AN EXPLORATION OF HOMOTOPY SOLVING IN MAPLE

K. HAZAVEH,* D.J. JEFFREY,* G.J. REID,*
S.M. WATT,* A.D. WITTKOPF†

* *Ontario Research Centre for Computer Algebra,*
The University of Western Ontario,
London, Ontario, Canada
† *Centre for Experimental and Computational Mathematics,*
Simon Fraser University,
Vancouver, British Columbia, Canada

Homotopy continuation methods find approximate solutions of a given system by a continuous deformation of the solutions of a related exactly solvable system. There has been much recent progress in the theory and implementation of such path following methods for polynomial systems. In particular, exactly solvable related systems can be given which enable the computation of all isolated roots of a given polynomial system. Extension of such methods to determine manifolds of solutions has also been recently achieved. This progress, and our own research on extending continuation methods to identifying missing constraints for systems of differential equations, motivated us to implement higher order continuation methods in the computer algebra language *Maple*. By higher order, we refer to the iterative scheme used to solve for the roots of the homotopy equation at each step. We provide examples for which the higher order iterative scheme achieves a speed up when compared with the standard second order scheme. We also demonstrate how existing *Maple* numerical ODE solvers can be used to give a predictor only continuation method for solving polynomial systems. We apply homotopy continuation to determine the missing constraints in a system of nonlinear PDE, which is to our knowledge, the first published instance of such a calculation.

1. Introduction

Newton's local method for square polynomial systems is a classical method for finding a root of systems with finitely many roots. Homotopy continuation methods [1] deform the known roots of a related system into the roots of the system of interest, and can calculate all the (isolated complex) roots of such systems.

Recent developments by Sommese, Verschelde and Wampler [20,19] include the extension of such homotopy methods to non-square (over- and

under-determined systems), and characterize the components or manifolds of solutions of such systems. This is part of the rapidly developing area of Numerical Algebraic Geometry initiated in [23]. This yields new methods for problems which have been traditionally approached with symbolic methods from Computer Algebra, such as factorization [3], Gröbner bases and the completion of systems of partial differential equations. We have extended this work to systems of differential equations in [14], and initiated a study of Numerical Jet Geometry, using homotopy methods.

Despite the availability of very well developed implementations for homotopy continuation methods [26,12] surprisingly little has been implemented in the context of computer algebra systems for numerical solutions for polynomial systems. We note that, for example, Gröbner bases in *Maple* are limited to polynomials with rational coefficients. In *Maple*, the existing solvers focus on univariate equations. Even when working with a powerful Polynomial Homotopy Continuation package (in our case we have extensively used Verschelde's PHCpack [26]) we found the ability to perform experiments and try out ideas in a rich environment such as *Maple* to be a valuable asset. The work we discuss here represents a starting point for *Maple*, since many of the other standard algorithms of Numerical Algebraic Geometry (such as the computation of mixed volumes) still are not implemented in that context.

Existing Homotopy implementations in *Maple* include the univariate program of Fee [6]. In that work Fee truncates the Riemann zeta function, and uses a very efficient homotopy method he has developed for analytic functions to find roots of this truncated function in a given domain. Root counts are verified by using Cauchy's integral formula, using numerical quadrature, around the boundary of the domain. Kotsireas [11] has developed a multivariate fixed step homotopy method in *Maple*.

We have implemented a variable step homotopy continuation method in *Maple*, both for second and third orders, using the code of Smith [18] as a starting point. We compare the methods, and apply them to a variety of problems arising in polynomial system solving. For scalar functions, higher-order schemes are often called Halley methods [7], because of Halley's discovery in Newton's era. Higher-order schemes allow more rapid convergence and larger step sizes in processes such as homotopy solution techniques.

In this paper, we first present the higher order method for solving a single scalar equation. Then in the next section we apply it to systems, and extend it to a homotopy method. In the applications section we apply

it to some well-known examples having finitely many roots. Finally we give the first published example of a method using homotopy continuation to identify the missing constraints in a nonlinear system of PDE.

2. Iterative schemes

Newton's method to find solutions of a single nonlinear equation $f(x) = 0$ is well known; it is also well known that the method is second order and that higher-order methods have been derived [25,7]. Here we start by giving a uniform treatment of the higher-order scalar schemes, as a preparation for the vector case.

Consider solving the scalar equation $f(x) = 0$, given an initial estimate x_0 for the solution. We expand $f(x)$ as a Taylor series around x_0

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \tfrac{1}{2}(x - x_0)^2 f''(x_0) + \cdots . \tag{2.1}$$

Setting $\Delta = x - x_0$ and assuming $f(x) = 0$, we can solve for Δ by series reversion. Abbreviating $f(x_0)$ to f for clarity, gives

$$\Delta = -\frac{1}{f'}f - \frac{f''}{2(f')^3}f^2 + \frac{3(f'')^2 - f'f'''}{6(f')^5}f^3 + \cdots . \tag{2.2}$$

The series is written as shown to emphasize that it is a series in powers of $f(x_0)$, where $f(x_0)$ will be small in some sense when x_0 is close to the root being sought. The classical Newton iteration is obtained by taking one term of this series; taking two terms gives the third-order scheme

$$\Delta = -\frac{f}{f'} - \frac{f''f^2}{2(f')^3} \ , \tag{2.3}$$

which has been called Chebyshev's method. The Halley form of (2.3) is

$$\Delta = -\frac{f}{f' - \frac{1}{2}ff''/f'} \, . \tag{2.4}$$

One derivation of this form solves (2.1) by writing $0 = f + f'\Delta + \frac{1}{2}f''\Delta^2$ as

$$-f = (f' + \tfrac{1}{2}f''\Delta)\Delta \, .$$

Now assume that the Δ within the parentheses can be approximated by its Newton approximation, obtaining

$$-f \ = \ (f' + \tfrac{1}{2}(-f/f')f'')\Delta \, ,$$

and solve this equation for Δ. None of the methods above can be applied at a point x_0 where $f'(x_0) = 0$, and Halley's method cannot be used for a function satisfying $2(f')^2 - ff'' = 0$, which means any function of the form $f(x) = 1/(Ax + B)$.

