Journal of Computational Physics 230 (2011) 5957–5973

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A numerical methodology for the Painlevé equations

Bengt Fornberg [a,*], J.A.C. Weideman [b]

[a] University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA
[b] University of Stellenbosch, Department of Mathematical Sciences, Private Box X1, Matieland 7602, South Africa

ARTICLE INFO

Article history:
Received 31 January 2011
Accepted 3 April 2011
Available online 13 April 2011

Keywords:
Painlevé transcendents
P_I equation
Taylor series method
Padé approximation
Chebyshev collocation method

ABSTRACT

The six Painlevé transcendents $P_I - P_{VI}$ have both applications and analytic properties that make them stand out from most other classes of special functions. Although they have been the subject of extensive theoretical investigations for about a century, they still have a reputation for being numerically challenging. In particular, their extensive pole fields in the complex plane have often been perceived as 'numerical mine fields'. In the present work, we note that the Painlevé property in fact provides the opportunity for very fast and accurate numerical solutions throughout such fields. When combining a Taylor/Padé-based ODE initial value solver for the pole fields with a boundary value solver for smooth regions, numerical solutions become available across the full complex plane. We focus here on the numerical methodology, and illustrate it for the P_I equation. In later studies, we will concentrate on mathematical aspects of both the P_I and the higher Painlevé transcendents.

1. Introduction

The six Painlevé transcendents were discovered about 100 years ago [4,13,23,24], and have since received extensive attention. They satisfy nonlinear second order ODEs, cannot be reduced to other types of special functions, and obey the Painlevé property of all its solutions being free from movable branch points (however, the solutions may have movable poles or movable isolated essential singularities) [18,21]. They arise in many branches of physics, including classical and quantum spin models, scattering theory, self-similar solutions to nonlinear dispersive wave PDEs, string theory and random matrix theory. For a summary of their applications, with extensive references, we refer to [7] and to Sections 32.13–32.16 in the NIST Handbook of Mathematical Functions [21].

The attention that the Painlevé functions have received over the last century has primarily been limited to their analytic properties and applications. The numerical computation of these functions, by contrast, has received comparatively little attention. With the exceptions of [8,20,22] (discussed later), numerical studies have so far mostly focused on pole-free intervals along the real line [17,21,25] or on a pole-free region in the complex plane [11]. A possible explanation for this state of affairs is that the large pole fields in the complex plane associated with the Painlevé ODEs may be thought of as a challenge to standard numerical methods for the integration of ODEs.

We argue in the present work that these pole fields provide, in fact, especially advantageous integration opportunities. A 'pole friendly' algorithm based on Padé approximation is devised that is capable of integrating accurately and stably through pole fields for distances in the $10^5 - 10^6$ range. This allows numerical studies not only of 'near-fields', but also of transition processes towards 'far fields', for which an extensive body of asymptotic work is available [16,21]. When the pole fields do

* Corresponding author. Tel.: +303 492 5915; fax: +303 492 4066.
E-mail addresses: fornberg@colorado.edu (B. Fornberg), weideman@sun.ac.za (J.A.C. Weideman).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.04.007

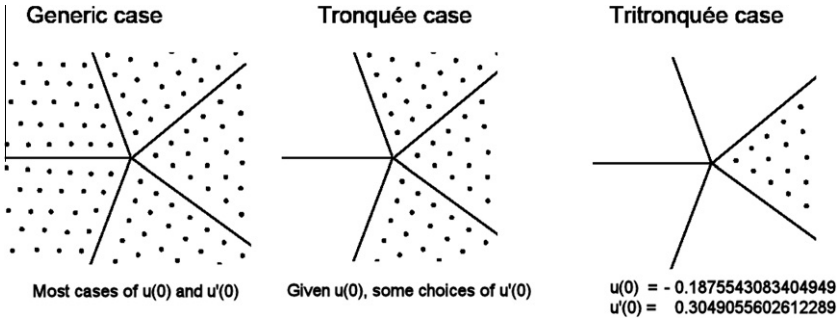

Fig. 1.1. Highly schematic illustration of some pole fields for the P_I equation in case of real-valued ICs at $z = 0$.

not cover the full region of interest in the complex plane, pole-free parts can be obtained by solving ODE boundary value problems (BVPs).

The first numerical initial value problem (IVP) algorithm appropriate for dealing with poles was given by Willers in 1974 [30]. It is here generalized from a single integration path along the real axis to networks of paths throughout regions of the complex plane. The present study is focused on numerical issues, and uses for illustrations the P_I equation

$$\frac{d^2u}{dz^2} = 6u^2 + z. \tag{1.1}$$

In forthcoming studies, we will change the focus towards exploring the solution properties of both the P_I and the higher Painlevé transcendents.

As a background to the present work, we note first a few analytical results for the P_I function. It follows from (1.1) that the ODE is left invariant by the variable changes $u \to \omega^3 u$, $z \to \omega z$ when $\omega^5 = 1$. Related to this result, the pattern of solutions for the ODE form (at sufficiently large distances from the origin) five distinct asymptotic sectors in the complex plane, each of angular width $2\pi/5$. Assuming that $u(0)$ and $u'(0)$ are both real (i.e. $u(z)$ is real-valued along the real axis), the solutions can form pole patterns similar to the very schematic illustrations in Fig. 1.1. Further analytic motivations for these sectors, including their relations to the tronquée (truncated) cases and the regularities within the pole fields, go back to [4], and are also summarized in Section 12.3 of [16]. Within smooth areas, the left-hand side (LHS) of (1.1) becomes negligible for increasing $|z|$, and the two terms in the RHS will balance, leading to the estimate

$$u(z) = \pm\sqrt{-\frac{z}{6}} + o(1). \tag{1.2}$$

More detailed asymptotic expansions were noted already in [4]. However, (1.2) suffices for the present work. In the vicinity of a singularity, local analysis yields

$$u(z) = \frac{1}{(z-z_0)^2} - \frac{z_0}{10}(z-z_0)^2 - \frac{1}{6}(z-z_0)^3 + O(z-z_0)^4, \tag{1.3}$$

(with a second parameter, beyond the pole location z_0, entering at the order $O(z-z_0)^4$). All the singularities are therefore double poles of equal strength and with zero residue.

In the remaining sections of this work, we first introduce the numerical pole field solver. After combining it with a boundary value method, we generate a collection of illustrations displaying some different types of solutions to P_I. This is followed by a discussion of some numerical implementation variations, and by some concluding remarks.

2. Padé method for solving ODE IVPs

One of the key components in the present work is an IVP solver that is able to integrate accurately throughout pole fields of any extent (including right up to poles at unknown locations). We first introduce the relevant issues in the case of the standard first order ODE

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0. \tag{2.1}$$

The most basic numerical scheme for (2.1) is known as Forward Euler. This first order accurate method is obtained by considering only the linear part of a Taylor expansion, i.e.

$$y(t+h) = y(t) + hf(t, y(t)). \tag{2.2}$$

There are three main directions by which this inefficient FE scheme can be improved, leading the classes of (i) Runge–Kutta methods, (ii) Linear multistep methods, and (iii) Taylor expansion methods. The third of these classes forms our starting point.

2.1. Taylor series methods

The idea is to use the ODE to generate, at each step, a much more accurate truncated Taylor expansion than (2.2)

$$y(t+h) = c_0 + c_1 h + c_2 h^2 + c_3 h^3 + \cdots + c_n h^n + O(h^{n+1}). \tag{2.3}$$

Here n is typically in the range of 10–40. A surprisingly large number of text books on numerical methods present only a very ineffective implementation, and then dismiss the concept altogether.

2.1.1. Inefficient version

By Taylor's theorem, $c_n = y^{(n)}(t)/n!$, so all the coefficients can be obtained by repeated differentiation of the ODE $y'(t) = f(t,y(t))$. This process starts:

$$\begin{aligned}
y' &= f,\\
y'' &= f\frac{\partial f}{\partial y} + \frac{\partial f}{\partial t},\\
y''' &= f^2\frac{\partial^2 f}{\partial y^2} + f\left\{\left(\frac{\partial f}{\partial y}\right)^2 + 2\frac{\partial^2 f}{\partial t\partial y}\right\} + \left\{\frac{\partial^2 f}{\partial t^2} + \frac{\partial f}{\partial t}\frac{\partial f}{\partial y}\right\},\\
y'''' &= f^3\frac{\partial^3 f}{dy^3} + f^2\left\{3\frac{\partial^3 f}{\partial t\partial y^2} + 4\frac{\partial f}{\partial t}\frac{\partial^2 f}{\partial y^2}\right\} + f\left\{\left(\frac{\partial f}{\partial y}\right)^3 + 5\frac{\partial^2 f}{\partial t\partial y}\frac{\partial f}{\partial y} + 3\frac{\partial^3 f}{\partial t^2\partial y} + 3\frac{\partial f}{\partial t}\frac{\partial^2 f}{\partial y^2}\right\} + \left\{\frac{\partial^3 f}{\partial t^3} + 3\frac{\partial f}{\partial t}\frac{\partial^2 f}{\partial t\partial y} + \frac{\partial^2 f}{\partial t^2}\frac{\partial f}{\partial y}\right\}.
\end{aligned}$$

The number of terms increases at a rate that quickly makes the approach impractical.

2.1.2. Efficient versions

It also follows from the ODE $y'(t) = f(t,y(t))$ that

$$\frac{dy(t+h)}{dh} = f(t+h, y(t+h)). \tag{2.4}$$

Two effective approaches for determining the coefficients now become available:

1. substitute (2.3) into (2.4) and equate coefficients, and
2. at first, use only one term of (2.3), substitute into (2.4) and integrate, giving two correct terms. For each similar step, one gains one correct coefficient in the expansion.

