ODE Solving via Automatic Differentiation and Rational Prediction

Andreas Griewank*

December 20, 1995

Abstract

We consider the classical Taylor series approximation to the solution of initial value problems in ordinary differential equations and examine implicit variants for the numerical solution of stiff ODEs. The Taylor coefficients of the state vector are found to be closely related to those of the Jacobian of the right hand side along the solution trajectory. These connections between state and Jacobian coefficients are exploited for their efficient evaluation by automatic differentiation with a small number of forward and reverse sweeps. It is shown how these coefficients can be utilized in a new rational predictor for the Hermite-Obreshkov-Padé (HOP) methods, a family of high order numerical integrators, last examined by Wanner in the sixties. The linearly implicit predictor and the full HOP methods yield in the constant coefficient case Padé approximants of the matrix exponential. A- and L-stability is achieved for the diagonal and first two subdiagonal choices of the Padé parameter pair (q, p). Preliminary numerical results demonstrate that on stiff and highly oscillatory problems large steps can be realized with a single correction iteration and acceptable discretization error.

1 Introduction

Like other areas of numerical analysis, the design and inplementation of methods for the integration of ordinary differential equations has largely been based on the assumption that derivatives of the righthand side function are hard to come by and that their algorithmic use should therefore be avoided in general purpose solvers. While this is true in general, there has always been a group of theoreticians and practitioners who have developed and used Taylor series methods with the aim of achieving high accuracy and/or enclosing the exact solution in intervals [12, 18]. For the most part these methods have been employed for comparatively benign problems, where the

*Technical University Dresden, Mathematics Department, Institute of Scientific Computing, Mommsenstr. 13, 01062 Dresden, Germany

Jacobian of the righthand side has a moderate spread of eigenvalues and varies not too rapidly along the solution trajectory. Taylor series methods were apparently used to compute the reentry path of NASA space vehicles [13], but they were mostly ignored by the main stream of researchers and implementers as the interest focused onto stiff problems, for which explicit methods are unsuitable. While this is true for the classical Taylor series method, the Hermite-Obreshkov-Padé (HOP) methods examined in the second part of this paper are fully implicit. Also, the predictor advocated here represents by itself a linearly implicit A-stable integrator, similar to the Rosenbrock methods considered in [19].

Two internationally renowned researchers, namely Wanner and Gear, considered automatic differentiation as a tool for the numerical solution of stiff systems more than twenty years ago. Gerhard Wanner developed the theory of the class of integrators, which we call HOP (for Hermite-Obreshkov-Padé) methods here, and wrote a Fortran implementation STIFFI [21] based on a low order scheme from the family. Due to software limitations his method required a complete recoding of the righthand side by the user. His contributions on this topic were not published, as he apparently turned his interest to Implicit Runge Kutta schemes, which yield identical results to the HOP methods in the special case of linear constant coefficient problems. Bill Gear initially wrote a 'symbolic' differentiator to obtain the Jacobian of righthand sides for use in his Backward-Differentiation Codes for stiff problems. One of his students Bert Speelpenning [17] became a pioneer in automatic differentiation as he developed JAKE, probably the first, and definitely the most sophisticated implementation of the so-called reverse, or adjoint mode at the time. For more recent surveys on automatic differentiation see for example the proceedings [3] and the Siam News article [6].

There have been many implementations of the forward, or direct mode of automatic differentiation, some specifically for the purpose of integrating stiff systems. Even more often, users have relied on computer algebra packages to generate 'explicit formulas' for derivatives, an approach that is also sometimes employed for index reduction on differential algebraic systems. We cannot explain here how the chain rule based technique named automatic, or computational differentiation, departs from what is usually, but not very precisely, understood to be 'symbolic' differentiation. One important difference is that AD always provides a priori bounds on the operations count and memory requirement for evaluating certain derivatives. These bounds are typically the corresponding costs of the underlying function multiplied by a factor that depends on the derivative degree and the number of independent or dependent variables. It should be noted that AD does not incur any truncation errors and therefore has nothing in common with the practice of approximating derivatives by difference quotients.

In the context of differential equation solving there are really two differentiation tasks. One is the recursive generation of Taylor coefficient vectors for the solution path itself. The other task is the generation of Taylor coefficient matrices for the Jacobian of the righthand side along the solution trajectory. As we will see, these square matrices can also be interpreted as partial derivatives of the vector coefficients with respect to the current solution point. These higher order Jacobians are needed if one wants to apply Newton corrections to the algebraic system defining the HOP methods and other implicit integrators. After convergence of the corrector, we will use the same higher order matrix information for the predictor on the next step. Wanner [22] showed how the derivative vectors and their Jacobians can be computed recursively in the forward mode. In our pilot implementation [4] we are using the forward mode for calculating the derivative vectors and the reverse mode for calculating the derivative matrices. This involves d forward sweeps and one reverse sweep through the sequence of elementary operations and functions defining the right hand side. Since neither approach is in general optimal, we expect a significant reduction in the cost of derivative evaluations in the future.

A somewhat problematic subtask for all implicit integrators is the design of a predictor that requires only a small number of corrections on typical time-steps. On stiff and highly oscillatory problems the initial terms in the Taylor series diverge rather rapidly, so that any polynomial approximation to the solution is likely to be poor. Therefore, stiff integrators tend to employ a low order extrapolant through the last few points or simply use the current point as a prediction for the next approximate solution. As a result, for current implementations of the Implicit Runge-Kutta schemes, an average number of five simplified Newton corrections is apparently considered quite acceptable. We will show here how the Taylor coefficients and matrices, evaluated at the current point, can be used to form a rational predictor of high order for the next solution point. In our pilot implementation of the HOP methods one Newton correction is usually enough to reduce the residual of the nonlinear system orders of magnitude below the discretization error level.