For the vector case, we use Cartesian-tensor notation [9]. When applying these results to homotopy methods, we shall give equivalent results in vector-matrix notation. Let $f : \mathbb{R}^m \to \mathbb{R}^m$ be a vector function, with component functions f_i. Let f depend upon the vector x, which in turn has components x_j. We wish to solve $f_i(x) = 0$, starting from an initial estimate $x^{(0)}$. We direct the reader to the literature where a multivariate Halley method of the type below is given [5].

The Taylor series for f about $x^{(0)}$ can be written using $\Delta_j = x_j - x_j^{(0)}$:

$$f_i(x^{(0)}) = f_i(x^{(0)}) + f_{i,k}(x^{(0)})\Delta_k + \tfrac{1}{2} f_{i,kh}(x^{(0)})\Delta_k\Delta_h + \cdots . \tag{2.5}$$

Let $\check{f}_{ki}$ be the inverse of $f_{i,k}$, defined by $\check{f}_{ki} f_{i,j} = \delta_{kj}$, where δ_{kj} is the Kronecker delta. The inverse exists and can be readily computed. Setting the left side of (2.5) to zero and solving to first order in Δ, we obtain the standard Newton iteration: $\Delta_k \approx -\check{f}_{ki} f_i$. To obtain a third-order formula, we avoid reverting (2.5) by adopting the simplified approach used above. We write $\Delta_j \approx -\check{f}_{ji} f_i + \widetilde{\Delta}_j$, and substitute this into (2.5). Solving for $\widetilde{\Delta}$, we obtain a third-order expression for Δ, analogous to the Chebyshev form above:

$$\Delta_k \approx -\check{f}_{ki} f_i - \tfrac{1}{2} \check{f}_{ki}\, f_{i,jh}\, \check{f}_{j\ell}\, f_\ell \check{f}_{hm}\, f_m \,. \tag{2.6}$$

An equivalent form of the third-order scheme can be found that is closer to Halley's scalar form. Convert (2.5) into an equation for Δ as

$$-f_i = f_{i,k}\Delta_k + \tfrac{1}{2} f_{i,kh}\Delta_k\Delta_h = \Delta_k(f_{i,k} + \tfrac{1}{2} f_{i,kh}\Delta_h) \,.$$

We replace Δ_h with its Newton approximation and get

$$-f_i = \Delta_k(f_{i,k} - \tfrac{1}{2} f_{i,kh}\check{f}_{h,m} f_m) = \Delta_k T_{ik} \,.$$

Denoting the inverse of T by $\check{T}$ we obtain the Halley form as

$$\Delta_k = -f_i \check{T}_{ki} \,. \tag{2.7}$$

In the scalar case, if the two third-order schemes (2.3) and (2.4) are computed in a naïve way, Halley's form reduces the operation count by one multiplication, but in the vector case, we have an additional matrix inverse to compute (or linear system to factor).

In moving from a second-order method to a third-order method, there are two separate effects to consider: the speed of convergence and the

basin of attraction. If an estimate $x^{(0)}$ is sufficiently close to an isolated non-singular root $x^{(e)}$, then with each iteration a second-order method will approximately double the number of digits that are correct, while a third-order method will triple that number [8]. Thus, for example, an estimate that is correct to 1 digit can be improved to 8 digits in 3 second-order steps or 2 third-order steps. Since the computation of the second derivative term is often expensive, in the scalar case, a higher-order method is usually not an advantage. However, in the vector case, an iteration requires a matrix inverse, making the iteration more expensive. In addition higher order derivatives for polynomial systems can be cheaply obtained (e.g. by automatic differentiation [4]) and this opens the possibility that the third-order method will be more efficient.

If an initial estimate $x^{(0)}$ is further away from the root, and the convergence theorems do not apply, then we must consider the basins of attraction of the root. Graphical presentations of how particular basins of attraction change with the iterative scheme have been published recently [25]. We expect the basin of attraction to be larger for higher-order methods, but have yet to investigate this.

3. Homotopy Method

Consider a system of equations $p(x) = 0$, which we wish to solve. Both p and x are vectors. Suppose we possess a system of equations $q(x) = 0$ whose solutions are already known. Then the homotopy function

$$H(x,t) = q(x)(1-t) + tp(x) = q(x) + t(p(x) - q(x)) \ , \tag{3.1}$$

is such that $H(x,0) = q(x) = 0$ is a vector system with known solutions, and the system we want to solve is $H(x,1) = p(x) = 0$. We deform from the system at $t = 0$ to the one at $t = 1$ in variable steps Δt. The homotopy parameter t is often called "time."