In many cases, such as for the P_I and the P_{II} equations, the first approach is easy to carry out 'by hand', leading to simple codes for the numerical evaluation of any number of coefficients in (2.3). For more complex ODEs, symbolic algebra systems can be useful. However, the second approach may then be preferable since it always can be implemented entirely numerically in a straightforward manner [2].

2.2. Padé version of the Taylor series method

The Taylor method, like virtually all other numerical ODE approaches, assumes that solutions can be well approximated locally by polynomials. This assumption is violated near poles. Starting from the Taylor method, we therefore take the additional step of converting the finite expansion (2.3) to Padé rational form

$$y(t+h) = \frac{a_0 + a_1 h + a_2 h^2 + \cdots + a_\nu h^\nu}{1 + b_1 h + b_2 h^2 + \cdots + b_{n-\nu} h^{n-\nu}} + O(h^{n+1}). \tag{2.5}$$

Two mathematically equivalent but numerically different conversion strategies are discussed in Section 5.2. We choose n even and take $\nu = n/2$, making the numerator and denominator degrees equal in (2.5). In this rational form, a pole in $y(t+h)$ located near a location t, will just introduce a zero in the denominator of (2.5).

2.2.1. Comparison between Taylor and Padé integration

We choose as a demonstration problem

$$y'(t) = t^2 + y^2, \quad y(0) = 0. \tag{2.6}$$

The analytic solution can be expressed in terms of Bessel functions

$$y(t) = t\, J_{3/4}(t^2/2)/J_{-1/4}(t^2/2),$$

i.e. it has first order poles as its only singularities along the real t-axis. The 14th order Taylor method and its (7,7) Padé counterpart, both using steps $h = 4/27$, are compared in Fig. 2.1. The Taylor method fails when the first pole is approached. In

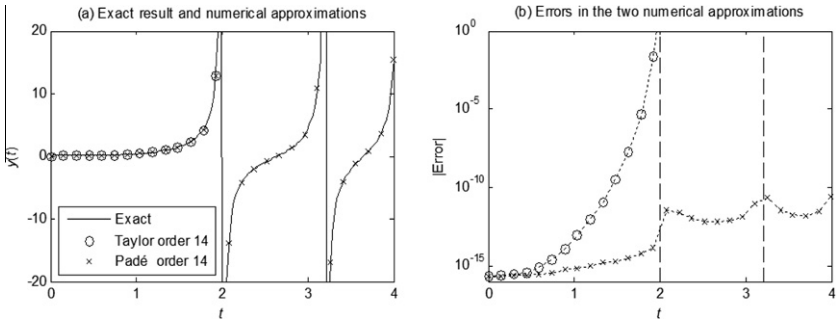

Fig. 2.1. Comparison between the Taylor and Padé methods for Eq. (2.6). Both are based on the same local Taylor expansions to 14^{th} order, using steps of $h = 4/27 \approx 0.15$. (a) Exact solution and numerical approximations. (b) Magnitude of the errors. The dashed vertical lines mark the pole positions.

contrast, the Padé algorithm steps right through any number of these. Some accuracy is lost because, following a pole, small values have been generated by cancellation of larger ones – a well known recipe for losing significant digits. As described in Section 3.1, this loss is avoided in our pole field calculations by choosing continuation paths that reach their goals by following low magnitude passages in-between poles (rather than going directly through them).

3. Computational aspects of the Padé method when applied to 2-D pole fields

Consider again the P_I functions, defined by (1.1). Fig. 3.1 shows a typical example of a computed solution over a region surrounding the point where the initial conditions (ICs) have been provided – here $u(0) = -0.1875$ and $u'(0) = 0.3049$. Since these values are close but not exactly equal to those producing the tritronquée case (cf. Fig. 1.1), there are pole fields in all the five main sectors. However, in four of the sectors, the pole fields have moved some distance away from the origin. This is in good agreement with the pole location calculations in [20] (based on computationally quite expensive high order and high arithmetic precision continued fraction expansions, centered at the origin). By contrast, the present Padé-based IVP solver produced the 161×161 array of data for Fig. 3.1 in about 0.75 sec, using MATLAB on a standard notebook computer.

Tests on three different Windows PC notebooks, with dual core processors of clock speeds 2.8 GHz, 2.0 GHz and 1.34 GHz, respectively, show that relative times on different systems can, to within 10%, be estimated directly from these speeds. Our 'reference computer' for which we quote actual times is the faster of the three (a Fujitsu T4410), when using one core only. With MATLAB's parallel or distributed computing tool boxes, the simple strategy of partitioning the domain (and changing a

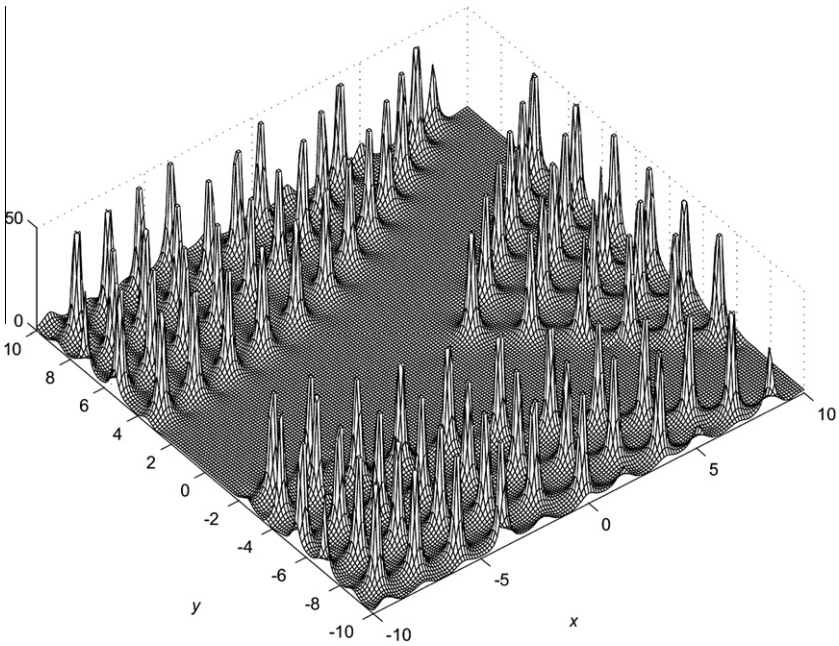

Fig. 3.1. Magnitude of the solution $u(z)$ to the P_I equation in case of ICs $u(0) = -0.1875$, $u'(0) = 0.3049$, displayed over the domain $z = x + iy$, $-10 \leqslant x \leqslant 10$, $-10 \leqslant y \leqslant 10$.

single 'for' statement to a 'parfor' statement) gives speed-ups that become essentially proportional to the number of cores available. For example, each pole field shown in Fig. 4.7a–e is a 5×5 composite of 25 completely independent runs (each starting from the same ICs at $z = 0$, and covering areas of the extent 20×20 in the complex plane). Similarly, Fig. 4.8 is a composite of 18 independent runs. When 'animating' solutions as functions of the ICs (and, for the higher Painlevé equations, also as functions of their parameters), producing each frame in such movie clips would furthermore constitute an independent task.

We next describe some implementation details for the present algorithm and explain how it can be combined with a BVP solver to handle also situations in which there are wide smooth bands between separate pole fields, as well as when smooth sectors extend to infinity.

3.1. Selection of integration paths

The Painlevé property ensures that the solution to (1.1) will remain single-valued, no matter how the integration path wanders around in the complex plane (or divides itself into a tree-like branch pattern). As noted above, numerical cancellations are minimized if one follows paths along which the solution stays small in magnitude. With such integration paths, we need to reach every one of the $161 \times 161 = 25921$ node points seen in Fig. 3.1. These considerations lead us to the following two stage strategy:

1. Create a coarse grid across the domain (in our case with $40 \times 40 = 1600$ nodes) and run a path to near (but not quite up to) each of these target points, and
2. From each end location of the path set just described, take a single step right up to each nearby node on the fine node set.

With regard to Stage 1, Fig. 3.2 illustrates a tree-like structure of paths starting at the origin and with branches reaching close to each of the 1600 nodes on the coarse set, using a step length $h = 0.3$. Initially, one of the nodes was randomly selected, and a path from the origin to it was created as follows. For each step of length h, we consider five choices of direction: one aiming straight at the goal and the remaining ones aiming in directions 22.5° and 45° to either side. The step is then

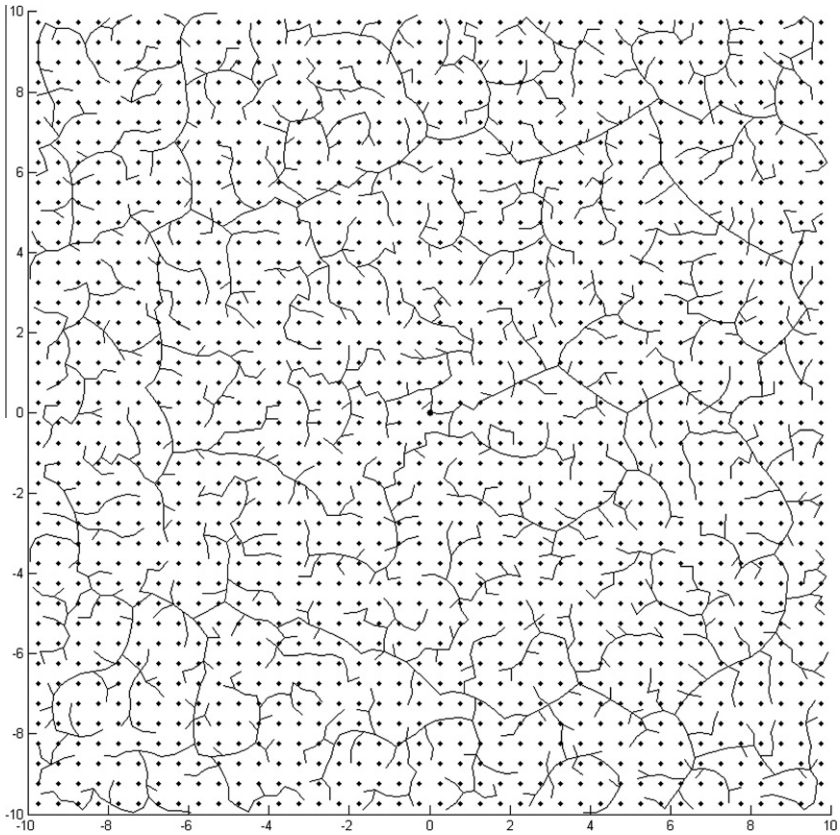

Fig. 3.2. Illustration of integration paths when using steps of length $h = 0.3$, starting from the origin (solid black circle at the center) and reaching near every one of the $40 \times 40 = 1600$ target points.

