solved for the unknowns M_i. Carry each solution to step T3. In the event that one or more of the exponents M_i remain undetermined, assign integer values to the free M_i so that every equation in (11) has at least two different terms with equal highest exponents.

Example: For the Boussinesq case, substituting $U_1(T) = T^{M_1}$ and $U_2(T) = T^{M_2}$ into (13) and equating the highest exponents of T for each equation, gives

$$M_1 - 1 = M_2 - 1, \quad 2M_1 - 1 = M_1 + 1. \tag{15}$$

So, $M_1 = M_2 = 2$, and

$$\begin{aligned} U_1(T) &= a_{10} + a_{11}T + a_{12}T^2, \\ U_2(T) &= a_{20} + a_{21}T + a_{22}T^2. \end{aligned} \tag{16}$$

Step T3: Derive the algebraic system for the coefficients a_{ij}

To derive the system for the unknown coefficients a_{ij} and wave parameters c_j, substitute (14) into (11) and set the coefficients of the various power terms of T to zero. The resulting nonlinear algebraic system for the unknowns a_{ij} is parameterized by the wave parameters c_j and the parameters (in lower Greek letters) of system (3), if any.

Example: For (5), after substitution of (16) into (13), one collects the terms of like degree in T, which must vanish independently. Thus,

$$\begin{aligned} a_{11}\, c_1\, (3a_{12} + 2\alpha\, c_1^2) &= 0, \\ a_{12}\, c_1\, (a_{12} + 4\alpha\, c_1^2) &= 0, \\ a_{21}\, c_1 + a_{11}\, c_2 &= 0, \\ a_{22}\, c_1 + a_{12}\, c_2 &= 0, \\ a_{11}\, c_1 - 3a_{10}\, a_{11}\, c_1 + 2\alpha a_{11}\, c_1^3 + a_{21}\, c_2 &= 0, \\ -3a_{11}^2\, c_1 + 2\, a_{12}\, c_1 - 6a_{10}\, a_{12}\, c_1 + 16\alpha\, a_{12}\, c_1^3 + 2a_{22}\, c_2 &= 0, \end{aligned} \tag{17}$$

with unknowns $a_{10}, a_{11}, a_{12}, a_{20}, a_{21}, a_{22}$, parameters c_1, c_2, and α.

Step T4: Solve the nonlinear parameterized algebraic system

The hardest part of the method is solving the nonlinear algebraic system, for which we designed a customized, yet powerful, nonlinear solver (see Section 11 for details).

The nonlinear algebraic system is solved under the following assumptions: (i) all parameters, α, β, etc., in (3) are nonzero. Note that for vanishing parameters the exponents M_i of step T2 might change; (ii) the coefficients $a_{i\,M_i},\ i = 1, \cdots, M$ of the highest power terms in (14) are nonzero (again for consistency with step T2); and (iii) all c_j are nonzero (demanded by the physical nature of the solutions).

Example: Assuming c_1, c_2, a_{12}, a_{22}, and α nonzero, the solution of (17) reads

$$\begin{aligned} a_{10} &= (c_1^2 - c_2^2 + 8\alpha c_1^4)/(3c_1^2), \\ a_{11} &= 0, \\ a_{12} &= -4\alpha c_1^2, \\ a_{20} &= \text{arbitrary}, \\ a_{21} &= 0, \\ a_{22} &= 4\alpha c_1 c_2. \end{aligned} \tag{18}$$

There is no condition on the parameters c_1, c_2 and α.

Step T5: Build and test the solitary wave solutions

Substitute the solutions obtained in step T4 into (14) and reverse step T1 to obtain the explicit solutions in the original variables. Finally, test the solutions by substituting them into (3). If so desired, reject complex solutions. See Section 11 for the reasons for the extra testing.

Example: Inserting (18) into (16), and using $T = \tanh(c_1 x + c_2 t + \Delta)$, a closed form solution for (5) (or (4)) reads

$$\begin{aligned} u(x,t) = u_1(x,t) &= \frac{c_1^2 - c_2^2 + 8\alpha c_1^4}{3c_1^2} - 4\alpha c_1^2 \tanh^2(c_1 x + c_2 t + \Delta), \\ u_2(x,t) &= a_{20} + 4\alpha c_1 c_2 \tanh^2(c_1 x + c_2 t + \Delta), \end{aligned} \tag{19}$$

where a_{20}, c_1, c_2, α and Δ are arbitrary. Steps T1 through T5 should be repeated if one or more of the external parameters (lower Greek letters) were set to zero.

3. Sech method for nonlinear PDEs

In this section we give the steps of the sech-method as it applies to (3). We restrict ourselves to solutions that are polynomial in sech alone. More general solutions involving both sech and tanh will be dealt with in Section 4.

Using (8), or equivalently,

$$\tanh^2\xi + \operatorname{sech}^2\xi = 1, \tag{20}$$

the solution (19) of (5) can be expressed in terms of sech. Indeed,

$$\begin{aligned} u_1(x,t) &= \frac{c_1^2 - c_2^2 - 4\alpha c_1^4}{3c_1^2} + 4\alpha c_1^2 \operatorname{sech}^2(c_1 x + c_2 t + \Delta), \\ u_2(x,t) &= a_{20} + 4\alpha c_1 c_2 - 4\alpha c_1 c_2 \operatorname{sech}^2(c_1 x + c_2 t + \Delta). \end{aligned} \tag{21}$$

Any even order solution in tanh can be written as an even order solution in sech. However, some PDEs have polynomial solutions in sech of odd order. For example, the modified KdV equation (Ablowitz and Clarkson, 1991),

$$u_t + \alpha u^2 u_x + u_{xxx} = 0, \tag{22}$$

has the solution

$$u(x,t) = \pm c_1 \sqrt{\frac{6}{\alpha}} \operatorname{sech}(c_1 x - c_1^3 t + \Delta), \tag{23}$$

which is not expressible as a polynomial in tanh, and so cannot be found using the tanh method. We now introduce a sech method to compute solutions involving any integer order of sech.

Example: We illustrate the steps for a three dimensional modified KdV (3D-mKdV) equation,

$$u_t + 6u^2 u_x + u_{xyz} = 0, \tag{24}$$

or

$$u_{1,x_4} + 6u_1^2 u_{1,x_1} + u_{1,x_1 x_2 x_3} = 0, \tag{25}$$

identifying $u_1(x_1, x_2, x_3, x_4) = u(x, y, z, t)$.

Step S1: Transform the PDE into a nonlinear ODE

Adhering to a travelling frame of reference (7), use (20) to compute

$$\operatorname{sech}'\xi = -\operatorname{sech}\xi \tanh\xi = -\operatorname{sech}\xi\sqrt{1-\operatorname{sech}^2\xi}. \tag{26}$$

Set $S = \operatorname{sech}\xi$ and repeatedly apply the chain rule,

$$\frac{\partial \bullet}{\partial x_j} = \frac{d\bullet}{dS}\frac{dS}{d\xi}\frac{\partial \xi}{\partial x_j} = -c_j S\sqrt{1-S^2}\frac{d\bullet}{dS}, \tag{27}$$

to transform (3) into a nonlinear ODE of the form:

$$\mathbf{\Gamma}(S, \mathbf{U}(S), \mathbf{U}'(S), \ldots) + \sqrt{1-S^2}\,\mathbf{\Pi}(S, \mathbf{U}(S), \mathbf{U}'(S), \ldots) = \mathbf{0}, \tag{28}$$

for $\mathbf{U}(S) = \mathbf{u}(\mathbf{x})$, and where all components of $\mathbf{\Gamma}$ and $\mathbf{\Pi}$ are ODEs with polynomial coefficients in S. Note that any term in (3) for which the total number of derivatives is even contributes to the first term in (28); terms of odd order contribute to the second term.

First, we consider a special case of (28), assuming that either $\mathbf{\Gamma}$ or $\mathbf{\Pi}$ is identically $\mathbf{0}$.

$$\mathbf{\Delta}(S, \mathbf{U}(S), \mathbf{U}'(S), \ldots) = \mathbf{0}. \tag{29}$$

where $\mathbf{\Delta}$ stands for either $\mathbf{\Gamma}$ or $\mathbf{\Pi}$, whichever is nonzero. To get (29), the order of all terms in any equation in (3) must be even or odd (the latter is true in (25)).

Eq. (28) for which both $\mathbf{\Gamma} \neq \mathbf{0}$ and $\mathbf{\Pi} \neq \mathbf{0}$ will be dealt with in Section 4.

Example: Substituting

$$\begin{aligned}
u_{i,x_j} &= -c_j S\sqrt{1-S^2}U_i', \\
u_{i,x_j x_k} &= c_j c_k S\sqrt{1-S^2}\left[S\sqrt{1-S^2}U_i'\right]' \\
&= c_j c_k S[(1-2S^2)U_i' + S(1-S^2)U_i''], \\
u_{i,x_j x_k x_l} &= -c_j c_k c_l S\sqrt{1-S^2}\left[S(1-2S^2)U_i' + S(1-S^2)U_i''\right]' \\
&= -c_j c_k c_l S\sqrt{1-S^2}\left[(1-6S^2)U_i' + 3S(1-2S^2)U_i'' + S^2(1-S^2)^2 U_i'''\right],
\end{aligned} \tag{30}$$

into (25), and cancelling the common factor $S\sqrt{1-S^2}$, yields

$$c_4 U_1' + 6c_1 U_1^2 U_1' + c_1 c_2 c_3 \left[(1-6S^2)U_1' + 3S(1-2S^2)U_1'' + S^2(1-S^2)^2 U_1'''\right] = 0, \quad (31)$$

with $U_1(S) = u_1(x_1, x_2, x_3, x_4)$. Note that (31) matches (29) with $\mathbf{\Delta} = \mathbf{\Pi}$, since $\mathbf{\Gamma} = \mathbf{0}$. A common factor $\sqrt{1-S^2}$ was indeed cancelled.

Step S2: Determine the degree of the polynomial solutions
Seek polynomial solutions,

$$U_i(S) = \sum_{j=0}^{M_i} a_{ij} S^j. \quad (32)$$

To determine the exponents M_i, substitute $U_i(S) = S^{M_i}$ into the left hand side of (29) and proceed as in step T2, assuming $M_i \geq 1$. Continue with step S3 for each solution M_i. If some exponents M_i remain undetermined, try all legitimate values for such free M_i. See Section 7.3 for an example.

Example: For (25), substituting $U_1(S) = S^{M_1}$ into (31) and equating the highest exponents yields

$$3M_1 - 1 = M_1 + 1. \quad (33)$$

So, $M_1 = 1$, and

$$U_1(S) = a_{10} + a_{11}S. \quad (34)$$

Step S3: Derive the algebraic system for the coefficients a_{ij}
Follow the strategy of step T3.