At a given time t, our problem is to solve $H(x,t) = 0$, and this is done using a iterative scheme. Usually this is a second-order (Newton) scheme, but here a third-order scheme will be used. Since the solution is iterative, a starting estimate is needed. This can be obtained either from the known solutions to $q(x) = 0$, or from the solution at an earlier time, t'. Typically this is the previous time step, that is, $t = t' + \Delta t$. Each homotopy step consists of a predictor stage and a corrector stage. During the predictor stage, a starting estimate is generated for the root, then refined in the corrector stage.

An appropriate start system is now described for the homotopy. Let $p(x) = (p_1(x), \ldots, p_n(x)) = 0$ denote the system of n polynomial equations in n unknowns that we wish to solve. Let d_j denote the total degree of the jth equation (that is, the degree of the highest order monomial in the equation). Then such a start system is

$$q_j(x) = \mathrm{e}^{i\phi_j}\left[x_j^{d_j} - \left(\mathrm{e}^{i\theta_j}\right)^{d_j}\right] = 0 \qquad (j = 1, 2, \ldots, n), \tag{3.2}$$

where ϕ_j, θ_j are random real numbers in the interval $[0, 2\pi]$. The equation above has the obvious particular solution $x_j = \mathrm{e}^{i\theta_j}$ and the complete set of starting solutions for $j = 1, 2, \ldots, n$, is given by:

$$\{\exp(i\theta_j + 2\pi ik/d_j) : k = 0, 1, \ldots, d_j - 1\}. \tag{3.3}$$

Bézout's Theorem states that the number of isolated roots of such a system is bounded above by $d_1 d_2 \cdots d_n$. In particular, with the above start system, with probability 1, each isolated root will lie at the end of an analytic homotopy path originating from one of the starting roots [1]. Bézout's Theorem can be proved by the "method of degeneration," which uses arguments very similar to homotopies [29,27]. More efficient methods use mixed volume to get a smaller upper bound on the number of isolated roots [12], and indeed Bernstein's proof [2] provides a constructive homotopy method.

3.1. *Predictor*

First note that solving $H(x, t) = 0$, where $x(0) = \alpha$ is an exact known root of $q(x) = 0$, is equivalent to solving $\frac{d}{dt}H = 0, x(0) = \alpha$ and so to solving

$$H_x \frac{dx}{dt} + H_t = 0,\ x(0) = \alpha,\ \ 0 \le t \le 1 \tag{3.4}$$

where H_x is the Jacobian of H with respect to $x_1, \ldots, x_n$. Naturally this has led to the use of ODE software for higher order predictors in continuation methods [1, Section 6.3] and [13]. One extreme is not to use any corrector steps.

For convergence the path should stay in the basin of attraction of the sought after isolated root at $t = 1$. To ensure this, most approaches use a combination of corrector and predictor steps. In particular the residuals are monitored to see that they have the characteristic pattern of reduction consistent with the convergence of Newton's method when in the basin of attraction of a root [26]. More recent results on the rigorous identification of the onset of convergence are given in Shub and Smale [17]. If only predictor steps are used, then at the end of a step H will typically have

a small nonzero value, and the next step will involve solving a slightly perturbed problem $H = \delta, x(\Delta t) = \widetilde{\alpha}$. This can lead to an accumulation of error, unless residuals are carefully monitored.

ODE integrators usually adapt their step length according to relative error, whereas corrector methods for Newton's method usually use a combination of residual and relative errors. Specifically the residual error $\|H(x,t)\|$ at time t and the error, $\|\Delta x\|$, where Δx is the difference between successive values of x, are compared with a working tolerance ε. A challenge then is to reflect the additional residual control in ODE integrators.

Another view of homotopy solving is that of solving a differential equation on a manifold (that is, solving a differential-algebraic equation). Both Visconti [28] and Arponen [24] have implemented DAE solving methods in *Maple*, and it would be of interest to use these in homotopy solving.

Differentiating the ODE (3.4) yields expressions for the higher order derivatives:

$$H_x \frac{d^2x}{dt^2} + \frac{d}{dt} H_x \frac{dx}{dt} + \frac{d}{dt} H_t = 0, \quad H_x \frac{d^3x}{dt^3} + \quad \cdots \quad = 0, \quad \cdots \text{ etc}, \tag{3.5}$$

where $\frac{d}{dt} = \frac{\partial}{\partial t} + \Sigma_j \frac{dx_j}{dt} \frac{\partial}{\partial x_j}$. As is well-known, this leads to predictor methods of any order. For example, a higher order predictor method is easily given which is analogous to the higher order corrector method given in the next section.