taken for which the new u value is smallest in magnitude among the five choices. Since the choices were calculated from the same Padé approximation, the additional cost of the multiple evaluations is negligible. Once we reach within a distance of h from the target point, this path is halted. Along such a path, we store for each discrete point not just u and u' but also the Padé coefficients. We next choose, again randomly, a remaining node (target point) and repeat the process, however starting from the closest (discrete) previously visited location (reusing its Padé expansion data). By this strategy, paths cannot cross, and they will quickly reach within h of all target points. In the case illustrated in Fig. 3.2, about 1540 steps were needed for 1600 target points, i.e. on average less than a single step per target point. Changing h from 0.3 to 0.5 (the case used throughout the present work) reduces the number of steps to about 990.

Stage 2 – to step from the end of a path to its nearby node points on the fine grid can be done based on a single Taylor/Padé expansion. The creation of these expansions can be vectorized across all the 1600 cases. Of the computer time of 0.75 sec quoted above, the two stages consume 0.41 sec and 0.34 sec, respectively.

We note in Section 5.5 a situation when a minor modification to the fully random path strategy is called for.

3.2. Numerical solution method for smooth regions

In areas where the solution is smooth, the LHS of (1.1) becomes negligible, and the two terms in the RHS will balance each other. Already tiny changes in $u(z)$ will cause significant changes in $u''(z)$ – an immediate cause for instability when solving an IVP. In this situation, (1.1) is however perfectly well conditioned as a BVP. This allows us to connect an edge of a pole field with (i) another pole field edge, (ii) the smooth far field asymptotic approximation, or (iii) with some other value far out. In all cases, the BVP solver will provide u' values at the two end points. In case (iii) the u and u' values at the distant end locations are all that are needed as ICs for calculating the pole field that is located beyond it. The smooth region between such pole fields can then be completed according to (i).

3.2.1. BVP solvers

Our preferred choice for solving (1.1) as a BVP is Chebyshev (spectral) collocation, as described in [12,26,29]. With BCs $u(z_a) = u_a$ and $u(z_b) = u_b$, the first step is to map the line segment between z_a and z_b linearly to the canonical Chebyshev interval $[-1,1]$, by

$$z = \frac{1}{2}((z_b + z_a) + (z_b - z_a)t), \quad -1 \leqslant t \leqslant 1.$$

The ODE (1.1) is then transformed into

$$\frac{d^2u}{dt^2} = \frac{1}{4}(z_b - z_a)^2\left(6u^2 + \frac{1}{2}(z_b + z_a) + \frac{1}{2}(z_b - z_a)t\right). \tag{3.1}$$

The second step is to introduce collocation points $t_j = \cos(j\pi/N)$, $j = 0,\ldots,N$, which are the extrema of the Chebyshev polynomial of degree N on the interval $[-1,1]$. We denote the approximate function values on this grid by $u_j \approx u(t_j)$, and define the $(N+1)\times 1$ vectors $\boldsymbol{t} = [t_0,\ldots,t_N]^T$, $\boldsymbol{u} = [u_0,\ldots,u_N]^T$, $\boldsymbol{u}^2 = [u_0^2,\ldots,u_N^2]^T$, and $\mathbf{1} = [1,\ldots,1]^T$. The transformed ODE (3.1) can then be approximated by the nonlinear system of equations

$$D_2\boldsymbol{u} = \frac{1}{4}(z_b - z_a)^2\left(6\boldsymbol{u}^2 + \frac{1}{2}(z_b + z_a)\mathbf{1} + \frac{1}{2}(z_b - z_a)\boldsymbol{t}\right), \tag{3.2}$$

where D_2 is the second derivative matrix associated with the nodal set $\boldsymbol{t}$. (It represents the operator obtained by fitting the data $\boldsymbol{u}$ by a polynomial of degree N, differentiating the polynomial twice, and evaluating the result at the nodes $\boldsymbol{t}$; see [12,26,29].)

It remains to enforce the BCs, $u_0 = u_b$ and $u_N = u_a$, as described for example in [29], Section 4, and then to solve the resulting nonlinear system. For this we use Newton's method, since the convergence is fast and the Jacobian is trivial to calculate. The asymptotic solution (1.2) provides a good initial guess, since this BVP approach is only used in regions where the second derivative is negligible. In no case were more than six iterations needed. Solving (3.2) provides approximate solutions u_j at the Chebyshev nodes t_j. We next evaluate the interpolation polynomial at our equispaced lattice points using its barycentric form, which is both fast and numerically stable [3,15]. The total computer time to fill in a smooth band (161 separate BVP solutions; one for each grid line) is generally less than a second. In contrast, it is remarked in [11] that the use of elliptic solvers for a similar task was "computationally expensive".

The pole field solver not only provides boundary values u_a and u_b, but also derivative values u'_a and u'_b. This extra information provides an error diagnostic. Once the approximate solution $\boldsymbol{u}$ has been computed by the BVP solver, derivatives at the endpoints can be estimated via $D_1\boldsymbol{u}$, where D_1 is the first derivative matrix. These derivative values were matched with those of the pole field solver. If the agreement is not to within some tolerance (typically set to 10^{-7} or 10^{-8}), an increase in N, the degree of the interpolant, is indicated.

To assemble the Chebyshev differentiation matrices and to perform the barycentric interpolation, we have used, respectively, the MATLAB codes `chebdif` and `chebint`, from the DMSUITE package [29].

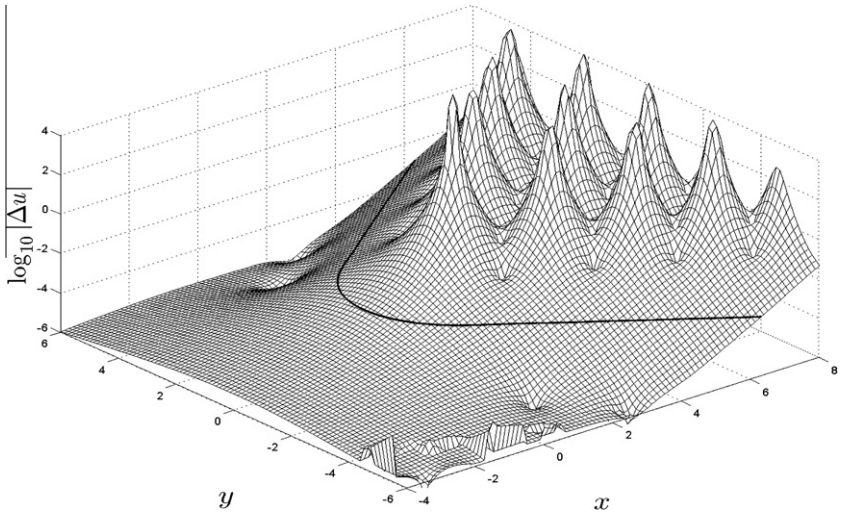

Fig. 3.3. The approximation (3.3) applied to a numerical near-tritronquée solution in the vicinity of the origin (cf. Figs. 1.1c and 3.1). The level curve 0.001 (solid line) is an example of a suitable pole field edge description.

We found another MATLAB package, Chebfun [27], to be quite convenient during the initial explorations. It solves BVPs in essentially the same way as described here, but it offers adaptive sampling to reach a specified accuracy. An automated deflation strategy for computing multiple solutions (cf. Section 4.4) is under development at the time of writing. Computer algebra systems, like Maple and Mathematica, are particularly effective for solving nonlinear BVPs when extended precision arithmetic is required. For example, the tritronquée parameters given in (4.1) were obtained with a 32 decimal digit calculation in Maple. The conveniences offered by Chebfun and Maple come, however, with a sizeable overhead in computational cost. As a result, most BVP calculations in this paper used the direct application of the Chebyshev–Newton method, as described above.

3.2.2. Pole field edge detection

The strategy outlined in Section 3.2 requires us to be able to automatically detect 'edges' of pole fields. Writing $z = x + i\,y$, the meromorphic function $u(z) = u(x,y)$ obeys, away from its poles, Laplace's equation $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$. The residual when applying the 5-point stencil

$$\Delta u \approx \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \frac{u}{h^2} = O(h^2) \tag{3.3}$$

will feature a proportionality constant in the $O(h^2)$ that is much smaller where $u(z)$ is smooth (i.e. away from pole fields compared to within pole fields). Fig. 3.3 shows $\log_{10}|\Delta u|$ when (3.3) has been applied to a numerical solution near the tritronquée case. It is easy to select a contour level (here 0.001) that gives a suitable pole field edge description. It is typical that low level 'ridges' appear within the flat areas, as visible here near the bottom left corner. These irregularities are caused by the fact that different u entries in (3.3) may have been obtained following entirely different paths through the complex plane, and then been amplified by the division by h^2 in (3.3). Since our IVP solver is stable within and near the pole fields, these irregularities will not appear in regions that are of concern in the present context.