Example: For (25), after substitution of (34) into (31), one obtains

$$\begin{aligned} a_{11}(c_4 - 6c_1 a_{10}^2 + c_1 c_2 c_3) &= 0, \\ 12 c_1 a_{10} a_{11}^2 &= 0, \\ 6 a_{11} c_1 (a_{11}^2 - c_2 c_3) &= 0. \end{aligned} \quad (35)$$

Step S4: Solve the nonlinear parameterized algebraic system
Similar strategy as in step T4.

Example: For c_1, c_2, c_3, and a_{11} nonzero, the solution of (35) is

$$\begin{aligned} a_{10} &= 0, \\ a_{11} &= \pm\sqrt{c_2 c_3}, \\ c_4 &= -c_1 c_2 c_3, \end{aligned} \quad (36)$$

where c_1, c_2 and c_3 are arbitrary.

Step S5: Build and test the solitary wave solutions
Substitute the result of step S4 into (32) and reverse step S1. Test the solutions.

Example: The solitary wave solution of (24) reads

$$u(x,y,z,t) = \pm\sqrt{c_2 c_3}\,\mathrm{sech}(c_1\,x + c_2\,y + c_3\,z - c_1 c_2 c_3\,t + \Delta), \quad (37)$$

where c_1, c_2, c_3 and Δ are arbitrary. Steps S1-S5 should be repeated if one or more of the parameters in (3) were set to zero.

4. Mixed sech-tanh method for nonlinear PDEs

In this section we give the steps of a sech-tanh method to solve (3). The method below is designed to find particular solutions of (28) when $\mathbf{\Gamma} \neq \mathbf{0}$ and $\mathbf{\Pi} \neq \mathbf{0}$.

What came to mind first was applying the method of Section 3 to (28) in 'squared' form $\mathbf{\Gamma}^2(S, \mathbf{U}(S), \mathbf{U}'(S), \ldots) - (1 - S^2)\,\mathbf{\Pi}^2(S, \mathbf{U}(S), \mathbf{U}'(S), \ldots) = \mathbf{0}$. For anything but simple cases, the computations were unwieldy and the idea was shelved.

Since $T(S) = \sqrt{1-S^2}$ one envisions that, among others, (28) might admit solutions of the form

$$U_i(S) = \sum_{j=0}^{\tilde{M}_i} \sum_{k=0}^{\tilde{N}_i} \tilde{a}_{i,jk} S^j T^k. \tag{38}$$

However, (38) can always be rearranged as

$$U_i(S) = \sum_{j=0}^{M_i} a_{ij} S^j + T \sum_{j=0}^{N_i} b_{ij} S^j = \sum_{j=0}^{M_i} a_{ij} S^j + \sqrt{1-S^2} \sum_{j=0}^{N_i} b_{ij} S^j. \tag{39}$$

Remarkably, Eq. (29) where $\sqrt{1-S^2}$ is not explicitly present, also admits solutions of the form (39). See Section 7.6 for an example. The polynomial solutions in S from Section 3 are special cases of this broader class.

Computing solutions of type (32) with the sech-tanh method (presented below) would be inefficient and costly, as the example below and the examples in Sections 7.5 and 7.6 will show.

Example: We illustrate the method for the system (Gao and Tian, 2001):

$$\begin{aligned} &u_t - u_x - 2v = 0, \\ &v_t + 2uw = 0, \\ &w_t + 2uv = 0, \end{aligned} \tag{40}$$

or

$$\begin{aligned} &u_{1,x_2} - u_{1,x_1} - 2u_2 = 0, \\ &u_{2,x_2} + 2u_1 u_3 = 0, \\ &u_{3,x_2} + 2u_1 u_2 = 0, \end{aligned} \tag{41}$$

upon identification of $u_1 = u$, $u_2 = v$, $u_3 = w$, $x_1 = x$, and $x_2 = t$.

Step ST1: Transform the PDE into a nonlinear ODE

Same as in step S1.

Example: Use (27) to transform (41) into

$$\begin{aligned} &(c_1 - c_2) S\sqrt{1-S^2}\, U_1' - 2U_2 = 0, \\ &c_2 S\sqrt{1-S^2}\, U_2' - 2U_1 U_3 = 0, \\ &c_2 S\sqrt{1-S^2}\, U_3' - 2U_1 U_2 = 0. \end{aligned} \tag{42}$$

with $U_i(S) = u_i(x_1, x_2),\ i = 1, 2, 3.$

Step ST2: Determine the degree of the polynomial solutions

Seeking solutions (39), first determine the leading exponents M_i and N_i.

Substitute $U_i(S) = a_{i0} + a_{i\,M_i} S^{M_i} + \sqrt{1-S^2}\,(b_{i0} + b_{i\,N_i}\,S^{N_i})$ into the left hand side of (28) to get an expression of the form

$$\mathbf{P}(S) + \sqrt{1-S^2}\,\mathbf{Q}(S), \tag{43}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are polynomials in S.

Consider the possible balances of highest exponents in all P_i and Q_i separately. Then solve the resulting linear system(s) of $2M$ (or less) equations for the $2M$ unknown M_i and N_i. For each solution continue with step ST3.

In contrast to step S2, we no longer assume $M_i \geq 1, N_i \geq 1$. Even with some M_i or N_i zero one finds non-constant solutions $U_i(S)$. In most examples, however, the set of balance equations for the M_i and N_i is too large or the corresponding linear systems are strongly underdetermined (i.e. many leading exponents remain arbitrary) and we had to pre-select values for exponents. We set $M_i = 2$ and all $N_i = 1$, thus restricting the solutions to (at most) quadratic in S and T. See Section 11 for alternative strategies.

Example: For (41), substituting

$$U_i(S) = a_{i0} + a_{i\,M_i} S^{M_i} + \sqrt{1-S^2}\,(b_{i0} + b_{i\,N_i} S^{N_i}), \quad i = 1, 2, 3, \tag{44}$$

into (42), the highest exponents remain undetermined. Notwithstanding the limited number of terms and the low order in (42), the computation is unwieldy. Hence, we set all $M_i = 2$, $N_i = 1$, and continue with

$$U_i(S) = a_{i0} + a_{i1} S + a_{i2} S^2 + \sqrt{1-S^2}\,(b_{i0} + b_{i1} S), \quad i = 1, 2, 3. \tag{45}$$

Step ST3: Derive the algebraic system for the coefficients a_{ij} and b_{ij}

Substitute (39) into (28) gives (43) which must vanish. Hence, in $\mathbf{P} = \mathbf{0}$ and $\mathbf{Q} = \mathbf{0}$ separately, equate to zero the coefficients of the various power terms in S.

Example: For (41), after substitution of (45) into (42), the resulting nonlinear algebraic system for the coefficients a_{ij} and b_{ij} contains 25 equations (not shown).

Step ST4: Solve the nonlinear parameterized algebraic system

In contrast to step S4 we no longer assume that $a_{i\,M_i}$ and $b_{i\,N_i}$ are nonzero (at the cost of generating some constant solutions, which we reject later).

Example: For (41), there are 11 solutions. Three are trivial, leading to constant U_i. Eight are nontrivial solutions giving the results below.

Step ST5: Build and test the solitary wave solutions

Proceed as in step S5.

Example: The solitary wave solutions of (40) are:

$$\begin{aligned} u(x,t) &= \pm c_2 \tanh\xi, \\ v(x,t) &= \mp\frac{1}{2}c_2(c_1-c_2)\operatorname{sech}^2\xi, \\ w(x,t) &= -\frac{1}{2}c_2(c_1-c_2)\operatorname{sech}^2\xi, \end{aligned} \tag{46}$$

which could have been obtained with the tanh method of Section 2;

$$\begin{aligned} u(x,t) &= \pm i c_2 \operatorname{sech}\xi, \\ v(x,t) &= \pm\frac{1}{2}ic_2(c_1-c_2)\tanh\xi \operatorname{sech}\xi, \\ w(x,t) &= \frac{1}{4}c_2(c_1-c_2)\left(1-2\operatorname{sech}^2\xi\right), \end{aligned} \tag{47}$$

reported in Gao and Tian (2001); and the two new solutions

$$\begin{aligned} u(x,t) &= \pm\frac{1}{2}ic_2\left(\operatorname{sech}\xi + i\tanh\xi\right), \\ v(x,t) &= \pm\frac{1}{4}c_2(c_1-c_2)\operatorname{sech}\xi\left(\operatorname{sech}\xi + i\tanh\xi\right), \\ w(x,t) &= -\frac{1}{4}c_2(c_1-c_2)\operatorname{sech}\xi\left(\operatorname{sech}\xi + i\tanh\xi\right). \end{aligned} \tag{48}$$

The complex conjugates of (48) are also solutions. In all solutions $\xi = c_1 x + c_2 t + \Delta$, with c_1, c_2 and Δ arbitrary. Remark: for $M_i = N_i = 2$ we obtained the same results.

Steps ST1-ST5 must be repeated if any of the parameters in (3) were set to zero.

5. Cn method for nonlinear PDEs

By modifying the chain rule in step T1 (S1 or ST1), it is possible to generate polynomial solutions in other *fundamental* functions, such as the Jacobi elliptic sine (sn) and cosine (cn) functions.

In this section, we give the steps (labelled with CN) of a cn method to solve (3). At the end we show the modifications needed for a sn method.

Example: Consider the KdV equation (Ablowitz and Clarkson, 1991),

$$u_t + \alpha u u_x + u_{xxx} = 0, \tag{49}$$

(α real constant) which models waves in shallow water, ion-acoustic waves in plasmas, etc.

Step CN1: Transform the PDE into a nonlinear ODE

Similar to the strategy in T1 and S1, using (Lawden, 1989)

$$\begin{aligned} \operatorname{sn}^2(\xi;m) &= 1-\operatorname{cn}^2(\xi;m), \\ \operatorname{dn}^2(\xi;m) &= 1-m+m\operatorname{cn}^2(\xi;m), \end{aligned} \tag{50}$$

and

$$\mathrm{cn}'(\xi; m) = -\mathrm{sn}(\xi; m)\mathrm{dn}(\xi; m), \tag{51}$$

one has $\mathrm{CN}' = -\sqrt{(1-\mathrm{CN}^2)(1-m+m\,\mathrm{CN}^2)}$ where $\mathrm{CN} = \mathrm{cn}(\xi; m)$ is the Jacobi elliptic cosine with argument ξ and modulus m, subject to $0 \leq m \leq 1$.

By repeatedly applying the chain rule

$$\frac{\partial \bullet}{\partial x_j} = \frac{d\bullet}{d\mathrm{CN}}\frac{d\mathrm{CN}}{d\xi}\frac{\partial \xi}{\partial x_j} = -c_j\sqrt{(1-\mathrm{CN}^2)(1-m+m\,\mathrm{CN}^2)}\frac{d\bullet}{d\mathrm{CN}}, \tag{52}$$

one transforms (3) into a nonlinear ODE with polynomial coefficients in CN.