3.2. *Corrector*

In path-following methods, corrector methods refine the solutions at a fixed time t. The appropriate inclusion of such a corrector phase after each predictor step, can eliminate the accumulation of error involved in pure predictor ODE based methods. In this section we describe a multivariate corrector method (also see a related multivariate Halley method given in [5]). Suppose we have an estimate x, which we wish to improve by computing $x + \Delta x$ at a fixed t. Expanding about (x,t) to second order gives:

$$H(x + \Delta x, t) = H(x,t) + H_x \Delta x + O(\Delta x^2) \ . \tag{3.6}$$

Here H_x is the Jacobian matrix of the vector system H. We use the language of linear algebra here, because it is more convenient for implementation in *Maple*. Then to second order $H(x,t) + H_x \Delta x \approx 0$ and we define this second order approximation of the solution $x + \Delta x$ as $x + \widetilde{\Delta x}$ where $\widetilde{\Delta x}$ satisfies the (vector) system

$$H_x \widetilde{\Delta x} = -H, \tag{3.7}$$

which as usual will be directed to linear solvers, instead of the more expensive computational approach of inverting H_x. Now expanding to third order gives:

$$H(x+\Delta x,t) = H(x,t) + H_x\Delta x + (\Delta x)^T H_{xx}(\Delta x)/2 + O(\Delta x^3)\,, \quad (3.8)$$

where $(\Delta x)^T$ is the transpose of the column vector Δx. If the i-th component of H is denoted by H^i, then in the equation above, H^i_{xx} is an $n \times n$ matrix with entries $H^i_{x_j x_k}$. Suppose $\Delta x = \widetilde{\Delta x} + \widetilde{\widetilde{\Delta x}}$. Then $\widetilde{\Delta x}$ is of order 2 and $\widetilde{\widetilde{\Delta x}}$ satisfies (3.8) to order 3. Substitution of this expression into (3.8), using (3.7), ignoring terms in $\Delta x \widetilde{\Delta x}$, and the above, then shows that $\widetilde{\widetilde{\Delta x}}$ satisfies to third order the following:

$$H_x \widetilde{\widetilde{\Delta x}} = -(\widetilde{\Delta x})^T H_{xx}(\widetilde{\Delta x})/2. \quad (3.9)$$

¿From a computational point of view, notice the difference between the above expression (3.9) and (2.6). Here the number of computations has been reduced. Also notice that two linear systems must be solved: (3.7) and (3.9). The coefficient matrix is H_x in both cases, so naturally a gain in performance can be realized by computation of an LU factorization, which can then be used twice.

3.3. *Implementation of the Homotopy Algorithm*

We were guided in part by Verschelde's excellent program PHCPack [26]. Another highly developed program is the one by T.Y. Li and his team [12] which enables highly efficient computation of mixed volumes. An efficient and parallel implementation of polyhedral continuation in MATLAB is the work of Kim and Kojima [10].

We have made use of the LinearAlgebra Package in *Maple*, and in particular its interface to NAG Library routines.

Using the predictor and corrector methods given above, we now wish to step from $t = 0$ to $t = 1$. The key to efficiency lies in the use of a variable step size. The strategy used in our code is based on counting the number of iterations used at each step. If the corrector step succeeds in only one iteration, then the step size is increased by the factor 1.25 before the next step is taken. If the number of iterations needed at any step is too large, then the step size is decreased. The number of iterations that we count as too large is a parameter that can be tuned. More elaborate stepping schemes have been described elsewhere.

Again for efficiency, the problem $H(x,t) = 0$ is solved to a lower accuracy along the path, and then when a possible solution is obtained at $t = 1$, the "end game" is entered and the solutions are obtained to a greater accuracy. Typically the maximum number of iterative improvements is increased.

Having an environment such as *Maple* opens up the possibilities to use automatic differentiation to calculate the higher order derivatives. Specifically the derivatives are encoded as programs for which good complexity estimates are known [4].

4. Application to some polynomial systems

The algorithms above were coded in *Maple* with a parameter that allowed us to turn the third-order terms on and off. In comparing performance of the codes, we can select between many different metrics. In complicated systems such as *Maple*, there is a particular difficulty of separating the efficiency of the mathematical method from the details of the programming. For this reason, we have selected to test the average size of a step and the number of iterative loops used by the corrector code.

We apply the code to the following simple problems: the intersection of 2 curves given by

$$x^2 + y^2 = 1, \quad x + 2y - 6 = 0\,; \tag{4.1}$$

a univariate problem similar to the well-known Wilkinson polynomial

$$(x-1)(x-2)(x-3)(x-4)(x-5) = 0\,; \tag{4.2}$$

and a cyclic 3 roots problem

$$a + b + c = 0\ ,\ ab + bc + ca = 0,\ abc - 1 = 0. \tag{4.3}$$

Results for these three problems are denoted by the rows labelled *Wilkonson5*, *curves2* and *cyclic3* in the Table below. Also included in that table are some relatively small problems taken from the selection of demo problems given at the Demos link on Jan Verschelde's web site [26]. These problems have rows labelled by their designation at Verschelde's web site (*noon3, lorentz, eco5d*).

The six problems listed were solved using a second order homotopy code based on the algorithms described above. Then the same problems were solved using basically the same code, but with a third-order iterative scheme. For each case, the number of steps taken was recorded and the average step size computed. Also counted was the total number of times

the iterative solver was called (abbreviated as Iters below). In all cases, the third-order method used fewer steps and fewer iterations. These statistics are presented in the table below, using N for second order and H for third order.