4. Some further illustrations of P_I solutions

In this section we test the performance and reliability of our method against some important special cases of the P_I equation.

4.1. The tritronquée case

Fig. 4.1 illustrates a case much closer than the one shown in Fig. 3.1 to the limiting case, for which the upper and lower (not shown) pole fields would have moved out to infinity. The upper end of the solid line marks the location $z = +20i$. The P_I Eq. (1.1) was formulated as a BVP between this point and $z = -20i$, using as end data the leading term of (1.2) (an approach previously used in [11]). From this solution, we obtain in particular $u(z)$ and $u'(z)$ at $z = 20i$ and at $z = 0$. Each of these points can then form starting points for the Padé pole field integrator, allowing accurate calculations of the two pole fields shown.

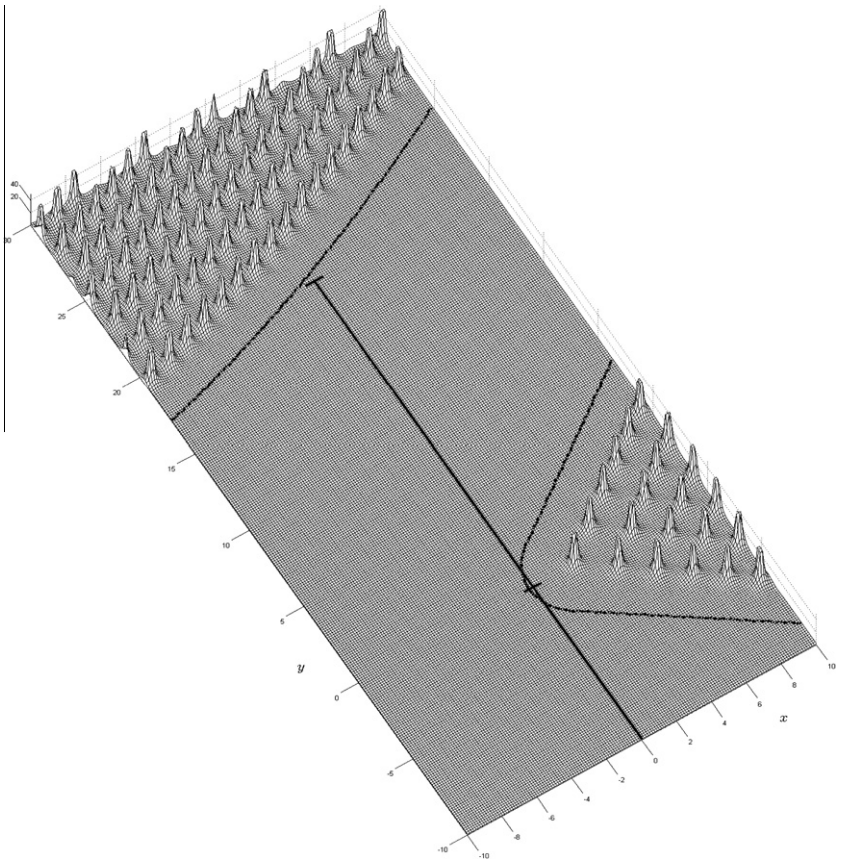

Fig. 4.1. A near-tritronquée case calculated by the steps (i) use a BVP solver over $[-20i, 20i]$ to obtain values for u and u' at $z = 0$ and $z = 20i$ (ii) use this data to separately run out the two pole fields, (iii) fill in the area between pole field edges by further BVP solutions.

The complete solution (accurate to better than 10^{-10}) is then obtained by use of the pole field edge detection algorithm and filling in the smooth band in-between using the Chebyshev–Newton BVP solver.

By placing the BVP's endpoints at $\pm 20i$ and using an extended precision BVP solver in Maple, we find for the tritronquée solution

$$u(0) \approx -0.1875543083404949, \quad u'(0) \approx 0.3049055602612289, \tag{4.1}$$

(cf. approximations previously provided in [17,20]). Already when solving the BVP over $[-20i, 20i]$, using only the rather crude leading term approximation of (1.2), these two values above are obtained to an accuracy of better than 10^{-20}, indicating that the right pole field in Fig. 4.1 is essentially equal to its tritronquée form. The upper pole field is by no means unique, however, and it is here determined by arbitrarily having assigned the value of $u(20i)$ from (1.2).

4.2. NIST Handbook example

Fig. 4.2 a,b (similar to Figs. 32.3.3 and 32.3.4 in [21]) illustrate along the real axis tronquée and near-tronquée solutions for which $u(0) = 0$. Figs. 4.3 and 4.4 show the extensions to the complex plane of the two near-tronquée cases in Fig. 4.2 a. The thick line along the real axis in each of these two figures correspond to the dashed and dot-dashed curves in Fig. 4.2 a. A short cross-line along the imaginary axis indicates the location of the origin. As the value of $u'(0)$ approaches the tronquée case from below, the left pole field seen in Fig. 4.3 is pushed infinitely far out to the left and it then returns again with the key difference that, on the return, there is a row of poles along the negative real axis (rather than the negative real axis passing between two adjacent rows of poles). During this process, there are no noticeable changes occurring in the right side pole field.

4.3. Overview of tronquée solutions

The tronquée solutions that were illustrated (along the real axis only) in Fig. 4.2 b are just two out of infinitely many such cases for which $u(0) = 0$ (however the only ones without some poles also just to the left of the origin). Fig. 4.5 shows, in an $(u(0), u'(0))$-plane, all the ICs that produce tronquée solutions. The two cases just mentioned correspond to the intersections

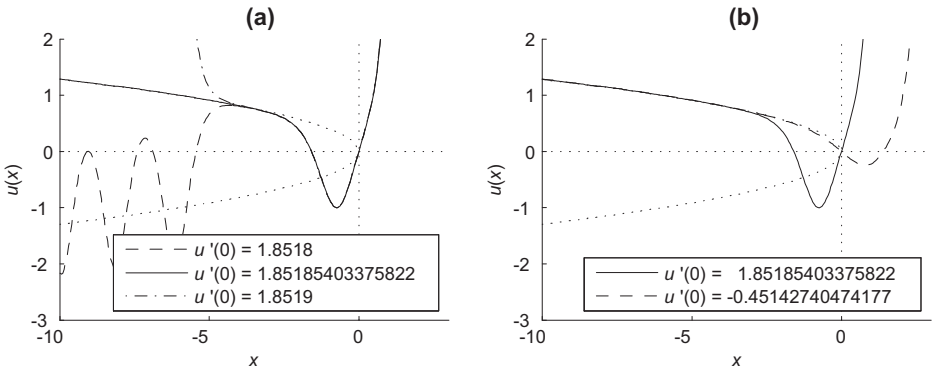

Fig. 4.2. Solutions along the real axis obeying $u(0) = 0$: (a) One of the tronquée cases together with two solitions for similar values for $u'(0)$ near one of the tronquée cases, (b) The two tronquée cases. In both subplots, the dotted curves correspond to the leading term $\pm\sqrt{-z/6}$ in (1.2).

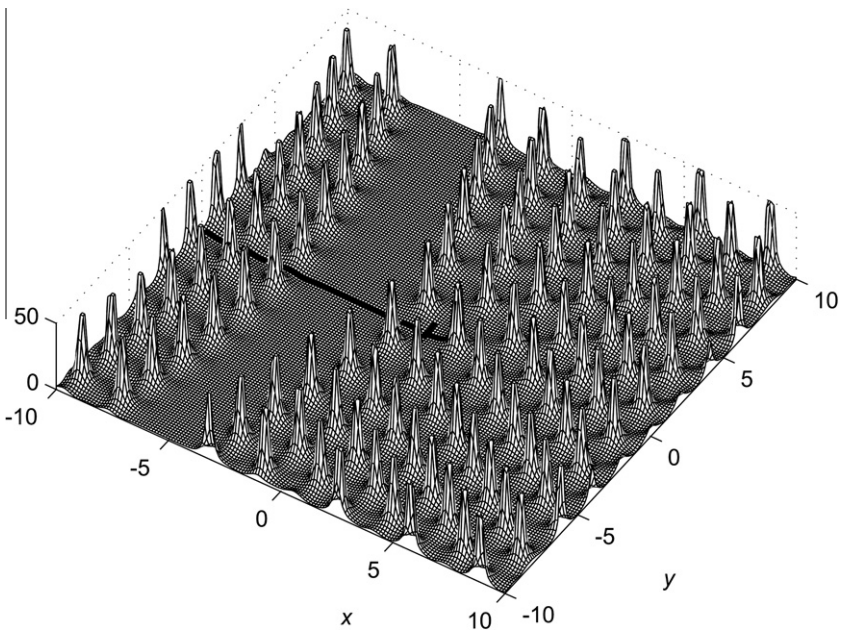

Fig. 4.3. NIST Handbook example, $u(0) = 0$, $u'(0) = 1.8518$.

between the innermost curve and the vertical axis (defined by $u(0) = 0$). We discuss at the end of Section 4.4 a convenient way of generating Fig. 4.5.