The modulus m becomes another parameter to be added to the c_j. Note that in many books the modulus m is denoted by k^2.

Example: Use (52) to transform (49) into

$$\begin{aligned} &\left(c_1^3(1-2m+6m\,\mathrm{CN}^2) - c_2 - \alpha c_1 U_1\right) U_1' \\ &+ 3c_1^3\mathrm{CN}(1-2m+2m\,\mathrm{CN}^2)U_1'' - c_1^3(1-\mathrm{CN}^2)(1-m+m\,\mathrm{CN}^2)U_1''' = 0 \end{aligned} \tag{53}$$

Step CN2: Determine the degree of the polynomial solutions

Follow the strategy in step T2.

Example: For (49), substituting $U_1(\mathrm{CN}) = \mathrm{CN}^{M_1}$ into (53) and equating the highest exponents gives

$$1 + M_1 = -1 + 2\,M_1. \tag{54}$$

So, $M_1 = 2$, and

$$U_1(\mathrm{CN}) = a_{10} + a_{11}\mathrm{CN} + a_{12}\mathrm{CN}^2. \tag{55}$$

Step CN3: Derive the algebraic system for the coefficients a_{ij}

Follow the strategy of step T3.

Example: For (49), after substituting (55) into (53), one finds

$$\begin{aligned} -3\,a_{11}\,c_1\,(\alpha\,a_{12} - 2\,m\,c_1^2) &= 0, \\ -2\,a_{12}\,c_1\,(\alpha\,a_{12} - 12\,m\,c_1^2) &= 0, \\ -a_{11}\,(\alpha\,a_{10}\,c_1 - c_1^3 + 2\,m\,c_1^3 + c_2) &= 0, \\ -\alpha\,a_{11}^2\,c_1 - a_{12}\,(2\,\alpha\,a_{10}\,c_1 - 16m\,c_1^3 - 8\,c_1^3 + 2\,c_2) &= 0. \end{aligned} \tag{56}$$

Step CN4: Solve the nonlinear parameterized algebraic system

Same strategy as step T4.

Example: For c_1, c_2, m and a_{12} nonzero, the solution of (56) is

$$\begin{aligned} a_{10} &= \frac{4c_1^3\,(1-2\,m) - c_2}{\alpha\,c_1}, \\ a_{11} &= 0, \\ a_{12} &= \frac{12\,m\,c_1^2}{\alpha}. \end{aligned} \tag{57}$$

Step CN5: Build and test the solitary wave solutions

Substitute the results from CN4 into (55) and reverse step CN1. Test the solutions.

Example: The cnoidal wave solution of (49) is

$$u(x,t) = \frac{4c_1^3(1-2m) - c_2}{\alpha c_1} + \frac{12m\, c_1^2}{\alpha}\,\mathrm{cn}^2(c_1 x + c_2 t + \Delta; m), \tag{58}$$

where c_1, c_2, α, Δ, and the modulus m are arbitrary.

Steps CN1-CN5 should be repeated if one or more of the (Greek) parameters in (3) are set to zero.

Without showing the steps explicitly, it should be clear that if one wants to find solutions in terms of Jacobi's sn function, one should use

$$\begin{aligned} \mathrm{cn}^2(\xi;m) &= 1 - \mathrm{sn}^2(\xi;m), \\ \mathrm{dn}^2(\xi;m) &= 1 - m\,\mathrm{sn}^2(\xi;m), \\ \mathrm{sn}'(\xi;m) &= \mathrm{cn}(\xi;m)\,\mathrm{dn}(\xi;m). \end{aligned} \tag{59}$$

Hence, $\mathrm{SN}' = \sqrt{(1-\mathrm{SN}^2)(1-m\,\mathrm{SN}^2)}$, where $\mathrm{SN} = \mathrm{sn}(\xi;m)$ is the Jacobi elliptic sine with argument ξ and modulus m $(0 \le m \le 1)$. The rest of the steps is similar to the cn case, except that one uses the chain rule

$$\frac{\partial \bullet}{\partial x_j} = \frac{d\bullet}{d\mathrm{SN}}\frac{d\mathrm{SN}}{d\xi}\frac{\partial \xi}{\partial x_j} = c_j\sqrt{(1-\mathrm{SN}^2)(1-m\,\mathrm{SN}^2)}\,\frac{d\bullet}{d\mathrm{SN}}. \tag{60}$$

Obviously, since (52) involves roots, there is no reason to restrict the solutions to polynomials in cn or sn alone. Compare with Sections 3 and 4. Solutions involving both sn and cn (or combinations with dn) are beyond the scope of this paper.

6. Tanh method for nonlinear DDEs

The tanh method can be adapted to solve polynomial nonlinear DDEs. Apart from slight, yet important modifications, the steps are as in Section 2. We did not attempt to adapt any of the sech methods from Sections 3 and 4 towards solving DDEs.

Given is a system of M polynomial DDEs of order m with constant coefficients.

$$\Delta(\cdots, \mathbf{u}_{n-1}(t), \mathbf{u}_n(t), \mathbf{u}_{n+1}(t), \cdots, \dot{\mathbf{u}}_n(t), \ddot{\mathbf{u}}_n(t), \cdots, \mathbf{u}_n^{(k)}(t), \cdots, \mathbf{u}_n^{(m)}(t)) = \mathbf{0}, \tag{61}$$

where the vector dynamical variables $\mathbf{u}_n$ (and their shifts) have M components $u_{i,n}$. Furthermore, $\mathbf{u}_n^{(k)}(t)$ is the kth order derivative in t. For simplicity of notation, in the examples in Section 8 we denote these components by u_n, v_n, w_n, etc. Any constant parameters in (61) will be denoted by lower Greek letters.

Example: To illustrate the method, we derive an exact solution of the one-dimensional Toda lattice (Toda, 1981),

$$\ddot{y}_n = \exp\left(y_{n-1} - y_n\right) - \exp\left(y_n - y_{n+1}\right), \tag{62}$$

where $y_n(t)$ is the displacement from equilibrium of the nth unit mass under an exponential decaying interaction force between nearest neighbors. With the change of variable,

$$\dot{u}_n = \exp\left(y_{n-1} - y_n\right) - 1, \tag{63}$$

(64) can be written as

$$\ddot{u}_n = \left(1 + \dot{u}_n\right)\left(u_{n-1} - 2u_n + u_{n+1}\right), \tag{64}$$

or

$$\ddot{u}_{1,n} = \left(1 + \dot{u}_{1,n}\right)\left(u_{1,n-1} - 2u_{1,n} + u_{1,n+1}\right), \tag{65}$$

with $u_{1,n}(t) = u_n(t)$.

Step D1: Transform the DDE into a nonlinear DDE in T

In a travelling frame of reference,

$$\xi = c_1 n + c_2 t + \Delta, \tag{66}$$

seek solutions $\mathbf{U}_n(T) = \mathbf{u}_n(t)$ with $T = \tanh\xi$. Repeatedly apply the chain rule,

$$\frac{d\bullet}{dt} = \frac{d\bullet}{dT}\frac{dT}{d\xi}\frac{d\xi}{dt} = c_2(1 - T^2)\frac{d\bullet}{dT}, \tag{67}$$

and transform (61) into

$$\mathbf{\Delta}(T, \cdots, \mathbf{U}_{n-1}, \mathbf{U}_n, \mathbf{U}_{n+1}, \cdots, \mathbf{U}'_n, \mathbf{U}''_n, \cdots, \mathbf{U}_n^{(m)}) = \mathbf{0}, \tag{68}$$

where $\mathbf{U}_{n\pm p}(T) = \mathbf{u}_{n\pm p}(t)$. Each DDE in (68) has polynomial coefficients in T.

Example: Substituting

$$\begin{aligned}
\dot{u}_{1,n} &= c_2(1 - T^2)U'_{1,n}, \\
\ddot{u}_{1,n} &= c_2^2(1 - T^2)[(1 - T^2)U'_{1,n}]' = c_2^2(1 - T^2)\left[-2TU'_{1,n} + (1 - T^2)U''_{1,n}\right],
\end{aligned} \tag{69}$$

into (64) yields

$$c_2^2(1-T^2)\left[2TU'_{1,n} - (1-T^2)U''_{1,n}\right] + \left[1 + c_2(1-T^2)U'_{1,n}\right]\left[U_{1,n-1} - 2U_{1,n} + U_{1,n+1}\right] = 0. \tag{70}$$

Note that (65) is free of time derivatives of shifts of $u_{1,n}$. If these were present, our method would no longer work since T depends on n. Indeed, $T = T_n = \tanh(c_1 n + c_2 t + \Delta)$ and (67) cannot be applied to $u_{i,n\pm p}$. In the next two steps we explicitly write T_n for T. Also, in several instances we will use

$$\tanh(x + y) = \frac{\tanh x + \tanh y}{1 + \tanh x \tanh y}. \tag{71}$$

Step D2: Determine the degree of the polynomial solutions
Seek solutions

$$U_{i,n}(T_n) = \sum_{j=0}^{M_i} a_{ij}\, T_n^j, \tag{72}$$

and first compute the leading exponents M_i. Substitute $U_{i,n} = T_n^{M_i}$, and

$$U_{i,n\pm p}\,(T_n) = [T_{n\pm p}]^{M_i} = \left[\frac{T_n \pm \tanh(pc_1)}{1 \pm T_n \tanh(pc_1)}\right]^{M_i}, \tag{73}$$

where (71) was used to accommodate for the shifts, into the left hand side of (68). In each component, balance the highest power terms in T_n and solve the resulting linear system(s) for unknowns M_i. Note that $U_{i,n\pm p}\,(T_n)$ can be interpreted as being of degree zero in T_n. Consequently, upon clearing denominators, these terms do not affect the computation of M_i.

Example: For (70), one readily obtains

$$2M_1 + 1 = M_1 + 2. \tag{74}$$

Thus, $M_1 = 1$, and $U_{1,n}(T_n) = a_{10} + a_{11} T_n$.

Step D3: Derive the algebraic system for the coefficients a_{ij}
Substitute (72), and

$$U_{i,n\pm p}\,(T_n) = \sum_{j=0}^{M_i} a_{ij}\, \left[T_{n\pm p}\right]^j = \sum_{j=0}^{M_i} a_{ij}\, \left[\frac{T_n \pm \tanh(pc_1)}{1 \pm T_n \tanh(pc_1)}\right]^j \tag{75}$$

into (68), clear denominators, and equate to zero the coefficients of the various power terms in T_n. Express $\tanh(pc_1)$ in powers of $\tanh(c_1)$ via repeated application of (71). The resulting nonlinear algebraic system for the unknowns a_{ij}, is parameterized by $\tanh(c_1)$ and c_2 and any parameters (in lower Greek letters) in (61).