Problem	N-Time	N-Steps	N-Iters	H-Time	H-Steps	H-Iters
Wilkinson5	2.83	161	285	2.59	119	163
curves2	0.97	60	93	1.05	51	71
cyclic3	4.20	207	360	3.84	149	218
noon3	26.90	1202	2381	24.06	722	1242
lorentz	9.64	391	904	8.80	236	422
eco5d	83.93	2481	5479	78.71	1528	2723

5. Pure Predictor Method in *Maple*

This section contains a description, illustrated by an example, of the use of existing tools in *Maple* to apply homotopy methods to the computation of the roots of a polynomial system. Consider a system of three quadratic equations in three variables,

$$\begin{aligned} p_1 &= 26x^2 - 55xy + 37xz - 94x - 65y^2 + 90yz - 38y - 46z^2 + 28z + 88 = 0, \\ p_2 &= 64x^2 - 22xy - 37xz + 68x - 84y^2 + 80yz + 23y - 20z^2 - 7z + 4 = 0, \\ p_3 &= -77x^2 + 40xy + 21xz + 55x + 61y^2 + 5yz + 66y - 83z^2 - 26z + 27 = 0. \end{aligned} \tag{5.1}$$

We solve this system by deforming the roots of the simpler exactly solvable system

$$q_1 = (x - s_1)^2 - 1 = 0,\ q_2 = (y - s_2)^2 - 1 = 0,\ q_3 = (z - s_3)^2 - 1 = 0. \tag{5.2}$$

The homotopy equations are

$$H_j = tp_j + \alpha_j(1 - t)q_j = 0 \qquad (\alpha_j \neq 0,\ j = 1, 2, 3). \tag{5.3}$$

Following the method of Section 3.1, we differentiate each equation in (5.3) with respect to t and obtain a differential system for $x(t), y(t), z(t)$. The system, with initial conditions corresponding to the known solutions of (5.2), is numerically integrated from $t = 0$ to $t = 1$.

The complex parameters s_j in (5.2) are used in the heuristic method of this section to avoid path problems. For example these problems include path crossing when the starting points are real. Values of $s_1 = 2 + \mathrm{i}$, $s_2 = 1 - \mathrm{i}$, $s_3 = -1 + 2\mathrm{i}$ were chosen for the experiment, though a more rigorous choice would have been the starting system (3.2). In addition, the free constants α_j, used in construction of the homotopy equation (5.3),

were chosen to be the leading coefficients of the input equations (5.1) with respect to the same variable as the corresponding known solution equation from (5.2).

Integration of the system should be done with care. Direct techniques are of little use, as they require bringing the system into a symbolic solved form for its derivatives before application of the integration technique. The coefficients of the system can be arbitrarily large, depending on the order and density of the system. Performing Gaussian elimination on such a system to bring the system into a solved form for its derivatives is very expensive and often results in numerical instability. Here this problem is addressed by retaining the matrix form of the system, and numerically solving for the derivatives at each predictor step after the coefficients are evaluated at the corresponding time t. This is done in the numerical ODE solvers in *Maple* 8 using the new option *implicit=true* in the call to the numerical ODE solution procedure.

Many existing numerical ODE solvers are restricted to the use of real data, though the path of the system solutions must clearly be complex (as the solutions themselves are complex valued in general). *Maple*'s numerical solvers can handle complex data, but in general the use of the real solver in *Maple* applied to the real and imaginary parts of the system is more efficient for low degree polynomial systems.

The following *Maple* 8 script applies the process described above.

```
# The input system
pols1:=[26*x^2-55*x*y+37*x*z-94*x-65*y^2+90*y*z-38*y-46*z^2+28*z+88,
        64*x^2-22*x*y-37*x*z+68*x-84*y^2+80*y*z+23*y-20*z^2-7*z+4,
       -77*x^2+40*x*y+21*x*z+55*x+61*y^2+5*y*z+66*y-83*z^2-26*z+27]:
# Polynomials with known roots
pols0 := [coeff(pols1[1],x,2)*(x^2-1), coeff(pols1[1],y,2)*(y^2-1),
   coeff(pols1[1],z,2)*(z^2-1)        ]:
# A complex shift - for evaluation of known solution polynomials
sx := 2+I:  sy := 1-I:  sz := -1+2*I:
pols0 := eval(pols0,{x=x-sx,y=y-sy,z=z-sz}):
# Now construct the homotopy system
hsys := eval({seq((1-t)*pols0[i]+t*pols1[i],i=1..3)},
        {x=x(t),y=y(t),z=z(t)}):
dsys := diff(hsys,t):
# Split the system into real and imaginary parts
split:={x(t)=xr(t)+I*xi(t),y(t)=yr(t)+I*yi(t),z(t)=zr(t)+I*zi(t)}:
tmp := collect(eval(expand(eval(dsys,split)),I=II),II):
dsys:={seq(coeff(tmp[i],II,0),i=1..3),
         seq(coeff(tmp[i],II,1),i=1..3)}:
```

```
# The initial conditions corresponding to the known solutions
idata :=[seq(seq(seq({xr(0)=2*i-1+Re(sx),xi(0)=Im(sx),
 yr(0)=2*j-1+Re(sy),yi(0)=Im(sy), zr(0)=2*k-1+Re(sz),zi(0)=Im(sz)},
          k=0..1),j=0..1),i=0..1)]:
# Loop through the data, and obtain the solution for each
for id in idata do
   # Construct the dsolve/numeric procedure
   dsn := dsolve(dsys union id, numeric, implicit=true):
   # Obtain the solution at t=1 and print it.
   sol := dsn(1);
   print(evalf[7](
        eval([x=xr(t)+I*xi(t),y=yr(t)+I*yi(t),z=zr(t)+I*zi(t)],sol)
   ));
end do:
```