4.4. Creation of a smooth band far out in left half-plane

Similarly to how we used the BVP solver along a section of the imaginary axis in Section 4.1, we can do the same along a section of the negative real axis. If we arbitrarily decide on $z \in [-18, -14]$ and use as initial guess the values $u = \sqrt{-z/6}$ apart from a too large value at $z = -18$ and a too small one at $z = 14$, as shown by circles in Fig. 4.6 a, the Chebyshev–Newton scheme immediately converges to the solution shown by the dashed line. Simply reading off the resulting value for $u'(-18)$ and $u'(-14)$ allows us to run out the two pole fields and, again using the BVP solver, stably fill in the smooth band in-between, with the result seen in the main part of the figure. Choosing a u value at $z = -18$ lower than $\sqrt{-z/6}$ similarly produces a left pole field without poles further out along the negative real axis, and choosing a u value at $z = -14$ higher than $\sqrt{-z/6}$ a right pole field with a front-row pole on the real axis. All together, we can thus combine four different types of solutions with the same width of the smooth band. Tronquée solutions with the right pole field extending far along the negative real axis were analyzed already in [4]. There does not appear to be any restrictions on where bands of this type can be placed in the left half-plane. In the right half-plane, the situation is different since, along the positive real axis, the two terms in the RHS of (1.1) will never approximately cancel each other.

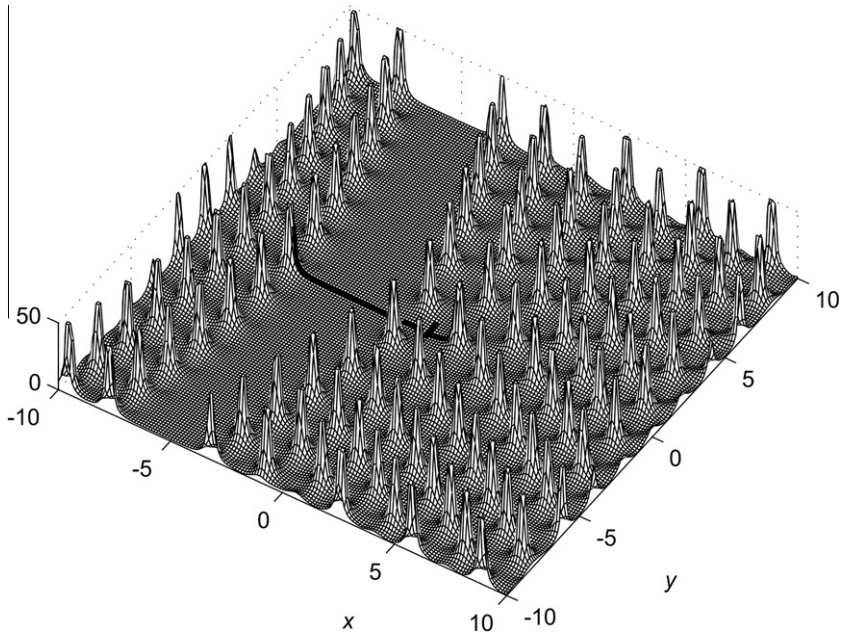

Fig. 4.4. NIST Handbook example, $u(0) = 0$, $u'(0) = 1.8519$.

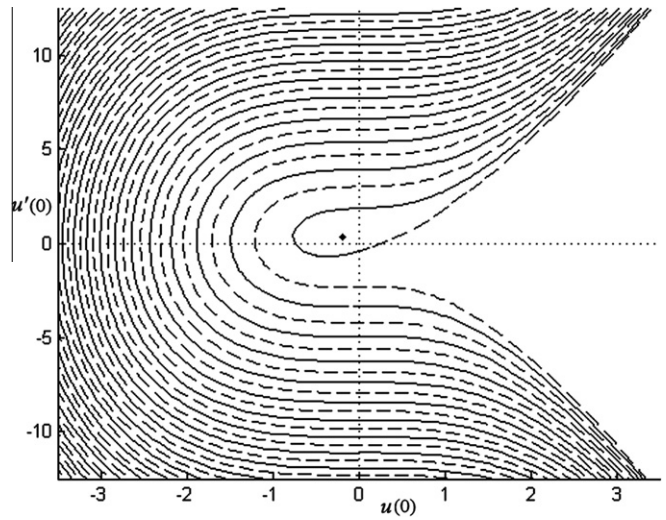

Fig. 4.5. Initial conditions $u(0)$, $u'(0)$ that lead to tronquée solutions. For curves marked as solid lines, the first nontrivial feature encountered when descending the real z-axis is a gap between two poles, else dashed if it is a pole. An isolated dot marks the tritronquée case.

This idea for creating smooth bands was used to produce the tronquée display in Fig. 4.5. We generated smooth bands that extend from $z = -\infty$ and up to some point that is moved to the right along the z-axis and, during this process, recorded the values that the pole field solver gives for $u(0)$ and $u'(0)$.

4.5. Transitions in pole field patterns

Asymptotic theory requires that, sufficiently far out from the origin, the P_I solutions feature the sectorial structure indicated in Fig. 1.1, with the sector boundaries as shown (independently of whether the ICs at $z = 0$ are real or not). Since the solutions locally around the origin can be of different form (for example with a sixfold rather than a fivefold sector pattern if both $u(0)$ and $u'(0)$ are large in magnitude), some forms of transitions must occur. Fig. 4.7a–f provide some illustrations. Two cases agree with pole fields already displayed; subplot (b) with Fig. 4.3 and subplot (e) with Fig. 3.1. We note that, with increasing distances from the origin, the pole fields tend to become denser (consistent with the estimate that every P_I

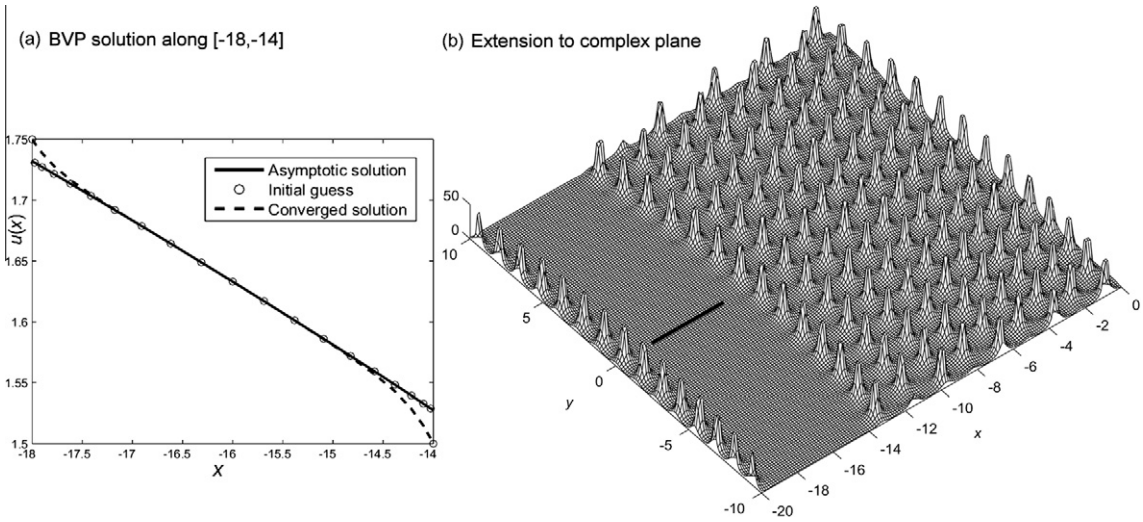

Fig. 4.6. Solution associated with the BVP described in Section 4.4.

solution will have $O(R^{5/2})$ poles within a radius of R from the origin [16]), and the smooth bands between them become narrower. Fig. 4.8 illustrates that changes in pole patterns need neither be gradual, nor be associated with well defined smooth bands. With the ICs $u(0) = -5$, $u'(0) = 0$, a sharp pattern transition is visible near $\mathrm{Re}(z) = -60$.

5. Discussion of implementation options

Several choices need to be made when converting a numerical concept into efficient code. Some of these (such as the order of accuracy in the Padé-based integrator) relate mostly to the numerical algorithm itself, whereas other choices depend more on the software system that is used for the numerical implementation.

5.1. Choice of order for the Padé integrator

The next two subsections motivate our standard choice of *order* = 30 (i.e. a (15,15) Padé expansion) together with a step size of $h = 0.5$.

5.1.1. Test problem – Weierstrass $\wp$-function

In order to obtain a test problem with the same numerical issues as (1.1) within pole fields, but with a known solution, we drop the z-term and obtain the ODE

$$\frac{d^2u}{dz^2} = 6u^2, \tag{5.1}$$

for which the general solution is given by the Weierstrass $\wp$-function [[13], Ch. 23]

$$u(z) = \wp(z + c_1, 0, c_2). \tag{5.2}$$

This is an elliptic (doubly periodic meromorphic) function with second order poles on a rhombic lattice composed of equilateral triangles. We pick $c_1 = -1$, $c_2 = 2$, which determines the initial conditions

$$u(0) \approx 1.071822516416917, \quad u'(0) \approx 1.710337353176786.$$

On the real axis, the poles are located at $x = 1 + 2\omega k$, $k = 0, \pm1, \pm2, \ldots$, where $\omega = \Gamma\left(\frac{1}{3}\right)^3/(2^{13/6}\pi) \approx 1.363$.