Example: Substituting

$$\begin{aligned} U_n(T_n) &= a_{10} + a_{11} T_n, \\ U_{n\pm1}(T_n) &= a_{10} + a_{11} T_{n\pm1} = a_{10} + a_{11} \frac{T_n \pm \tanh(c_1)}{1 \pm T_n \tanh(c_1)}, \end{aligned} \tag{76}$$

into (70), cancelling common factor $T_n(1 - T_n^2)$, and setting coefficients of power terms in T_n to zero, yields

$$\begin{aligned} &a_{11}(c_2 - a_{11}) = 0, \\ &a_{11}(c_2^2 - \tanh^2(c_1) - a_{11} c_2 \tanh^2(c_1)) = 0. \end{aligned} \tag{77}$$

Step D4: Solve the nonlinear parameterized system

In contrast with step T4 in Section 2, the algebraic system has unknowns a_{ij}, and parameters $\alpha, \beta, \cdots, c_2$ and $\tanh(c_1)$, instead of c_1. Assume again that all parameters and the coefficients $a_{i\,M_i}$ (of the highest power terms in T_n) are nonzero.

Example: Assuming c_1, c_2, and a_{11} nonzero, the solution of (77) is

$$\begin{aligned} a_{10} &= \text{arbitrary}, \\ a_{11} &= c_2 = \pm\sinh(c_1). \end{aligned} \tag{78}$$

Step D5: Build and test the solitary wave solutions

Substitute the solutions obtained in step D4 into (72) and reverse step D1. Finally, test the solutions (in the original variables) by substitution into (61). If so desired, reject complex solutions.

Example: The closed form solution of (64) reads

$$u_n(t) = a_{10} \pm \sinh(c_1)\tanh[c_1 n \pm \sinh(c_1)\, t + \Delta], \tag{79}$$

where a_{10}, c_1 and Δ are arbitrary. Steps D1-D5 must be repeated if any of the parameters in (61) were set to zero.

7. Examples of solitary wave solutions for PDEs

We implemented the methods from Sections 2-5 in our comprehensive *Mathematica* package **PDESpecialSolutions.m** and used the code to solve the equations in subsections 7.1-7.8.

7.1. The Zakharov-Kuznetsov KdV-type equations

The KdV-Zakharov-Kuznetsov (KdV-ZK) equation,

$$u_t + \alpha u u_x + u_{xxx} + u_{xyy} + u_{xzz} = 0, \tag{80}$$

models ion-acoustic waves in magnetized multi-component plasmas including negative ions (see e.g. Das and Verheest, 1989),

With *PDESpecialSolutions* (Tanh and Sech options) we found the solution

$$\begin{aligned} u(x,y,z,t) &= \frac{8c_1(c_1^2+c_2^2+c_3^2)-c_4}{\alpha c_1} - \frac{12(c_1^2+c_2^2+c_3^2)}{\alpha}\tanh^2\xi, \\ &= -\frac{4c_1(c_1^2+c_2^2+c_3^2)+c_4}{\alpha c_1} + \frac{12(c_1^2+c_2^2+c_3^2)}{\alpha}\operatorname{sech}^2\xi, \end{aligned} \tag{81}$$

where $\xi = c_1 x + c_2 y + c_3 z + c_4 t + \Delta$, with $c_1, c_2, c_3, c_4, \Delta$ and α arbitrary.

For $c_2 = c_3 = 0$ and replacing c_4 by c_2, one gets the well-known solitary wave solution

$$\begin{aligned} u(x,t) &= \frac{8c_1^3 - c_2}{\alpha c_1} - \frac{12c_1^2}{\alpha}\tanh^2(c_1 x + c_2 t + \Delta), \\ &= -\frac{4c_1^3 + c_2}{\alpha c_1} + \frac{12c_1^2}{\alpha}\text{sech}^2(c_1 x + c_2 t + \Delta), \end{aligned} \tag{82}$$

of the ubiquitous KdV equation (49).

At this point, the function *PDESpecialSolutions* accepts the options Tanh (default), Sech, SechTanh, JacobiCN and JacobiSN, but does not take boundary or initial conditions as input. One can *aposteriori* impose conditions to e.g. (82), like $\lim_{x\to\pm\infty} u(x,t) = 0$, which fixes $c_2 = -4c_1^3$.

For the modified KdV-ZK equation (Das and Verheest, 1989),

$$u_t + \alpha u^2 u_x + u_{xxx} + u_{xyy} + u_{xzz} = 0, \tag{83}$$

using the Tanh and Sech options, the *PDESpecialSolutions* returns

$$u(x,y,z,t) = \pm i\sqrt{\frac{6(c_1^2 + c_2^2 + c_3^2)}{\alpha}}\tanh\xi, \tag{84}$$

with $\xi = c_1 x + c_2 y + c_3 z + 2c_1(c_1^2 + c_2^2 + c_3^2)t + \Delta$, and

$$u(x,y,z,t) = \pm\sqrt{\frac{6(c_1^2 + c_2^2 + c_3^2)}{\alpha}}\text{sech}\,\xi, \tag{85}$$

with $\xi = c_1 x + c_2 y + c_3 z - c_1(c_1^2 + c_2^2 + c_3^2)t + \Delta$. For $c_2 = c_3 = 0$, (84) and (85) reduce to the well-known solitary wave solutions

$$u(x,t) = \pm i c_1\sqrt{\frac{6}{\alpha}}\tanh(c_1 x + 2c_1^3 t + \Delta). \tag{86}$$

and

$$u(x,t) = \pm c_1\sqrt{\frac{6}{\alpha}}\text{sech}(c_1 x - c_1^3 t + \Delta). \tag{87}$$

(c_1, Δ and α arbitrary real numbers) of the modified KdV (mKdV) equation (Ablowitz and Clarkson, 1991),

$$u_t + \alpha u^2 u_x + u_{xxx} = 0. \tag{88}$$

7.2. The generalized Kuramoto-Sivashinsky equation

Consider the generalized Kuramoto-Sivashinsky (KS) equation (see e.g. Parkes and Duffy, 1996):

$$u_t + uu_x + u_{xx} + \alpha u_{xxx} + u_{xxxx} = 0. \tag{89}$$

Ignoring complex solutions, *PDESpecialSolutions* with the Tanh option automatically determines the special values of the real parameter α and the corresponding closed form solutions. For $\alpha = 4$,

$$u(x,t) = 9 - 2c_2 - 15\tanh\xi - 15\tanh^2\xi + 15\tanh^3\xi, \tag{90}$$

with $\xi = \frac{1}{2}x + c_2 t + \Delta$. For $\alpha = \frac{12}{\sqrt{47}}$,

$$u(x,t) = \frac{45 \mp 4418c_2}{47\sqrt{47}} \pm \frac{45}{47\sqrt{47}}\tanh\xi - \frac{45}{47\sqrt{47}}\tanh^2\xi \pm \frac{15}{47\sqrt{47}}\tanh^3\xi, \tag{91}$$

where $\xi = \pm\frac{1}{2\sqrt{47}}x + c_2 t + \Delta$. For $\alpha = \frac{16}{\sqrt{73}}$,

$$u(x,t) = \frac{2(30 \mp 5329c_2)}{73\sqrt{73}} \pm \frac{75}{73\sqrt{73}}\tanh\xi - \frac{60}{73\sqrt{73}}\tanh^2\xi \pm \frac{15}{73\sqrt{73}}\tanh^3\xi, \tag{92}$$

where $\xi = \pm\frac{1}{2\sqrt{73}}x + c_2 t + \Delta$.

The remaining solutions produced by *PDESpecialSolutions* are either complex (not shown here) or can be obtained from the above via the inversion symmetry of (89): $u \to -u$, $x \to -x$, $\alpha \to -\alpha$.

A separate run of the code after setting $\alpha = 0$ in (89) yields

$$u(x,t) = -2\sqrt{\frac{19}{11}}c_2 - \frac{135}{19}\sqrt{\frac{11}{19}}\tanh\xi + \frac{165}{19}\sqrt{\frac{11}{19}}\tanh^3\xi, \tag{93}$$

with $\xi = \frac{1}{2}\sqrt{\frac{11}{19}}x + c_2 t + \Delta$. In all solutions above c_2 is arbitrary.

7.3. Coupled KdV equations

The coupled KdV (cKdV) equations (Hirota-Satsuma system),

$$\begin{aligned} u_t - \alpha(6uu_x + u_{xxx}) + 2\beta vv_x &= 0, \\ v_t + 3uv_x + v_{xxx} &= 0, \end{aligned} \tag{94}$$

with real parameters α, β was solved in e.g. Hereman (1991).

Searching for sech solutions, in step S2 one must consider all balances of the potentially highest exponents in S. After transforming (94) into ODEs for $U(S)$ and $V(S)$, one substitutes of $U(S) = S^{M_1}, V(S) = S^{M_2}$. From the second equation one gets $M_1 + M_2 - 1 = 1 + M_2$. Hence, $M_1 = 2$. The maximal exponents coming from the first equation are $2M_1 - 1$ (from the term uu_x), $M_1 + 1$ (from

u_{xxx}), and $2M_2 - 1$ (from vv_x). Using $M_1 = 2$, two cases emerge: (i) all three exponents are equal, in which case $M_2 = 2$, or (ii) the third exponent is less than the first two (equal) exponents, i.e. $2M_2 - 1 < 3$, thus, $M_2 = 1$. So,

$$U(S) = a_{10} + a_{11}S + a_{12}S^2, \quad V(S) = a_{20} + a_{21}S + a_{22}S^2, \tag{95}$$

if $M_1 = M_2 = 2$, or

$$U(S) = a_{10} + a_{11}S + a_{12}S^2, \quad V(S) = a_{20} + a_{21}S, \tag{96}$$

if $M_1 = 2, M_2 = 1$.

PDESpecialSolutions (Sech option) automatically computes the two choices for the leading exponents automatically and produces

$$\begin{aligned} u(x,t) &= -\frac{c_2 + 4c_1^3}{3c_1} + 4c_1^2 \operatorname{sech}^2(c_1x + c_2t + \Delta), \\ v(x,t) &= \pm\frac{4\alpha c_1^3 + c_2 + 2\alpha c_2}{\sqrt{6\alpha\beta}c_1} \mp \frac{2\sqrt{6\alpha}c_1^2}{\sqrt{\beta}} \operatorname{sech}^2(c_1x + c_2t + \Delta), \end{aligned} \tag{97}$$

and

$$\begin{aligned} u(x,t) &= -\frac{c_2 + c_1^3}{3c_1} + 2c_1^2 \operatorname{sech}^2(c_1x + c_2t + \Delta), \\ v(x,t) &= \pm\frac{\sqrt{4\alpha c_1^4 - 2(1+2\alpha)c_1c_2}}{\sqrt{\beta}} \operatorname{sech}(c_1x + c_2t + \Delta), \end{aligned} \tag{98}$$

with $c_1, c_2, \alpha, \beta, \Delta$ arbitrary. These solutions comprise the two special solutions in Hereman (1991).