The output from running this script is

```
[x = 1.384601 + 0.5873485 I,
 y = -2.516459 - 0.1233784 I,
 z = -1.181569 + 0.7662005 I]
[x = 1.041660 + 2.196067 I,
 y = -0.1078828 - 3.715860 I,
 z = 1.630095 - 1.988129 I]
[x = 0.1122859 + 0.3201893*10^(-6) I,
 y = 0.1227375 + 0.1602351*10^(-6)I,
 z = -0.8616124 - 0.3224805*10^(-7)I]
[x = 0.4493258 + 0.8280515*10^(-6)I,
 y = 1.316899 - 0.7538583*10^(-6)I,
 z = 1.685232 - 0.1398326*10^(-6)I]
 [x = 1.384601 - 0.5873477 I,
 y = -2.516459 + 0.1233783 I,
 z = -1.181569 - 0.7661998 I]
[x = 1.041658 - 2.196065 I,
 y = -0.1078817 + 3.715856 I,
 z = 1.630094 + 1.988127 I]
[x = -2.656328 + 5.385065 I,
 y = -1.330291 + 4.515593 I,
 z = 1.681109 + 4.152297 I]
[x = -2.656328 - 5.385073 I,
 y = -1.330290 - 4.515598 I,
 z = 1.681113 - 4.152299 I]
```

The computation took around 1 second on a 1.5GHz machine. Of course a Newton improvement could be done at the end to increase accuracy. Evaluation of the original quadratic equations Eqn. (5.1) yields residuals of magnitude less than 10^{-4}. This reflects the working tolerances of the default method (10^{-7}). Tightening of these tolerances to 10^{-10} provides

solutions having residuals of less than 10^{-6} (in 2.5 sec.), and a further reduction of the tolerances to 10^{-12} provides solutions having residuals of less than 10^{-8} (in 5.5 sec.).

6. Application to a Nonlinear System of Partial Differential Equations

In this section we give a new application of homotopy methods to systems of partial differential equations. Specifically we apply our generalization [14] of the methods of Sommese, Verschelde and Wampler [20] to the following nonlinear system of PDE for $u = u(x, y)$:

$$\frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial x \partial y} = 0, \quad \left(\frac{\partial u}{\partial x}\right)^p + \frac{\partial u}{\partial x} - u = 0. \tag{6.1}$$

The aim is to complete (6.1) to an involutive system as defined by the geometric theory of PDE (see [16] and the references therein). This system first appeared in an article to illustrate a new exact elimination algorithm for simplifying systems of PDE [15]. We present, for the first time, a PDE example of the interpolation-free homotopy method described in [14, §6.4], made possible by the works [21,22].

We confine ourselves to the case $p = 2$. In terms of jet coordinates (which are formal indeterminates corresponding to derivatives of the dependent variables, etc.) this is a differential polynomial system in the jet space of second order $J^2 \approx \mathbb{C}^6$. The zero set of the maps defining the PDE, its so-called jet variety, is:

$$\{(u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) \in J^2 : u_{yy} - u_{xy} = 0,\ u_x^2 + u_x - u = 0\}. \tag{6.2}$$

Here we have suppressed the independent variables x, y since they don't appear explicitly in the PDE. In [14] some homotopy tools for the new area of Numerical Jet Geometry were described, which are now applied to identify the missing constraints of the system (6.2). This completion process can be viewed in J^∞ as generating a descending sequence of manifolds, until that sequence stabilizes.

Setting

$$\phi^1 = u_{yy} - u_{xy},\ \phi^2 = u_x^2 + u_x - u\ , \tag{6.3}$$

the system (6.3) is differentiated (prolonged) up to and including order 2 yielding a system denoted by R, with the associated jet variety:

$$\begin{aligned} V(R) := \{&(u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) \\ &\in J^2 : \phi^1 = 0, \phi^2 = 0, D_x\phi^2 = 0, D_y\phi^2 = 0\}. \end{aligned} \tag{6.4}$$

Here D_x and D_y are the usual formal total derivatives so that $D_x\phi^2 = (2u_x + 1)u_{xx} - u_x = 0$, etc. Thus we have 4 equations in the 6 unknowns $(u, u_x, u_y, u_{xx}, u_{xy}, u_{yy})$. Now regarded as a submanifold of J^2, the dimension of $V(R)$ satisfies $\dim V(R) \leq 4$ since we already have 2 obviously independent PDEs $\phi^1 = 0$ and $\phi^2 = 0$, and $\dim J^2 = 6$. To check if $V(R)$ has components of dimension 4 in J^2, we intersect it with a random 2 dimensional linear subspace of $\mathbb{C}^6$. This linear space is the solution set of 4 random linear equations of the form:

$$\psi^j := a_{j0}+a_{j1}u+a_{j2}u_x+a_{j3}u_y+a_{j4}u_{xx}+a_{j5}u_{xy}+a_{j6}u_{yy} = 0, \qquad (6.5)$$

where $j = 1, 2, 3, 4$ and the a_{jk} are random complex floating point numbers. The equations (6.5) together with those in (6.4) form a system of 8 equations for 6 variables for the intersection of $V(R)$ with this subspace. Following the procedure in [20] we square the system by incorporating 2 slack variables z_1, z_2. The resulting square system now has 8 equations in 8 unknowns:

$$\begin{aligned} \phi^1 + \nu_{11}z_1 + \nu_{12}z_2 &= 0, \\ \phi^2 + \nu_{21}z_1 + \nu_{22}z_2 &= 0, \\ D_x\phi^2 + \nu_{31}z_1 + \nu_{32}z_2 &= 0, \\ D_y\phi^2 + \nu_{41}z_1 + \nu_{42}z_2 &= 0, \\ \psi^j + \gamma_{j1}z_1 + \gamma_{j2}z_2 &= 0 \qquad (j = 1, 2, 3, 4), \end{aligned} \qquad (6.6)$$

where the ν's and γ's are random floating point complex numbers.