We compute below, using numerical ODE solvers, the solutions at $x = 30$ (medium range) and at $x = 10^4$ (long range), for which the reference values are

$$u(30) = 1.095098255959744, \quad u(10^4) = 21.02530339471055. \tag{5.3}$$

For some of the higher Painlevé transcendents there may be no 'companion' solution, like the Weierstrass $\wp$-function in the case of P_I. In those cases, accuracy should be checked by alternate means. One strategy we recommend is to compute a pole field with real initial data on the real axis. The theoretical solution in the upper and lower half-planes should be identical (up to conjugation) but, because of the random nature of our path selection algorithm, this will not be the case for the numerical

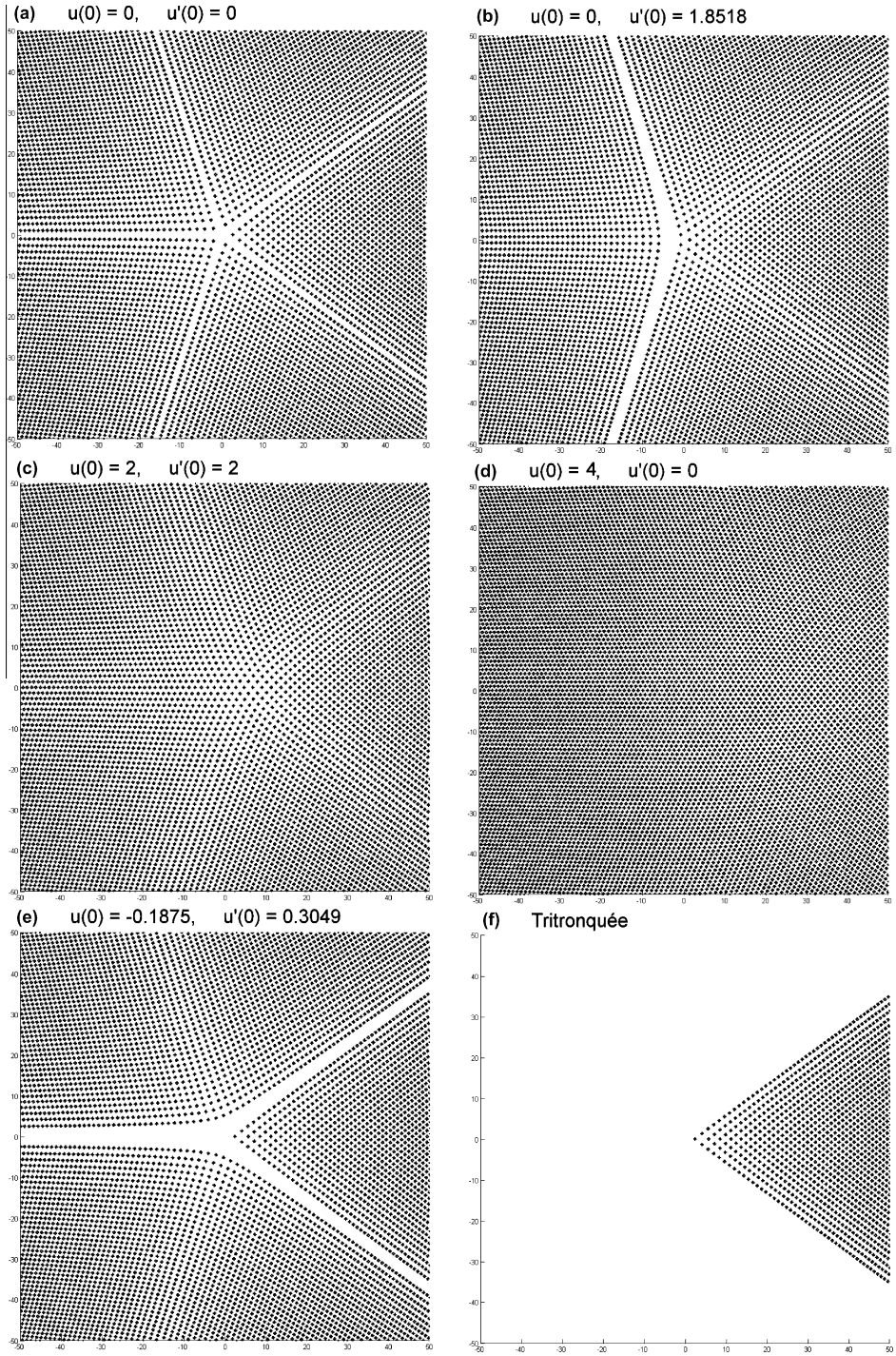

Fig. 4.7. Pole locations displayed over the region $[-50,50] \times [-50,50]$ for six different choices of initial conditions at $z = 0$. For the tritronquée case (subplot f), see (4.1) for the values of $u(0)$ and $u'(0)$.

solution. The difference between the upper and lower half-plane solutions gives a reasonable estimate for the numerical error and is good enough for determining practical choices of the parameters *order* and *h*. An alternative is to run a pole field twice in succession with exactly the same initial data (not necessarily real). Again, the random nature of the pole field algorithm ensures that the difference between the two solutions can serve as a practical accuracy indicator.

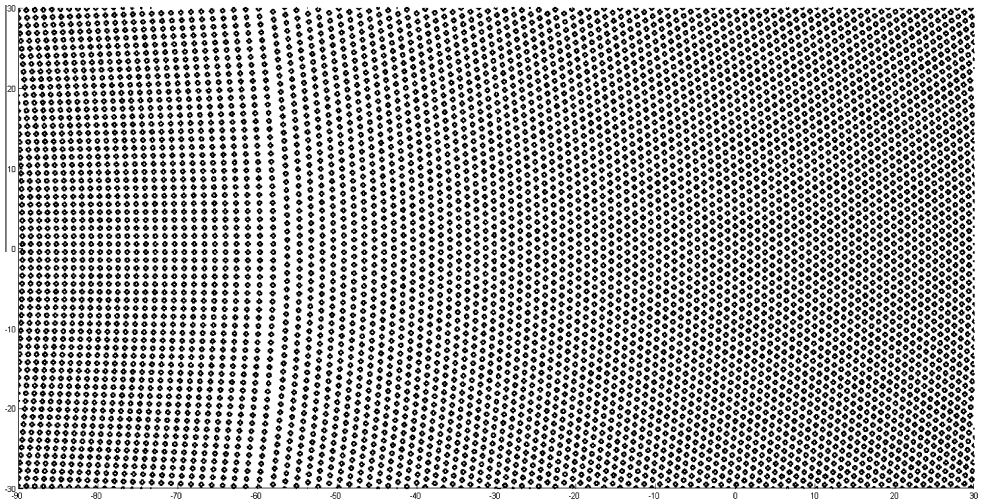

Fig. 4.8. Pole locations displayed over the region $[-90,30] \times [-30,30]$ in the case of ICs $u(0) = -5$, $u'(0) = 0$.

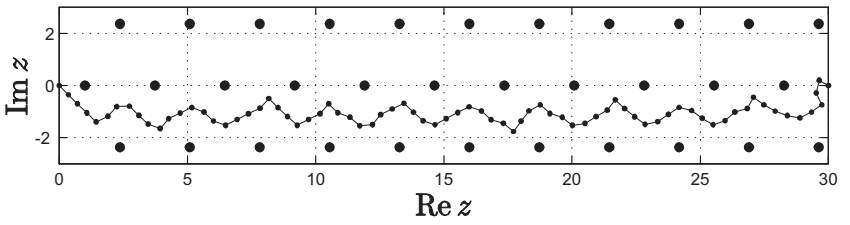

Fig. 5.1. Poles of the Weierstrass function (5.2) (large dots) along with the path taken by the Padé integrator (*order* = 30, $h = 0.5$). The absolute error at $x = 30$ is $8.34 \cdot 10^{-14}$ and the relative error (also shown in Table 5.1) is $7.62 \cdot 10^{-14}$.

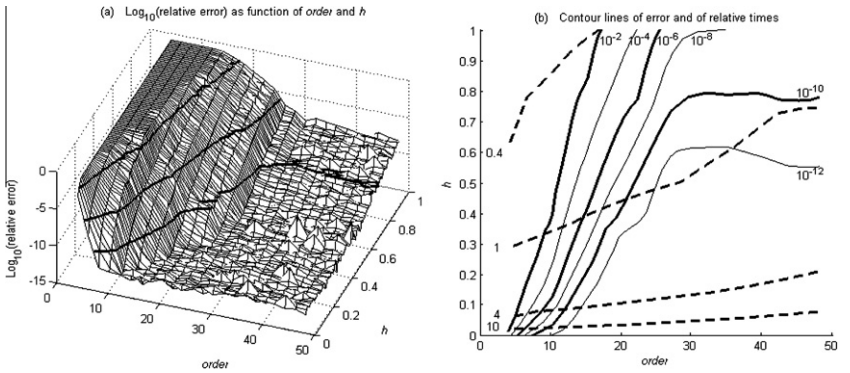

Fig. 5.2. (a) Accuracy of the Padé integrator for the test calculation over $[0,30]$ as function of h and *order*. The surface is truncated where the relative error exceeds 10^{-1}. (b) Contour lines for equal accuracy (solid lines; level values displayed along top and right edges) and for equal computer time (dashed lines; level values displayed along the left edge), indicating a particularly favorable region around *order* = 30 and $h = 0.5$. The three thicker contour lines for equal accuracy in this subplot (at levels 10^{-2}, 10^{-6} and 10^{-10}) correspond to the three level curves shown in part (a).

5.1.2. Accuracy and time comparisons

Fig. 5.1 shows the pole pattern for the test case just described, and also the integration path taken by the Padé integrator in the 'medium range' test case. Fig. 5.2a shows how the numerical error for the test problem above, measured at $z = 30$, varies with the two parameters *order* and h. The trade-off for increasing *order* and decreasing h is increased computer time. Fig. 5.2b displays contour lines for both relative computer times and relative errors. For best clarity, some fine-scale 'jitter' (visible in the raw data in Fig. 5.2a) was smoothed out when we created these contours. For conversion to actual computer times, the relative time "1" corresponds on our reference computer to 0.020 sec. The intersections of the different contours suggest that our standard choice of *order* = 30 and $h = 0.5$ represents a generally favorable combination of the two measures.