Another coupled system of KdV-like equations was studied by Guha-Roy (see e.g. Coffey (2001)),

$$\begin{aligned} u_t + \alpha vv_x + \beta uu_x + \gamma u_{xxx} &= 0, \\ v_t + \delta(uv)_x + \epsilon vv_x &= 0, \end{aligned} \tag{99}$$

where α through ϵ are real constants. The software computed:

$$\begin{aligned} u(x,t) &= -\frac{4\epsilon^2\gamma c_1^3 + (4\alpha\delta + \epsilon^2)c_2}{Ac_1} + \frac{12\epsilon^2\gamma c_1^2}{A} \operatorname{sech}^2(c_1x + c_2t + \Delta), \\ v(x,t) &= \frac{2\epsilon[4\delta\gamma c_1^3 + (\delta - \beta)c_2]}{Ac_1} - \frac{24\delta\epsilon\gamma c_1^2}{A} \operatorname{sech}^2(c_1x + c_2t + \Delta), \end{aligned} \tag{100}$$

where $A = 4\alpha\delta^2 + \beta\epsilon^2$, with c_1, c_2, Δ and α through ϵ arbitrary. For $\epsilon = 0$, (99) reduces to Kawamoto's system; for $\epsilon = 0, \delta = -2$ to Ito's system. Neither of these special cases has polynomial solutions in sech or tanh.

7.4. The Fisher and FitzHugh-Nagumo equations

For the Fisher equation (Malfliet, 1994),

$$u_t - u_{xx} - u(1-u) = 0, \tag{101}$$

with *PDESpecialSolutions* (Tanh option) we found the (real) solution

$$u(x,t) = \frac{1}{4} \pm \frac{1}{2}\tanh\xi + \frac{1}{4}\tanh^2\xi, \tag{102}$$

with $\xi = \pm\frac{1}{2\sqrt{6}}x \pm \frac{5}{12}t + \Delta$. In addition, there are 4 complex solutions.

Obviously, the function *PDESpecialSolutions* can handle ODEs. For example, we can put the FitzHugh-Nagumo (FHN) equation (Hereman, 1990),

$$u_t - u_{xx} + u(1-u)(\alpha - u) = 0, \tag{103}$$

where $-1 \leq \alpha < 1$, into a travelling frame,

$$\beta v_z + \sqrt{2}v_{zz} - \sqrt{2}v(1-\sqrt{2}v)(\alpha - \sqrt{2}v) = 0, \tag{104}$$

with $v(z) = v(x - \frac{\beta}{\sqrt{2}}t) = \sqrt{2}u(x,t)$, Ignoring the inversion symmetry $z \to -z$, $\beta \to -\beta$ of (104), we find with *PDESpecialSolutions* (Tanh option)

$$v(z) = \frac{1}{2\sqrt{2}}\left[\beta + (\beta-2)\tanh[\frac{\sqrt{2}}{4}(2-\beta)z + \Delta]\right], \tag{105}$$

if $\alpha = \beta - 1$;

$$v(z) = \frac{(\beta+2)}{2\sqrt{2}}\left[1 - \tanh[\frac{\sqrt{2}}{4}(\beta+2)z + \Delta]\right], \tag{106}$$

if $\alpha = \beta + 2$; and

$$v(z) = \frac{1}{2\sqrt{2}}\left[1 + \tanh[\frac{\sqrt{2}}{4}z + \Delta]\right], \tag{107}$$

if $\alpha = \frac{1}{2}(\beta+1)$. In these solutions (see e.g. Hereman (1990)) β and Δ are arbitrary.

7.5. A degenerate Hamiltonian system

Gao and Tian (2001) considered the following degenerate Hamiltonian system,

$$\begin{aligned} &u_t - u_x - 2v = 0, \\ &v_t - 2\epsilon\, uv = 0, \quad \epsilon = \pm 1, \end{aligned} \tag{108}$$

which was shown to be completely integrable by admitting infinitely many conserved densities.

PDESpecialSolutions (Sech option) transforms (108) into

$$\begin{aligned} &(c_1 - c_2)S\sqrt{1-S^2}\,U_1' - 2U_2 = 0, \\ &c_2 S\sqrt{1-S^2}\,U_2' + 2\epsilon\, U_1 U_2 = 0, \end{aligned} \tag{109}$$

with $U_1(S) = u_1(x_1, x_2) = u(x,t)$ and $U_2(S) = u_2(x_1, x_2) = v(x,t)$. The software did not find polynomial solutions in sech based on the method of Section 3.

With the SechTanh option, *PDESpecialSolutions* applies the mixed sech-tanh method of Section 4 and seeks

$$U_i(S) = a_{i0} + a_{i1}S + a_{i2}S^2 + \sqrt{1-S^2}\,(b_{i0} + b_{i1}S), \quad i = 1, 2. \tag{110}$$

After solving 16 equations for the coefficients a_{ij}, b_{ij}, the software returns three non-trivial solutions:

$$\begin{aligned} u(x,t) &= -\epsilon c_2 \tanh\xi, \\ v(x,t) &= \frac{1}{2}\epsilon\, c_2(c_1 - c_2)\mathrm{sech}^2\xi, \end{aligned} \tag{111}$$

which could have been obtained with the tanh method of Section 2;

$$\begin{aligned} u(x,t) &= \frac{1}{2} i c_2 \epsilon\ (\mathrm{sech}\,\xi + i\tanh\xi), \\ v(x,t) &= \frac{1}{4} c_2(c_1 - c_2)\epsilon\ \mathrm{sech}\,\xi\,(\mathrm{sech}\,\xi + i\tanh\xi), \end{aligned} \tag{112}$$

plus the two complex conjugates. There are no constraints on c_1, c_2, and ϵ. All six solutions with $\xi = c_1 x + c_2 t + \Delta$ were known (see e.g. Gao and Tian (2001)).

7.6. The combined KdV-mKdV equation

The combined KdV-mKdV equation (see Gao and Tian, 2001)

$$u_t + 6\alpha u u_x + 6\beta u^2 u_x + \gamma u_{xxx} = 0, \tag{113}$$

describes a variety of wave phenomena in plasma, solid state, and quantum physics. We chose this example to show that ODEs of type (29), which are free of $\sqrt{1-S^2}$, can admit mixed sech-tanh solutions.

First, *PDESpecialSolutions* with the Tanh option, produces

$$u(x,t) = -\frac{\alpha}{2\beta} \pm i\sqrt{\frac{\gamma}{\beta}} c_1 \tanh(c_1 x + \frac{c_1}{2\beta}(3\alpha^2 + 4\beta\gamma c_1^2)t + \Delta). \tag{114}$$

Next, with the Sech option, *PDESpecialSolutions* transforms (113) into an ODE of type (29). Using the algorithm of Section 3, the software computes

$$u(x,t) = -\frac{\alpha}{2\beta} \pm \sqrt{\frac{\gamma}{\beta}} c_1 \mathrm{sech}[c_1 x + \frac{c_1}{2\beta}(3\alpha^2 - 2\beta\gamma c_1^2)t + \Delta]. \tag{115}$$

Third, with the SechTanh option, *PDESpecialSolutions* applies the method of Section 4. Seeking solutions

$$U_1(S) = a_{10} + a_{11}S + a_{12}S^2 + \sqrt{1-S^2}\,(b_{10} + b_{11}S). \tag{116}$$

the code returns

$$u(x,t) = -\frac{\alpha}{2\beta} + \frac{1}{2}\sqrt{\frac{\gamma}{\beta}}c_1\,(\operatorname{sech}\xi \pm i\tanh\xi), \tag{117}$$

and

$$u(x,t) = -\frac{\alpha}{2\beta} - \frac{1}{2}\sqrt{\frac{\gamma}{\beta}}c_1\,(\operatorname{sech}\xi \mp i\tanh\xi), \tag{118}$$

where $\xi = c_1 x + \frac{1}{2}\frac{c_1}{\beta}(3\alpha^2 + \beta\gamma c_1^2)t + \Delta$. In all solutions c_1, Δ, and the Greek parameters are arbitrary. The solutions were reported in Gao and Tian (2001), although there were minor misprints.

7.7. A class of fifth-order PDEs with three parameters

To illustrate the limitations of our software, consider the family of fifth-order KdV equations (Göktaş and Hereman, 1997),

$$u_t + \alpha u^2 u_x + \beta u_x u_{xx} + \gamma u u_{xxx} + u_{xxxxx} = 0, \tag{119}$$

where α, β, and γ are nonzero parameters.

An investigation of the scaling properties of (119) reveals that only the ratios $\frac{\alpha}{\gamma^2}$ and $\frac{\beta}{\gamma}$ are important, but let us proceed with (119).

Special cases

Several special cases of (119) are well known (for references see Göktaş and Hereman (1997)). Indeed, for $\alpha = 30, \beta = 20, \gamma = 10$, Eq. (119) reduces to

$$u_t + 30u^2 u_x + 20u_x u_{xx} + 10u u_{xxx} + u_{xxxxx} = 0, \tag{120}$$

which belongs to the completely integrable hierarchy of higher-order KdV equations constructed by Lax. Equation (120) has two tanh solutions:

$$u(x,t) = 4c_1^2 - 6c_1^2\tanh^2(c_1 x - 56c_1^5 t + \Delta), \tag{121}$$

and

$$u(x,t) = a_{10} - 2c_1^2\tanh^2[c_1 x - 2(15a_{10}^2 c_1 - 40a_{10}c_1^3 + 28c_1^5)t + \Delta], \tag{122}$$

where a_{10}, c_1, Δ are arbitrary.

For $\alpha = \beta = \gamma = 5$, one gets the equation,

$$u_t + 5u^2 u_x + 5u_x u_{xx} + 5u u_{xxx} + u_{xxxxx} = 0, \tag{123}$$

due to Sawada and Kotera (SK) and also Dodd and Gibbon, which has two tanh solutions:

$$u(x,t) = 8c_1^2 - 12c_1^2 \tanh^2(c_1 x - 16c_1^5 t + \Delta), \tag{124}$$

and

$$u(x,t) = a_{10} - 6c_1^2 \tanh^2[c_1 x - (5a_{10}^2 c_1 - 40a_{10}c_1^3 + 76c_1^5)t + \Delta], \tag{125}$$

where a_{10}, c_1, Δ are arbitrary.

The KK equation due to Kaup and Kupershmidt,

$$u_t + 20u^2 u_x + 25u_x u_{xx} + 10uu_{xxx} + u_{xxxxx} = 0, \tag{126}$$

corresponding to $\alpha = 20, \beta = 25, \gamma = 10$, and again admits two tanh solutions:

$$u(x,t) = c_1^2 - \frac{3}{2}c_1^2 \tanh^2(c_1 x - c_1^5 t + \Delta), \tag{127}$$

and

$$u(x,t) = 8c_1^2 - 12c_1^2 \tanh^2(c_1 x - 176c_1^5 t + \Delta), \tag{128}$$

with c_1, Δ arbitrary, but no additional arbitrary coefficient.