Applying our *Maple* program Homotopy to this system yields no solutions with $z_1 = 0, z_2 = 0$. Thus we conclude, numerically, that there are no 4 dimensional components in $V(R)$. Next we check for 3 dimensional components by removing one of the random linear equations, and one of the slack variables (e.g. z_2 by setting $\nu_{j2} = \gamma_{j2} = 0$ in (6.6)) so that the system remains square. Again no solutions are found with $z_1 = 0$ so we conclude that there are no 3 dimensional components. Removing z_1 and one of the remaining random linear equations, and running Homotopy, we find that there are solutions, so we conclude that there exist 2 dimensional components in $V(R)$. A more thorough analysis using the methods of [20] shows that there is only one irreducible component. We conclude that $\dim V(R) = 2$.

We prolong (differentiate) the system R to order 3 in $J^3 \approx \mathbb{C}^{10}$ resulting in the system of equations whose variety we will denote by DR:

$$\begin{aligned} \phi^1 = 0, \phi^2 = 0, D_x\phi^2 = 0, D_y\phi^2 = 0, D_x\phi^1 = 0, \\ D_y\phi^1 = 0, D_{xx}\phi^2 = 0, D_{xy}\phi^2 = 0, D_{yy}\phi^2 = 0. \end{aligned} \qquad (6.7)$$

In addition to prolongation, yet another fundamental operation in the jet geometry of differential equations is the projection $\pi : J^q \longrightarrow J^{q-1}$. As described in [14] we implement projection (geometric elimination) onto J^{q-1} by adjoining random linear equations in the J^{q-1} variables alone.

For example to compute $\dim \pi(DR)$ we adjoin the random linear equations in the form (6.5). Slack variables are incorporated to square the system, and similarly it is determined that $\dim \pi(DR) = 1$. Next we calculate D^2R, its dimension, and the dimensions $\dim \pi(D^2R)$, $\dim \pi^2(D^2R)$ of its projections. We summarize the dimensions of these systems below:

$\dim R = 2$		
$\dim \pi(DR) = 1$	$\dim DR = 1$	
$\dim \pi^2(D^2R) = 1$	$\dim \pi(D^2R) = 1$	$\dim D^2R = 1$

For a projection of the output system $\pi^\ell(D^kR)$ to be involutive, it should satisfy a projected dimension test and have involutive symbol [30]. Verifying the projected dimension test involves checking for the maximum $\ell \leq k$, if it exists, such that its dimension satisfies $\dim \pi^\ell(D^kR) = \dim \pi^{\ell+1}(D^{k+1}R)$. This is first satisfied when $k = \ell = 1$, since $\dim \pi(DR) = \dim \pi^2(D^2R)$. Without going into technical details, the involutive symbol test is achieved by computing dimensions. In particular, the dimensions of the symbols of the systems above can be determined from their dimensions above. We find that the dimension of the symbol of $\pi(DR)$ is zero, and hence is involutive. This is in agreement with the results found in [15]. A finer analysis, using homotopy methods, shows that $V(R)$ in J^1 factors into 2 irreducible 1 dimensional components, in accordance with the exact results found in [15]. In particular the exact constraints found by [15] in J^1 are $u_y(u_y - u_x) = 0, u_x^2 + u_x - u = 0$.

The involutive system $\pi(DR)$ can be used to state existence and uniqueness theorems for solutions, and give also the conditions for initializing consistently numerical solvers [24], thus improving their stability.

Finally we mention that if $p > 2$ in (6.1) then as shown in [15] components with higher multiplicity than one can occur (we note that it is always numerically possible to determine when we are in the multiplicity one case, using the methods of Sommese, Verschelde and Wampler, and to bound the multiplicity in the other cases). In the exact case this means that formal derivatives of PDE may not yield the same results as geometric derivatives, and our interpolation-free method may terminate prematurely, before all constraints are found. In the case that the given ideals are radical, then

this problem does not occur (this is a generalization of the algebra-geometry correspondence to PDE), and is achieved in the exact case for our example by constructing representations for radicals of algebraic ideals occurring in the computation. In the approximate case the interpolation dependent methods play the same role. However constructing an interpolation-free method in the higher multiplicity case remains an open problem, which is important because of the higher complexity of the interpolation dependent methods.

7. Acknowledgements

Two of the authors (GR and KH) thank Jan Verschelde for helpful discussions. GR thanks Ilias Kotsireas, and Chris Smith for discussions.