5.2. Toeplitz-based implementation vs. the ϵ–algorithm

There are several methods available for computing Padé approximations (2.5) from truncated Taylor expansions (2.3). The most direct is to multiply both expressions with the denominator in (2.5) and then equate coefficients. The coefficients $b_1,\ldots,b_\nu$ for the denominator then follow from solving

$$\begin{pmatrix} c_\nu & c_{\nu-1} & \cdots & c_1 \\ c_{\nu+1} & c_\nu & & c_2 \\ \vdots & & \ddots & \vdots \\ c_{2\nu-1} & c_{2\nu-2} & & c_\nu \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_\nu \end{pmatrix} = - \begin{pmatrix} c_{\nu+1} \\ c_{\nu+2} \\ \vdots \\ c_{2\nu} \end{pmatrix}, \tag{5.4}$$

after which the numerator is obtained from $a_0 = c_0$ followed by

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_\nu \end{pmatrix} = \begin{pmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & & 0 \\ \vdots & & \ddots & \vdots \\ c_{\nu-1} & c_{\nu-2} & & c_0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_\nu \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_\nu \end{pmatrix}. \tag{5.5}$$

Opportunities that could possibly be exploited in Fortran or C codes, but which are unlikely to offer any advantages in MATLAB for the current small system sizes, include:

- Reversing the order of the unknowns in (5.4), giving a symmetric (although not generally positive definite) linear system.
- Using 'fast' Toeplitz (or Hankel) solvers, see e.g. [28]. Although, in theory, the operation count is reduced from $O(\nu^3)$ to $O(\nu^2)$ or faster, savings are unlikely to be seen for the relatively small present values of ν (in particular in view of MATLAB's very efficient implementation of the backslash ("\") operator for general linear systems).

It is well known that the condition number of the linear system (5.4) can grow rapidly with increasing ν [1]. In spite of this, one typically obtains a rational function that is far more accurate than the coefficients of the denominator polynomial. Partial explanations and further discussions of this are given in [5,19]. Especially since we are using only matrices of moderate size, we have not found the ill-conditioning to be a concern.

In rare cases, it can happen that (5.4) becomes outright singular. In the present context, this has only been observed in cases where the Taylor coefficients are very rapidly decreasing in size (or a number of them become exactly zero). When this occurs, we remove the last row of (5.4). Matlab's "\" solver will then give the solution to the underdetermined system that has the smallest norm, thereby restoring numerical stability. Although this version actually runs faster than using a square system, we did not use it routinely since it slightly reduces the accuracy in the opposite extreme case (with very rapidly growing Taylor coefficients, as encountered near poles).

A different option is to use the ϵ–algorithm (or the essentially equivalent η–algorithm, used in [30]). The Padé approximation can then be evaluated in $O(\nu^2)$ operations directly from the corresponding Taylor expansion [14,31]. However, there is no built-in version in MATLAB of the types of recursions involved, and tests showed that a 'for loop'-based implementation is about 10 times slower for the present problem sizes (when taking into account that separate evaluations are needed for $u(z+h)$ and $u'(z+h)$, whereas the Toeplitz approach can compute the derivative by the quotient rule of calculus). It should be noted that the ϵ- and η- algorithms, in which the system (5.4) is never formed, do not escape ill-conditioning issues. These algorithms are based on quotient-difference formulas, and the differencing of quantities of nearly equal magnitude causes instabilities.

Since we use MATLAB, our method of choice for computing Padé approximations has been the Toeplitz approach and solving the system (5.4) with the backslash operator.

5.3. Choice of directions for the ODE steps

The original 'pole vaulting' idea, proposed in [8], involved making semicircular or rectangular temporary path diversions into the complex plane. The present work extends the idea in three fundamental ways: (i) use of a 'pole friendly' Padé integrator, (ii) not using any rigid choices for the diversion paths, but instead use a freely branching network of paths throughout the complex plane, and (iii) targeting the paths towards whole regions in the complex plane rather than only towards other real axis locations. A key issue that arises is how to choose the direction to take for each step. Our strategy for this was described in Section 3: evaluate in five directions all roughly aiming towards the goal and select the lowest choice in terms of $|u(z)|$ (cf. additional discussion in Section 5.4.1).

5.4. Comparisons between some different ODE solvers

The tests in this section compare our Padé integrator against other ODE solvers. Since the circumstances are quite different for the two 'stages' (cf. Section 3.1), we discuss them separately. In typical pole field calculations, we noted above that the computer times for the two stages were somewhat comparable when using the Padé method. We will see here that a high order Runge–Kutta–Nyström (RKN) method can be competitive for Stage 1, but that the Padé approach is far superior for Stage 2.

5.4.1. Stage 1: Integration along 'valleys', and near a pole at a known location

We consider again the simplified ODE (5.1). The path selection strategy described in Section 3.1 remains unchanged, but on each segment of length h, a standard ODE solver is used rather than the Padé integrator. As alternate solvers to compare against, we chose MATLAB's standard ode45 solver and a 12th order Runge–Kutta–Nyström method (RKN12) [9,10], respectively.

In the RKN12 method, a step size $h_1 < h$ is used, where $h_1 = h/M_1$. That is, one step of the Padé integrator is emulated by M_1 steps of the RKN12 method. In the cases where the solution is required high up on the wall of a pole, however, it becomes necessary to decrease the RKN12 step size for the final ascent step in order to preserve accuracy. In that case, we use a step size $h_2 = d/M_2$, where $M_2 \gg M_1$ and d is the length of the final path segment.

With the Padé integrator, the cost penalty is very small for integrating in five directions (rather than only in one direction), in order to select an approximate minimum modulus direction for the next step. For the RKN12 method we could emulate this to some degree, by vectorizing the MATLAB code so that it integrates in all five directions simultaneously. This is not possible with the ode45 code and, for a fair comparison, we simply reduced the computational times recorded in Table 5.1 by a factor of five.

Relatedly, one could modify the path selection algorithm so that the approximate minimum modulus direction is determined without actually integrating the ODE, as alluded to at the end of Section 5.3. Such a strategy for finding the step direction at $z = z_0$ might be the preferred choice when the Padé algorithm is implemented in Fortran or C. We have implemented it as follows: Consider first the same 90° wedge as described in Section 3.1. Rather than integrating along five evenly spaced rays in this wedge, we compute instead the steepest descent direction for $|u(z)|$, i.e. $\theta = \arg(-u(z_0)/u'(z_0))$. If this direction falls inside the wedge, we accept it as the direction of the next step. If not, we choose the edge of the wedge closest to this steepest descent direction as the step direction. When combined with the RKN12 method, this method is denoted RKN12† in Table 5.1.

Table 5.1 shows relative errors at three values of z, namely (a) $z = 30$ (medium range, low valley run; target point seen in Fig. 5.1), (b) $z = 10^4$ (long range, low valley run), and (c) $z = 28.261$ (medium range, but with the end point high up on a pole 'wall'; the rightmost real pole in Fig. 5.1). The reference values for (a) and (b) were given in (5.3). For case (c), we have

$$u(28.261) = 9.876953517025014 \times 10^6.$$

For the Padé integrator, we used in all cases $h = 0.5$ and *order* = 30, as discussed in Section 5.1.2. For the RKN12 and ode45 methods we used $h = 0.5$ as well. We experimented with the parameters M_1 and M_2 in the RKN12 methods in order to obtain roughly the same accuracy as with the Padé method. These values, as well as the error tolerances for ode45, are listed in the caption of the table. For the timing results given in Table 5.1, we took the median cpu time over 100 repetitions of the test.

The numerical results show that ode45's computational times exceed those of the other methods by orders of magnitude, which is the reason it is not recorded in the long valley run in Table 5.1(b). In addition, it has difficulties meeting the specified error tolerances, especially near a pole, as seen in Table 5.1(c).

The Padé and the two RKN12 methods featured comparable speeds and accuracies in the low valley runs, but the Padé method was notably faster in the vicinity of a single pole. It should be kept in mind that in Table 5.1(c) it was known a priori that the target was near a pole, and hence the value of M_2 could be increased in an optimal manner.

Symplectic ODE schemes are also applicable because the P_l equation can be expressed as a Hamiltonian system; see [[21], Section 32.6]. We tested the Störmer–Verlet scheme, but found it not to be competitive, owing to its low (second) order of convergence. There exist higher order symplectic Runge–Kutta–Nyström schemes but, to quote from the abstract of [6]: "The authors construct and test symplectic, explicit Runge–Kutta–Nyström (RKN) methods of order 8. The outcome of the inves-

Table 5.1
Relative error and cpu times for various methods for computing $\wp(z-1,0,2)$. The RKN12 parameters are (a) $M_1 = M_2 = 5$, (b) $M_1 = M_2 = 5$, (c) $M_1 = 5$, $M_2 = 3000$, and the absolute and relative error tolerances of ode45 are both set to 10^{-12}.

	(a) $z = 30$		(b) $z = 10^4$		(c) $z = 28.261$	
	Rel. Err.	CPU Time	Rel. Err.	CPU Time	Rel. Err.	CPU Time
Padé	7.62(−14)	0.020	2.34(−10)	26.5	7.92(−10)	0.018
RKN12	1.74(−13)	0.023	1.44(−09)	38.0	9.61(−10)	0.12
RKN12†	1.02(−13)	0.020	4.11(−09)	15.6	1.71(−09)	0.11
ode45	1.47(−11)	1.36	—	—	4.30(−08)	1.44

tigation is that existing high-order, symplectic RKN formulae require so many evaluations per step that they are much less efficient than conventional eighth-order nonsymplectic, variable-step-size integrators even for low accuracy." Finally, symplectic schemes require fixed step sizes, whereas our path strategy requires frequent direction changes, thereby violating the conditions for symplecticness.