The equation

$$u_t + 2u^2 u_x + 6u_x u_{xx} + 3uu_{xxx} + u_{xxxxx} = 0, \tag{129}$$

for $\alpha = 2, \beta = 6, \gamma = 3$, was studied by Ito. It has one tanh solution:

$$u(x,t) = 20c_1^2 - 30c_1^2 \tanh^2(c_1 x - 96c_1^5 t + \Delta), \tag{130}$$

again with c_1 and Δ arbitrary constants. *PDESpecialSolutions* (Tanh option) easily produces all these solutions.

General case

Eq. (119) is hard to analyze by hand or with the computer. After a considerably amount of time, *PDESpecialSolutions* with the Tanh option, produced the solutions given below (but not in as nice a form). Our write-up of the solutions is the result of additional interactive work with *Mathematica*.

The coefficients a_{10}, a_{11}, and a_{12} in

$$u(x,t) = a_{10} + a_{11}\tanh(c_1 x + c_2 t + \Delta) + a_{12}\tanh^2(c_1 x + c_2 t + \Delta), \tag{131}$$

must satisfy the following nonlinear algebraic system with parameters c_1, c_2, α, β, and γ :

$$\begin{aligned}
&a_{11}\left(\alpha a_{12}^2 + 2\beta a_{12}c_1^2 + 6\gamma a_{12}c_1^2 + 24c_1^4\right) = 0,\\
&a_{11}\left(\alpha a_{11}^2 + 6\alpha a_{10}a_{12} + 6\gamma a_{10}c_1^2 - 12\beta a_{12}c_1^2 - 18\gamma a_{12}c_1^2 - 120c_1^4\right) = 0,\\
&\alpha a_{12}^2 + 6\beta a_{12}c_1^2 + 12\gamma a_{12}c_1^2 + 360c_1^4 = 0,\\
&2\alpha a_{11}^2 a_{12} + 2\alpha a_{10}a_{12}^2 + \beta a_{11}^2 c_1^2 + 3\gamma a_{11}^2 c_1^2 + 12\gamma a_{10}a_{12}c_1^2\\
&\qquad -8\beta a_{12}^2 c_1^2 - 8\gamma a_{12}^2 c_1^2 - 480a_{12}c_1^4 = 0,\\
&a_{11}\left(\alpha a_{10}^2 c_1 - 2\gamma a_{10}c_1^3 + 2\beta a_{12}c_1^3 + 16c_1^5 + c_2\right) = 0,\\
&\alpha a_{10}a_{11}^2 c_1 + \alpha a_{10}^2 a_{12}c_1 - \beta a_{11}^2 c_1^3 - \gamma a_{11}^2 c_1^3 - 8\gamma a_{10}a_{12}c_1^3 + 2\beta a_{12}^2 c_1^3\\
&\qquad +136a_{12}c_1^5 + a_{12}c_2 = 0.
\end{aligned} \tag{132}$$

Assuming nonzero $a_{12}, c_1, c_2, \alpha, \beta$, and γ, two cases must be distinguished:

CASE 1: $a_{11} = 0$.
In turn, this case splits into two sub-cases.
CASE 1a:

$$a_{11} = 0, \quad a_{12} = -\frac{3}{2}a_{10}, \quad c_2 = c_1^3(24c_1^2 - \beta a_{10}), \tag{133}$$

where a_{10} must be one of the roots of

$$\alpha a_{10}^2 - 4\beta a_{10}c_1^2 - 8\gamma a_{10}c_1^2 + 160c_1^4 = 0. \tag{134}$$

CASE 1b:

$$\begin{aligned} a_{11} &= 0, \quad a_{12} = -\frac{6\gamma}{\alpha}c_1^2, \\ c_2 &= -\frac{1}{\alpha}(\alpha^2 a_{10}^2 c_1 - 8\alpha\gamma a_{10}c_1^3 + 16\alpha c_1^5 + 12\gamma^2 c_1^5), \end{aligned} \tag{135}$$

provided that

$$\beta = \frac{1}{\gamma}(10\alpha - \gamma^2), \tag{136}$$

CASE 2: $a_{11} \neq 0$.
Then

$$\alpha = \frac{1}{392}(8\beta^2 + 38\beta\gamma + 39\gamma^2), \tag{137}$$

and

$$a_{12} = -\frac{168}{2\beta + 3\gamma}c_1^2, \tag{138}$$

provided β is one of the roots of

$$(104\beta^2 + 886\beta\gamma + 1487\gamma^2)(520\beta^3 + 2158\beta^2\gamma - 1103\beta\gamma^2 - 8871\gamma^3) = 0 \tag{139}$$

Thus, case 2 also splits into two sub cases:
CASE 2a
If $\beta^2 = -\frac{1}{104}(886\beta\gamma + 1487\gamma^2)$, then

$$\begin{aligned} \alpha &= -\frac{1}{26}(2\beta + 5\gamma)\gamma, \\ a_{10} &= -\frac{52(4378\beta + 9983\gamma}{7\gamma(958\beta + 2213\gamma)}c_1^2, \\ a_{11} &= \pm\frac{336}{2\beta + 3\gamma}c_1^2, \\ a_{12} &= -\frac{168}{2\beta + 3\gamma}c_1^2, \\ c_2 &= -\frac{364(1634\beta + 3851\gamma)}{2946\beta + 6715\gamma}c_1^5. \end{aligned} \tag{140}$$

where β is any root of $104\beta^2 + 886\beta\gamma + 1487\gamma^2 = 0$.

CASE 2b

If $\beta^3 = \frac{1}{520}(1103\beta\gamma^2 + 8871\gamma^3 - 2158\beta^2\gamma)$, then

$$\begin{aligned}
\alpha &= \frac{1}{392}(8\beta^2 + 38\beta\gamma + 39\gamma^2),\\
a_{10} &= \frac{28(1066\beta^2 + 5529\beta\gamma + 6483\gamma^2)}{(2\beta+3\gamma)(6\beta+23\gamma)(26\beta+81\gamma)}c_1^2,\\
a_{11}^2 &= \frac{28224(26\beta-17\gamma)(4\beta-\gamma)}{(2\beta+3\gamma)^2(6\beta+23\gamma)(26\beta+81\gamma)}c_1^4,\\
a_{12} &= -\frac{168}{2\beta+3\gamma}c_1^2,\\
c_2 &= -\frac{8(188900114\beta^2 + 1161063881\beta\gamma + 1792261977\gamma^2)}{105176786\beta^2 + 632954969\beta\gamma + 959833473\gamma^2}c_1^5.
\end{aligned} \tag{141}$$

where β is any root of $520\beta^3 + 2158\beta^2\gamma - 1103\beta\gamma^2 - 8871\gamma^3 = 0$.

7.8. The Duffing Equation

Duffing's equation (Lawden, 1989),

$$u'' + u + \alpha u^3 = 0, \tag{142}$$

models a nonlinear spring problem. Its cn and sn solutions

$$\begin{aligned}
u(x) &= \pm\frac{\sqrt{c_1^2-1}}{\sqrt{\alpha}}\,\mathrm{cn}(c_1x+\Delta; \frac{c_1^2-1}{2c_1^2}),\\
u(x) &= \pm\frac{\sqrt{2(c_1^2-1)}}{\sqrt{\alpha}}\,\mathrm{sn}(c_1x+\Delta; \frac{1-c_1^2}{c_1^2}),
\end{aligned} \tag{143}$$

are directly produced by *PDESpecialSolutions* with the JacobiCN and JacobiSN options. Note that the modulus m is fixed in terms of the arbitrary c_1. Since $0 \le m \le 1$, the cn solution is valid when $\alpha > 0,\ c_1^2 \ge 1$; the sn solution holds if $\alpha < 0,\ \frac{1}{2} \le c_1^2 \le 1$. Such conditions are not automatically produced by the software. In simplifications the software assumes that $\alpha > 0$ (see Section 11 for details).

8. Examples of solitary wave solutions for DDEs

In addition to **PDESpecialSolutions.m**, which symbolically solves PDEs, we created the package **DDESpecialSolutions.m** for solving DDEs. The key function in the latter package is *DDESpecialSolutions* and it uses the tanh method of Section 6. We now present two additional examples solved with *DDESpecialSolutions*.

8.1. The relativistic Toda equation

For the relativistic Toda equation (Suris, 1998),

$$\begin{aligned} \dot{u}_n &= (1+\alpha u_n)(v_n - v_{n-1}), \\ \dot{v}_n &= v_n(u_{n+1} - u_n + \alpha v_{n+1} - \alpha v_{n-1}), \end{aligned} \tag{144}$$

with *DDESpecialSolutions* we found

$$\begin{aligned} u_n(t) &= -\frac{1}{\alpha} - c_2 \coth(c_1) + c_2 \tanh(c_1 n + c_2 t + \Delta), \\ v_n(t) &= \frac{c_2}{\alpha}\coth(c_1) - \frac{c_2}{\alpha}\tanh(c_1 n + c_2 t + \Delta), \end{aligned} \tag{145}$$

with c_1, c_2, Δ and α arbitrary.

8.2. The Volterra lattice

For the Volterra lattice (Suris, 1998),

$$\begin{aligned} \dot{u}_n &= u_n(v_n - v_{n-1}), \\ \dot{v}_n &= v_n(u_{n+1} - u_n), \end{aligned} \tag{146}$$

we obtained with *DDESpecialSolutions*

$$\begin{aligned} u_n(t) &= -c_2 \coth(c_1) + c_2 \tanh(c_1 n + c_2 t + \Delta), \\ v_n(t) &= -c_2 \coth(c_1) - c_2 \tanh(c_1 n + c_2 t + \Delta), \end{aligned} \tag{147}$$

with c_1, c_2, Δ arbitrary. To our knowledge (147) does not appear in the literature.

9. Other perspectives and potential generalizations

Several researchers, including Fan (2000, 2001) and Gao and Tian (2001), have used variants of the tanh method to construct particular solutions to nonlinear PDEs from mathematical physics. Lacking explicit reference to the use of software packages, most likely the computations were done interactively with *Mathematica*.

Instead of introducing nonlinear ODEs as in steps T1, S1 and ST1, Fan seeks solutions

$$u_i(x,t) = U_i(\xi) = \sum_{j=1}^{M_i} a_{ij} w(\xi)^j, \quad \xi = c_1 x + c_2 t + \Delta, \tag{148}$$

constrained by the Riccati equation,

$$w'(\xi) = b + w^2(\xi), \quad b \ \text{ constant}, \tag{149}$$

which has the following explicit solutions

$$\begin{aligned} w(\xi) &= -a\ \tanh(a\xi + c), \quad \text{if } \ b = -a^2, \\ w(\xi) &= a\ \tan(a\xi + c), \quad \text{if } \ b = a^2, \\ w(\xi) &= \frac{1}{c-\xi}, \quad \text{if } \ b = 0. \end{aligned} \tag{150}$$

The integration constant c gets absorbed in Δ, and the constant a (or b) is an extra parameter in the nonlinear algebraic system for the a_{ij}. Ignoring the case $b = 0$, the solutions (148) are *apriori* polynomials in $\tanh\xi$ or $\tan\xi$.