References

1. E. L. Allgower, K. Georg. Numerical path following. In P. G. Ciarlet, J. L. Lions, eds. *Scientific Computing* (*Part 2*), 3–203. Volume 5 of *Handbook of Numerical Analysis*, North-Holland, 1997.
2. D. N. Bernstein. The number of roots of a system of equations. (Russian) *Functional Anal. Appl.* **9**(3) (1975), 183–185 (English Translation, 1976).
3. R. M. Corless, A. Galligo, I. S. Kotsireas, S. M. Watt. A geometric-numeric algorithm for absolute factorization of multivariate polynomials. In T. Mora, ed. *Proceedings of ISSAC 2002, Lille, France*, 37–45. ACM Press, 2002.
4. G. Corliss, C. Faure, A. Griewank, L. Hascoët, U. Naumann (eds.) *Automatic Differentiation 2000: ¿From Simulation to Optimization*. Springer, New York, 2001.
5. A. A. M. Cuyt, L. B. Rall. Computational implementation of the multivariate Halley method for solving nonlinear systems of equations. ACM Transactions on Mathematical Software (TOMS), Archive **11**(1) (1985), 20–36.
6. G. Fee. Computing Roots of Truncated Zeta Functions. Poster. MITACs Annual Meeting, Pacific Institute of the Mathematical Sciences, June, 2002.
7. E. Halley. Methodus nova, accurata & facilis inveniendi radices aequationum quarumcunque generaliter, sine praevia reductione. *Philos. Trans. Roy. Soc. London* **18** (1694), 139–148.
8. D. J. Jeffrey, M. W. Giesbrecht, R. M. Corless. Integer roots for integer-power-content calculations. In X.-S. Gao, D. Wang, eds. *Computer mathematics, Proceedings of the Fourth Asian Symposium ASCM 2000*, 195–203. *Lecture Notes Series on Computing* **8**, World Scientific, Singapore, 2000.
9. H. Jeffreys. *Cartesian tensors*. Cambridge University Press, 1965.
10. S. Kim, M. Kojima. CMPSm: A continuation method for polynomial systems (MATLAB Version). In A. M. Cohen, X. Gao, N. Takayama eds. *Mathematical Software, ICMS2002 Beijing, China, Aug 17–19, 2002*. World Scientific, Singapore, 2002.

11. I. S. Kotsireas. Homotopies and polynomial system solving I. Basic Principles. *SIGSAM Bulletin* **5**(1) (2001), 19–32.
12. T. Y. Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica* **6** (1997), 399–436.
13. B. N. Lundberg, A. B. Poore. Variable order Adams-Bashforth predictors with error-stepsize control for continuation methods. *SIAM J. Sci. Statist. Comp.* **12**(3) (1991), 695–723.
14. G.J. Reid, C. Smith, J. Verschelde Geometric completion of differential systems using numeric-symbolic continuation. *SIGSAM Bulletin* **36**(2) (2002), 1–17.
15. G. J. Reid, A.D. Wittkopf, A. Boulton. Reduction of systems of nonlinear partial differential equations to simplified involutive forms. *Eur. J. of Appl. Math.* **7**, 604–635.
16. G. J. Reid, P. Lin, A. D. Wittkopf. Differential elimination-completion algorithms for DAE and PDAE. *Studies in Applied Mathematics* **106**(1) (2001), 1–45.
17. M. Shub, S. Smale. Complexity of Bezout's theorem V: Polynomial time. *Theoretical Computer Science* **133**(1) (1994), 141–164.
18. C. Smith. Further Development in HomotopySolve for *Maple* 7. Undergraduate Thesis, Department of Applied Mathematics, University of Western Ontario. 2002.
19. A. J. Sommese, J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *J. Complexity* **16**(3) (2000), 572–602.
20. A. J. Sommese, J. Verschelde, C. W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM J. Numer. Anal.* **38**(6) (2001), 2022–2046.
21. A. J. Sommese, J. Verschelde, C. W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In C. Ciliberto, F. Hirzebruch, R. Miranda, M. Teicher (eds.) *Application of Algebraic Geometry to Coding Theory, Physics, and Computation*, 297–315. *Proceedings of a NATO Conference, February 25–March 1, 2001, Eilat, Israel*. Kluwer Academic Publishers, 2001.
22. A. J. Sommese, J. Verschelde, C. W. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM J. Numer. Anal.* **40**(6) (2002), 2026–2046.
23. A. J. Sommese, C.W. Wampler. Numerical algebraic geometry. In J. Renegar, M. Shub, S. Smale (eds.) *The Mathematics of Numerical Analysis, Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, July 17–August 11, 1995, Park City, Utah*, 749–763. *Lectures in Applied Mathematics* Volume **32**, 1996.
24. J. Tuomela, T. Arponen. On the numerical solution of involutive ordinary differential systems. *IMA J. Numer. Anal.* **20** (2000), 561–599.
25. J. L. Varona. Graphic and numerical comparison between itarative methods. *Mathematical Intelligencer* **24** (2002), 37–46.

26. J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software* **25**(2) (1999), 251–276. Software site: `http://www.math.uic.edu/~jan`.
27. J. Verschelde. Polynomial homotopies for dense, sparse and determinantal systems. *Mathematical Sciences Research Institute Preprint* # 1999-041, 1999. Available online at `http://www.msri.org`.
28. J. Visconti. *Numerical Solution of Differential Algebraic Equations, Global Error Estimation and Symbolic Index Reduction.* Ph.D. Thesis. Laboratoire de Modélisation et Calcul. Grenoble. 1999.
29. A. Weil. *Foundations of Algebraic Geometry. AMS Colloquium Publications.* Volume XXIX, Providence, Rhode Island, 1962.
30. A. D. Wittkopf, G. J. Reid. Fast differential elimination in C: The CDiffElim Environment. *Comp. Phys. Comm.* **139**(2) (2001), 192–217.