5.4.2. Stage 2: Evaluations at lattice points, with poles present at unknown locations

In this case, described as 'Stage 2' in Section 3.1, the advantage of the Padé approach becomes overwhelming. For each of the $40 \times 40 = 1600$ 'Stage 1' end locations, one single Padé step suffices to reach its associated lattice points on the fine grid – with no tests needed to check for nearby poles. The necessary Taylor and Padé expansions can be computed using vector operations of that length (1600). While around 900 Padé steps in 'Stage 1' take about 0.4 sec, over 25000 Padé steps in 'Stage 2' are completed in less time than that. In contrast, the RKN12 (or any other) approach cannot be vectorized similarly, since start and end points for the last step might be on the opposite side of a pole, necessitating case-by-case tests followed by local path and step size adjustments. Instead of greatly improving in speed, each step would now be significantly slowed down. Hence, we did not write the code to apply any alternative to the Padé method for 'Stage 2'.

5.5. 'Tuning' of random path selection strategy

In cases where both a pole field(s) and a smooth area are present in the domain of interest, it is important to prevent the pole field solver from wandering into the smooth region and then stepping back again into the same pole field. As noted above, smooth regions are unstable regions for any IVP solver, and the associated loss of accuracy ought not be carried back into the pole field. One way to prevent this is to force the path selection algorithm to complete pole fields before stepping into smooth regions. In the present test cases, we implemented this 'manually' after having inspected, in preliminary test calculations, where the pole field edges appeared. However, with the pole field edge detection procedure described in Section 3.2.2, this process can readily be fully automated.

6. Conclusions

In this study we have demonstrated that the pole fields of Painlevé-type equations offer excellent opportunities for accurate and computationally very fast numerical solutions. When supplemented with appropriate BVP solvers, accurate and fast computational solutions become available also across pole-free areas. Although we have here only presented results for the P_I equation, the numerical approach has been found to work equally well in preliminary computations with the P_{II} equation. We have every reason to believe this will be the case also for the $P_{III} - P_{VI}$ equations.

The present approach is based on a very general 'pole vaulting' strategy combined with the use of a Taylor/Padé-based IVP solver. Comparisons against some alternative approaches (such as different IVP solvers, different ways to convert Taylor expansions to Padé form, etc.) show that, while some variations also might work well in certain cases, the chosen method constitutes a simple and highly competitive overall approach – indeed the first one to permit accurate numerical solutions across extended regions of the complex plane.

We remark that an entirely different approach to solving the Painlevé II equation was recently presented in [22]. Based on a Riemann–Hilbert formulation of the problem, this approach evaluates integrals rather than solves differential equations. The displayed computed solutions include some poles, but are limited to relatively small values of $|x|$ (the interval $[-8,8]$), as the author reported a loss of accuracy and a deterioration of condition number as $|x|$ increases.

There are many opportunities for future work along the lines of the present study. These include:

- Compare numerical calculations against available theoretical results for the different Painlevé equations.
- Explore the Painlevé equations for yet unseen solution features.
- Create animations of how the Painlevé solutions evolve when parameters and/or initial conditions are varied.
- Develop a general purpose software package for Painlevé-type equations.
- Utilize the random path feature of the present scheme for numerically testing whether ODEs (maybe of higher order, and/or systems of ODEs) have the Painlevé property (based on the principle that different runs, following different continuation paths, would give different values if branch points are present).

Acknowledgements

The work of Bengt Fornberg was supported by the NSF Grants DMS-0611681 and DMS-0914647. Part of it was carried out in the fall of 2010 while he was an Oliver Smithies Lecturer at Balliol College, Oxford, and also a Visiting Fellow at OCCAM (Oxford Centre for Collaborative Applied Mathematics). The latter is supported by Award No. KUK-C1-013-04 to the University of Oxford, UK, by King Abdullah University of Science and Technology (KAUST). André Weideman was supported by the National Research Foundation of South Africa. He acknowledges the hospitality of the Numerical Analysis Group at the Mathematical Institute, Oxford University, during a visit in November 2010. We thank Jonah Reeger for creating Fig. 4.5. Discus-

sions with Peter Clarkson, John Ockendon, Nick Trefethen, Ben Herbst, Bryce McLeod and Rodney Halburd are gratefully acknowledged. The present work was stimulated by the workshop 'Numerical solution of the Painlevé Equations', held in May 2010 at the International Center for the Mathematical Sciences (ICMS), in Edinburgh.

References

[1] G.A. Baker Jr., P.R. Graves-Morris, Padé approximants, Encyclopedia of Mathematica and its Applications, second ed., 59, Cambridge Univ. Press, Cambridge, 1996.
[2] D. Barton, I.M. Willers, R.V.M. Zahar, An implementation of the Taylor series method for ordinary differential equations, Comput. J. 14 (1971) 243–248.
[3] J.P. Berrut, L.N. Trefethen, Barycentric Lagrange interpolation, SIAM Rev. 46 (2004) 501–517.
[4] P. Boutroux, Recherches sur les transcendantes de M. Painlevé et l'étude asymptotique des équations différentielles du second ordre, Ann. École Norm. 30 (1913) 255–375.
[5] O.P. Bruno, F. Reitich, Approximation of analytic functions: a method of enhanced convergence, Math. Comp. 63 (1994) 195–213.
[6] M.P. Calvo, J.M. Sanz-Serna, High-order symplectic Runge–Kutta–Nyström methods, SIAM J. Sci. Comput. 14 (1993) 1237–1252.
[7] P.A. Clarkson, Painlevé Equations – Nonlinear Special Functions, Lecture Notes in Mathematics, 1883, Springer, Berlin, 2006.
[8] G.F. Corliss, Integrating ODE's in the complex plane - Pole vaulting, Math. Comput. 35 (1980) 1181–1189.
[9] J.R. Dormand, M.A.E. El-Mikkawy, P.J. Prince, High-order embedded Runge–Kutta–Nystrom formulae, IMA J. Num. Anal. 7 (1987) 423–430.
[10] J.R. Dormand, P.J. Prince, Runge–Kutta–Nystrom triples, Comput. Math. Appl. 13 (1987) 937–949.
[11] B. Dubrovin, T. Grava, C. Klein, On universality of critical behaviors in the focusing nonlinear Schrödinger equation elliptic umbilic catastrophe and the tritronquée solution to the Painlevé-I equation, J. Nonlinear Sci. 19 (2009) 57–94.
[12] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, 1996.
[13] B. Gambier, Sur les équationes différentielles du second ordre et du premier degré don't l'intégral générale est à points critique fixés, Acta Math. 33 (1910) 1–55.
[14] P.R. Graves-Morris, D.E. Roberts, A. Salam, The epsilon algorithm and related topics, J. Comput. Appl. Math. 122 (2000) 51–80.
[15] N.J. Higham, The numerical stability of barycentric Lagrange interpolation, IMA J. Num. Anal. 24 (2004) 547–556.
[16] E. Hille, Ordinary differential equations in the complex domain. Reprint of the 1976 original. Dover Publications, Inc., Mineola, NY (1997).
[17] N. Joshi, A. Kitaev, On Boutroux's tritronquée solutions of the first Painlevé equation, Stud. Appl. Math. 107 (2001) 253–291.
[18] M.D. Kruskal, P.A. Clarkson, The Painlevé–Kowalevski and poly-Painlevé tests for integrability, Stud. Appl. Math. 86 (1992) 87–165.
[19] Y.L. Luke, Computations of coefficients in the polynomials of Padé approximations by solving systems of linear equations, J. Comput. Appl. Math. 6 (1980) 213–218.
[20] V.Yu. Novokshenov, Padé approximations for Painlevé I and II transcendents, Theor. Math. Phys. 159 (2009) 853–862.
[21] F.W.J. Olver, D.W. Lozier, R.F. Boisvert, C.W. Clark, NIST Handbook of Mathematical Functions, Cambridge University Press, 2010.
[22] S. Olver, Numerical solution of Riemann–Hilbert Problems: Painlevé II, Found. Comput. Math. 11 (2011) 153–179.
[23] P. Painlevé, Mémoire sur les équationes differentielles dont l'intégrale générale est uniforme, Bull. Soc. Math. 28 (1900) 201–261.
[24] P. Painlevé, Sur les équationes différentielles du second ordre et d'ordre supérieur don't l'intégrale générale est uniforme, Acta Math. 21 (1902) 1–85.
[25] H. Qin, Y. Lu, A note on an open problem about the first Painlevé equation, Acta Math. Appl. Sin. English Ser. 24 (2008) 203–210.
[26] L.N. Trefethen, Spectral Methods in MATLAB, SIAM, Philadelphia, 2000.
[27] L. N. Trefethen, N. Hale, R.B. Platte, T.A. Driscoll, R. Pachón, Chebfun Version 3, http://www.maths.ox.ac.uk/chebfun/, University of Oxford (2009).
[28] M. Van Barel, G. Heinig, P. Kravanja, A stabilized superfast solver for nonsymmetric Toeplitz systems, SIAM J. Matrix Anal. Appl. 23 (2001) 494–510.
[29] J.A.C. Weideman, S.C. Reddy, A MATLAB differentiation matrix suite, ACM TOMS 26 (2000) 465–519. See also http://dip.sun.ac.za/~weideman/research/differ.html.
[30] I.M. Willers, A new integration algorithm for ordinary differential equations based on continued fraction approximations, Comm. ACM. 17 (1974) 504–508.
[31] P. Wynn, The epsilon algorithm and operational formulas of numerical analysis, Math. Comput. 15 (1961) 151–158.