Gao and Tian (2001) seek solutions of type

$$u_i(x,t) = \sum_{j=0}^{M_i} a_{ij}(x,t)\tanh^j\Psi(x,t) + \sum_{j=0}^{N_i} b_{ij}(x,t)\operatorname{sech}\Psi(x,t)\tanh^j\Psi(x,t), \quad (151)$$

where $\Psi(x,t)$ is not necessarily linear in x and/or t. Of course, (151) arises from recasting the terms in (38) in a slightly different way than (39). Restricted to travelling waves, $\Psi(x,t) = c_1 x + c_2 t + \Delta$, both forms are equivalent.

Assuming (151), Gao and Tian (2001) found general mixed solutions where some u_i have only odd powers of either sech or tanh, or products of both. The coefficients in these solutions depend on derivatives of $\Psi(x,t)$; sometimes they must be solutions of PDEs in $\Psi(x,t)$.

Other types of solutions could be computed by modifying the chain rule in step T1 (S1, ST1, or CN1). Indeed, it suffices to know the underlying first-order differential equation for the desired fundamental function in the polynomial solution. For example, for solutions in $y(\xi) = \tanh\xi$, one has $y' = 1-y^2$; for $y(\xi) = \operatorname{sech}\xi$, one has $y' = -y\sqrt{1-y^2}$. Polynomial solutions in $y(\xi) = \tan\xi$ and $y(\xi) = \exp(\xi)$ solve $y' = 1+y^2$ and $y' = y$, respectively. The Jacobi elliptic functions $y(\xi) = \operatorname{cn}(\xi;m)$ and $y(\xi) = \operatorname{sn}(\xi;m)$ satisfy $y' = -\sqrt{(1-y^2)(1-m+m\,y^2)}$ and $y' = \sqrt{(1-y^2)(1-m\,y^2)}$, respectively.

10. Using the software

We illustrate the use of the packages **PDESpecialSolutions.m** and **DDESpecialSolutions.m** on a PC. Users should have access to *Mathematica* v. 3.0 or higher.

Put the packages in a directory, say myDirectory, on drive C. Start a *Mathematica* notebook session and execute the commands:

```
In[1] = SetDirectory["c:\\myDirectory"];  (* specify the directory *)

In[2] = <<PDESpecialSolutions.m         (* read in the package *)

In[3] = PDESpecialSolutions[
          {D[u[x,t],t]+alpha*u[x,t]*D[u[x,t],x]+D[u[x,t],{x,3}]==0},
          {u[x,t]}, {x,t}, {alpha}, Form -> Sech];

In[4] = <<DDESpecialSolutions.m         (* read in the package *)

In[5] = DDESpecialSolutions[{D[u[n,t],t]-u[n,t]*(v[n,t]-v[n-1,t])==0,
                             D[v[n,t],t]-v[n,t]*(u[n+1,t]-u[n,t])==0},
                             {u[n,t], v[n,t]}, {n,t}, {}];
```

The software will compute the sech solution (82) of the KdV equation (49), and the tanh solution (147) of the Volterra lattice (146).

The output is a list of sub-lists with solutions and constraints. The Backus-Naur Form of the functions are:

$$\begin{array}{rcl}
\langle PDEs\rangle & \rightarrow & \texttt{PDESpecialSolutions}[\langle Equations\rangle, \langle Functions\rangle, \\
& & \quad \langle Variables\rangle, \langle Parameters\rangle, \langle PDE\,Options\rangle] \\
\langle PDE\,Options\rangle & \rightarrow & \texttt{Form->}\langle Form\rangle \;\mid\; \texttt{Verbose->}\langle Bool\rangle \;\mid\; \texttt{InputForm->}\langle Bool\rangle \\
& \mid & \texttt{NumericTest->}\langle Bool\rangle \;\mid\; \texttt{SymbolicTest->}\langle Bool\rangle \\
\langle DDEs\rangle & \rightarrow & \texttt{DDESpecialSolutions}[\langle Equations\rangle, \langle Functions\rangle, \\
& & \quad \langle Variables\rangle, \langle Parameters\rangle, \langle DDE\,Options\rangle] \\
\langle DDE\,Options\rangle & \rightarrow & \texttt{Verbose->}\langle Bool\rangle \;\mid\; \texttt{InputForm->}\langle Bool\rangle \\
& \mid & \texttt{NumericTest->}\langle Bool\rangle \;\mid\; \texttt{SymbolicTest->}\langle Bool\rangle \\
\langle Form\rangle & \rightarrow & \texttt{Tanh} \;\mid\; \texttt{Sech} \;\mid\; \texttt{SechTanh} \;\mid\; \texttt{JacobiCN} \;\mid\; \texttt{JacobiSN} \\
\langle Bool\rangle & \rightarrow & \texttt{True} \;\mid\; \texttt{False}
\end{array}$$

11. Implementation issues

Both **PDESpecialSolutions.m** and **DDESpecialSolutions.m** have a series of modules, which perform the five steps of the algorithms described in Sections 2-6. The modules are seamlessly integrated by carefully matching output and input formats.

Below we describe how the function *PDESpecialSolutions* works. Apart from format issues to accommodate the shifts $n \pm p$, the function *DDESpecialSolutions* works similarly. However, it only uses the tanh method of Section 6.

The first non-trivial task is the computation of the leading exponents M_i and N_i. The possible leading order exponents are extracted and equated. The resulting relations are then combined in a system. That linear system is solved for M_i and N_i. If the solution for M_i and N_i has no freedom and if —upon testing— it leads to (at least two) balanced leading order terms in every equation, then the solution is kept. Fractional, negative, and complex solutions are discarded.

This is easy if the tanh- and sech-methods are applied to single PDEs, but quite a bit harder for systems of PDEs. For the mixed tanh-sech method it is exceedingly difficult. In future versions of the code we hope to design a complexity analyzer which automatically decides on the strategy to follow: (i) compute the values of the exponents and, if necessary, assign values for the free exponents; or (ii) determine upper bounds for the highest and/or free exponents; or (iii) fix all M_i, where the magnitude of the selected integer exponents depends on the complexity of the problem (number of balance equations, etc.). If (i) fails, the code would try (ii), if that fails, it does (iii). If there are multiple solutions for the leading exponents, the software will continue with the polynomials solutions for each case. Currently, for the mixed tanh-sech method, the code fixes $M_i = 2$ and $N_i = 1$.

The second major task is solving the nonlinear system of algebraic equations. We use the same custom-designed solver in all algorithms. However, for the mixed sech-tanh method we compute solutions without any assumptions for the coefficients (thus, $a_{i\,M_i} = a_{i\,2}$ and $b_{i\,N_i} = b_{i\,1}$ are allowed to be zero). Our solver for nonlinear algebraic systems implements the strategy one would follow if the solutions were computed by hand. First, try to solve for the coefficients a_{ij} in terms of wave parameters c_j and parameters α, β, etc. Next, solve for the wave parameters c_j in terms of the Greek parameters. Always back substitute the results. Finally, determine the compatibility conditions for the parameters α, β, etc.

Common sense dictates the following sequence of actions:

(i) Solve all equations *linear* in the a_{ij} first, at the cost of early branching. Favor linear equations without any parameters. Capture compatibility conditions for the parameters as they arise.

(ii) Solve equations linear in c_j, provided they are free of a_{ij}.

(iii) Solve equations linear in the external (Greek) parameters, if they are free of a_{ij} and c_j.

(iv) Solve quasi-linear equations for a_{ij}, c_j, and external parameters (quasi-linear equations involve products of the variables to the first power, e.g. $a_{11}a_{21}$).

(v) Solve quadratic equations for a_{ij}, c_j, and external parameters.

(vi) Cubic, quartic, and equations of higher degree in a_{ij}, c_j, and the (Greek) parameters are not explicitly solved.

The sequence of steps is controlled by a carefully designed complexity analyzer, which attaches weights to unknowns depending on the degree of nonlinearity, number of times they occur, etc.

The solver uses both the *Solve* and *Reduce* functions within *Mathematica* (Wolfram, 1999). The function *Reduce* uses Gröbner basis, which reduces the nonlinear algebraic system to an equivalent system, which is (hopefully) easier to solve. Loosely speaking, Gröbner basis performs eliminations in nonlinear systems similar to what Gauss-Jordan elimination does for linear systems (see Becker and Weispfenning (1993)). If there are many parameters in the algebraic system or if the system is of high degree, there is no guarantee that the algebraic solver will return a suitable result, let alone a complete result.

In an attempt to avoid the explicit use of *Mathematica*'s *Reduce* function, we considered various alternatives. For example, we used (i) variants of Gröbner bases on the complete system, or (ii) combinatorics on the coefficients in the polynomial solutions (setting $a_{ij} = 0$ or $a_{ij} \neq 0$, for the admissible i and j). For systems with several external (Greek) parameters none of these alternatives paid off.

Often, the nonlinear solver returns constraints on the wave parameters c_j and the external parameters. In principle, one should verify whether or not such constraints affect the results of the previous steps in the algorithm. In particular, one should verify the consistency with the results from step 2 of the algorithms. We have not yet implemented this type of sophistication.

To present solutions in the simplest way, we tacitly assume that all parameters (c_j, α, β, etc.) are real and positive. We added *MySimplificationRules* with rules like $\sqrt{\alpha^2} \rightarrow \alpha$, $\sqrt{-\alpha^2} \rightarrow i\,\alpha$, $\sqrt{-\beta} \rightarrow i\,\sqrt{\beta}$ and $\sqrt{-(c_1+c_2)^2} \rightarrow i\,(c_1+c_2)$.

A final task is analyzing and testing the exact solutions of the PDEs and DDEs. For example, we reject trivial (constant) solutions, and test nontrivial solutions by substituting them into the original PDE or DDE. The solutions should be tested because the algebraic solver (in step 4) repeatedly clears denominators and continues only with numerators. This may cause problems (undetermined or infinite coefficients a_{ij}) upon back substitution of constraints for c_j, α, β, etc. Such problems are largely taken care of in the algebraic solver. Nevertheless, testing the final solutions is prudent.

We have included two flavors of testing: numeric testing (which is fast) and symbolic testing (quite slow). The numeric testing scheme:

(i) Substitutes random numbers from 0 to 1 for c_i and x_i into the solutions. Also, sets $\Delta = 0$ and simplifies the solutions.

(ii) Next, substitutes random numbers for $a_{ij}, b_{ij}, m, \alpha, \beta$, etc. into the solutions and the equations.

(iii) Finally, substitutes the solutions into the PDEs or DDEs and tests that the resulting numerical value is less than 10^{-10}.

(iv) Repeats steps (i)-(iii) 13 times.

If one or more of these tests pass, the solution is accepted as being correct.

The test is performed 13 times, because *Mathematica* always evaluates $\sqrt{a^2}$ to a when a is numeric, whereas our internal simplification rules assume $\sqrt{a^2} = a$ instead of $|a|$. Therefore, solutions of this type may have a 1/2 probability of evaluating to true. By performing 13 evaluations, we reduce the probability of missing a valid solution to $1/2^{13} \approx 0.00012207$.

The symbolic test converts the solutions involving tanh and sech into exponential form. After substituting the solutions into the PDEs (or DDEs) it repeatedly (i) expands the result, (ii) applies *MySimplificationRules*, and (iii) factors the result until it no longer changes. If the final result is **0**, the solution is accepted. Otherwise, the expression is passed on to the user for inspection.

The packages **PDESpecialSolutions.m** and **DDESpecialSolutions.m** have been tested on both UNIX work stations and PCs with *Mathematica* versions 3.0, 4.0 and 4.1. A test set of over 50 PDEs and half a dozen DDEs were used.

12. Conclusions

We presented three straightforward algorithms to compute special solutions of nonlinear PDEs and DDEs, without using explicit integration. We designed the symbolic package **PDESpecialSolutions.m** which finds solitary wave solutions to nonlinear PDEs involving tanh, sech, cn and sn functions. The package **DDESpecialSolutions.m** finds closed form solutions of semi-discrete lattice equations by the tanh method.

While the software reproduces the known (and also a few presumably new) solutions for many equations, there is no guarantee that the code will compute the complete solution set of all polynomial solutions involving the tanh and/or sech functions, especially when the PDEs have parameters. This is due to restrictions on the form of the solutions and the limitations of the algebraic solver. There is so much freedom in mixed tanh-sech solutions that the current code is limited to quadratic solutions.

Furthermore, the nonlinear constraints which arise in solving the nonlinear algebraic system may be of degree higher than four, and therefore unsolvable in analytic form.

Our software package is fully automated and therefore may not return the solutions in the simplest form. The example in Section 7.7 illustrates this situation. By not solving quadratic or cubic equations explicitly the solutions (computed interactively with *Mathematica*) can be presented in a compact, more readable form.

There are obvious ways in which our software could be further generalized. One could, with considerable effort, attempt to find solutions involving complex exponentials multiplied by tanh or sech functions. A solution to the nonlinear Schrödinger equation (NLS) is of this form. Yet more general, one could attempt to find rational solutions involving tanh and sech for PDEs and DDEs.

Acknowledgements

The authors are grateful to B. Deconinck, B. Herbst, P.G.L. Leach, W. Malfliet, J. Sanders, and F. Verheest for valuable comments and suggestions.

WH thankfully acknowledges the hospitality and support of the Department of Applied Mathematics of the University of Stellenbosch, South Africa, where a substantial part of the research was done during a sabbatical leave in Spring 2001.

Part of the research was also carried out at Wolfram Research, Inc. while WH was supported by a Visiting Scholar Grant in Fall 2000.

E. Parkes and B. Duffy are thanked for sharing information about their *Mathematica* package ATFM. S. Nicodemus is gratefully acknowledged for testing early versions of our code. Undergraduate students A. Menz, S. Formanec, and B. Kowalski are thanked for their help with this project.

References

[1] Ablowitz, M.J., Clarkson, P.A. (1991). *Solitons, Nonlinear Evolution Equations and Inverse Scattering*. London Math. Soc. Lect. Note Ser. **149**. London: Cambridge University Press.

[2] Baldwin, D., Göktaş, Ü., Hereman, W., Hong, L., Martino, R.S., Miller, J.C. (2001). PDESpecialSolutions.m: A Mathematica program for the symbolic computation of exact solutions expressible in hyperbolic and elliptic functions for systems of nonlinear partial differential equations. Available at URL: http://www.mines.edu/fs_home/whereman/, and via FTP from mines.edu. Login: anonymous. Password: email address. Change to subdirectory pub/papers/math_cs_dept/software/pde-sols.

[3] Baldwin, D., Hereman, W., Hong, L., Martino, R.S., Miller, J.C. (2001). DDESpecialSolutions.m: A Mathematica program for the symbolic computation of tanh solutions for systems of nonlinear differential-difference equations, *ibid.*, subdirectory /software/dde-sols.

[4] Becker, T., Weispfenning, V. (1993). *Gröbner Bases*. Berlin: Springer Verlag.

[5] Cheb-Terrab, E.S. (2001a). Private communication. PDETools Project. Center for Experimental and Constructive Mathematics, Simon Fraser University, Vancouver, British Columbia, Canada. http://lie.uwaterloo.ca/pdetools.m

[6] Cheb-Terrab, E.S. (2001b). A computational approach for the analytical solving of systems of partial differential equations. Preprint. Submitted to: *Comp. Phys. Comm.*.

[7] Cheb-Terrab, E.S., von Bülow, K. (1995). A computational approach for the analytical solving of systems of partial differential equations. Preprint, *Comp. Phys. Comm.* **90**, 102-116.

[8] Coffey, M. (2001). Rational solution of coupled Korteweg-de Vries equations. Preprint. Submitted to: *J. Phys. A: Math. Gen.*.

[9] Das, G.C., Sarma, J. (1999). Response to "Comment on 'A new mathematical approach for finding the solitary waves in dusty plasmas'" [Phys. Plasmas 6, 4393 (1999), *Phys. Plasmas* **6**, 4394-4397.

[10] Das, K.P., Verheest, F. (1989). Ion-acoustic solitons in magnetized multi-component plasmas including negative ions, *J. Plasma Phys.* **41**, 139-155.

[11] Drazin, P.G., Johnson, R.S. (1989). *Solitons: An Introduction*. Cambridge Texts in Appl. Maths. Cambridge: Cambridge University Press.

[12] Estévez, P.G., Gordoa, P.R. (1995). Non-classical symmetries and the singular manifold method: Theory and six Examples, *Stud. Appl. Math.* **95**, 73-113.

[13] Estévez, P.G., Gordoa, P.R. (1998). Non-classical symmetries and the singular manifold method: a further two examples, *J. Phys. A: Math. Gen.* **31**, 7511-7519.

[14] Estévez, P.G., Hernáez, G. A. (2000). Non-isospectral problem in 2+1 dimensions, *J. Phys. A: Math. Gen.* **33**, 2131-2143.

[15] Fan, E.G. (2000). Extended tanh-function method and its applications to nonlinear equations, *Phys. Lett. A* **277**, 212-218.

[16] Fan, E.G. (2001). The multiform travelling wave solutions of nonlinear evolution equations, *Comp. Math. Appl.*, to appear.

[17] Gao, Y.T., Tian, B. (2001). Generalized hyperbolic-function method with computerized symbolic computation to construct the solitonic solutions to nonlinear equations of mathematical physics, *Comp. Phys. Comm.* **133**, 158-164.

[18] Hereman, W. (1990). Application of a Macsyma program for the Painlevé test to the Fitzhugh-Nagumo equation, in: Partially Integrable Evolution Equations in Physics, Proc. Winter School Theoretical Physics, (R. Conte, N. Boccara, eds.) Dordrecht: Kluwer Academic Publishers, pp. 585-586.

[19] Hereman, W. (1991). Exact solitary wave solutions of coupled nonlinear evolution equations using MACSYMA, *Comp. Phys. Comm.* **65**, 143–150.

[20] Hereman, W. (1996). Symbolic Software for Lie Symmetry Analysis. *CRC Handbook of Lie Group Analysis of Differential Equations, Volume 3: New Trends in Theoretical Developments and Computational Methods*. Chapter 13, 367–413. (N.H. Ibragimov, ed.) Boca Raton, Florida: CRC Press.

[21] Hereman, W., Banerjee, P.P., Korpel, A., Assanto, G, Van Immerzeele, A., Meerpoel, A. (1986). Exact solitary wave solutions of non-linear evolution and wave equations using a direct algebraic method, *J. Phys. A: Math. Gen.* **19**, 607-628.

[22] Hereman, W., Nuseir, A. (1997). Symbolic methods to construct exact solutions of nonlinear partial differential equations, *Math. Comp. in Simul.* **43**, 13-27.

[23] Hereman, W., Takaoka, M. (1990). Solitary wave solutions of nonlinear evolution and wave equations using Macsyma, *J. Phys. A: Math. Gen.* **23**, 4805-4822.

[24] Hereman, W., Zhuang, W. (1995). Symbolic software for soliton theory, *Acta Appl. Math.* **39**, 361–378. Also: *Proc. of KdV '95 Conf.*, 361–378. (M. Hazewinkel, H.W. Capel, E.M. de Jager, eds.) Dordrecht, The Netherlands: Kluwer Academic Publishers.

[25] Lawden, D.F. (1989). *Elliptic Functions and Applications*. Appl. Math. Sciences **80**. New York: Springer Verlag.

[26] Ludlow, D.K., Clarkson, P.A., Bassom, A.P. (1999). Similarity reductions and exact solutions of the two-dimensional incompressible Navier-Stokes equations, *Stud. Appl. Math.* **103**, 183-240.

[27] Malfliet, W. (1992). Solitary wave solutions of nonlinear wave equations, *Am. J. Phys.* **60**, 650–654.

[28] Malfliet, W. (1994). *De Tanh Methode in de Theorie van Niet-Lineaire Golven*. Habilitation thesis, Department of Physics, University of Antwerp, Antwerp, Belgium.

[29] Malfliet, W., Hereman W. (1996). The Tanh method: I. Exact solutions of nonlinear evolution and wave equations, *Phys. Scripta* **54**, 563–568.

[30] Parkes, E.J., Duffy, B.R. (1996). An automated tanh-function method for finding solitary wave solutions to non-linear evolution equations, *Comp. Phys. Comm.* **98**, 288–300.

[31] Parkes, E.J, Zhu, Z., Duffy, B.R., Huang, H.C. (1998). Sech-polynomial travelling solitary-wave solutions of odd-order generalized KdV equations, *Phys. Lett. A* **248**, 219–224.

[32] Suris, Y. (1998). Miura transformations for Toda-type integrable systems, with applications to the problem of integrable discretizations. Preprint SFB 288, Fachbereich Mathematik, Technische Universität Berlin, Berlin, Germany.

[33] Toda, M. (1981), *Theory of nonlinear lattices.* Berlin: Springer Verlag.

[34] Verheest, F., Hereman, W., Malfliet, W. (1999) Comment on "A new mathematical approach for finding the solitary waves in dusty plasma", *Phys. Plasmas* **6**, 4392-4393.

[35] Wolfram, S. (1999). *The Mathematica Book*. 4th Edition. Mathematica version 4. Urbana-Champaign, Illinois: Wolfram Media & London: Cambridge University Press.