This paper is organized as follows. In Section 2 we review the mathematical relationships and properties of the vector- and matrix Taylor coefficients and discuss discuss how these properties can be realized and exploited in automatic differentiation software. It is hoped that this material will be useful for various numerical purposes in dynamical systems. In Section 3 we describe the HOP methods with a rational predictor and provide estimates for the discretization error before and after a step is taken. In Section 4 we report some preliminary numerical results and draw some tentative conclusions. Proofs and more detailed derivations can be found in [4].

2 IVPs and their Taylor Series

Throughout this paper we consider an initial value problem

$$y'(t) \;=\; f(y(t),t) \quad , \quad y(0) = y_0 \in \mathbb{R}^n \tag{1}$$

with a righthand side

$$f \;:\; \mathcal{D} \times \mathbb{R} \subset \mathbb{R}^m \mapsto \mathbb{R}^n \quad \text{with} \quad m = n+1$$

that is smooth, i.e. infinitely often differentiable in an open domain $\mathcal{D} \times \mathbb{R}$.

In order to apply automatic differentiation principles and techniques we assume that f is defined by an evaluation procedure in a programming language like Fortran or C. In a mathematical sense, this means that f is a composite function made up of binary arithmetic operations and univariate scalar functions including exponentials, logarithms, sine and cosine. Whereas all these building blocks are smooth in the interiors of their domains, the situation changes drastically if one admits the absolute value, Euclidean norms, and possibly even the Heavyside function to the pool of elementary functions. As shown in [9], one can then still uniquely define and recursively compute arbitrarily many one-sided Taylor coefficients, which is of great potential use for the HOP methods and other one-step integrators. However, for simplicity we will assume in this paper that at all points of interest the elementary functions involved are smooth at their respective arguments.

The Vector Series

Excluding only very special cases, we may assume that all components of $y(t)$ are nonpolynomial functions whose values and derivatives can only vanish at isolated points in time. Hence, we have an infinite number of Taylor coefficients

$$y_j(t) \;\equiv\; \frac{\partial^j y(t)}{j!\,\partial t^j} \quad \text{for} \quad j = 0, 1, \ldots$$

so that at each time t and for any $d > 0$

$$y(t+h) \;=\; \sum_{j=0}^{d-1} y_j(t) h^j + \mathcal{O}(h^d)\;. \tag{2}$$

Given f and t, these vector coefficients can be computed at any y in the domain $\mathcal{D}$ by automatic differentiation (under the tacit assumption that y_0 has been defined such that $y(t) = y$). In other words, one can evaluate the vector functions

$$G_{t,j}(y) \;:\; \mathcal{D} \subset \mathbb{R}^n \mapsto \mathbb{R}^n \quad \text{for} \quad j = 0, 1, \ldots$$

defined by the relation

$$G_{t,j}(y) \;=\; y_j(t) \in \mathbb{R}^n \quad \text{with} \quad y = y(t) = y_0(t)$$

The efficiency of various ways of evaluating the $G_{t,j}$ and their Jacobians $G'_{t,j}$ will be discussed at the end of this section.

In some sense the difficulty of numerically approximating the solution $y(t)$ can be gauged by the convergence properties of its Taylor series (2). One may consider a problem as *stiff* if the terms in the series for a *reasonable* step-size h grow rapidly in size at first, and convergence happens only after hundreds or even thousands of terms have been taken into account. Of course, this definition is strongly dependent on what one means by a *reasonable* step-size h and how one defines the onset of convergence in an infinite series.

In most situations one will wish to restrict the step-size h a priori, such that for some given tolerance $\delta < 1$

$$\|y(t+h) - y(t)\| \;\leq\; \delta\|y(t)\| \quad .$$

In particular $y(t)$ may not have poles in the interval $[t, t+h]$. Having determined such a more or less natural step-size, one may then ask how many terms in the Taylor series (2) need to be included, such that for some constant $\rho < 1$ and all $j \geq d$

$$\|y_j\|\, h^j \quad \leq \quad \|y_{d-1}\|\, h^{d-1} \rho^{j-d+1} \quad .$$

In other words, we require that the terms in the series decline geometrically from the last term taken into account. This must eventually be the case when h is smaller than ρ times the convergence radius of the Taylor series at t. It also implies that the remaining error in the approximation (2) is bounded above by the last included term divided by $(1-\rho)$.

The informal definition of stiffness given above covers the classical dissipative and Hamiltonian cases with rapidly attenuating or highly oscillatory solutions, respectively. The simplest examples are

$$y(t) = \exp(-|\lambda| t) \quad \text{and} \quad y(t) = \sin(|\lambda|^2 t) \quad \text{with} \quad |\lambda| \gg 0\ .$$

In both cases the terms in the Taylor series alternate in sign and almost cancel each other out, until d is significantly larger than $|\lambda| h$. For large $|\lambda|$ this condition would either force very small steps or a high order d. Even though Taylor series methods have been used with more than a hundred terms, one usually considers only single digit values for d as reasonable. Then the only way to realize acceptable step sizes is to use methods that are not directly based on Taylor series expansion, as for example the HOP-methods described in Section 3.

So far, we have strived to describe the situation strictly in terms of the vector coefficients $y_j(t) = G_{t,j}(y)$. Especially stiffness is often discussed in terms of the Jacobian matrices, which we will consider now.

The Matrix Series

Because of our differentiability assumptions we have the extended Jacobian

$$J(y,t) \;\equiv\; [A(y,t), a(y,t)] \;=\; f'(y,t) \in \mathbb{R}^{n \times n+1} \quad ,$$

where $J(y,t)$ has been partitioned into

$$A(y,t) = \frac{\partial}{\partial y} f(y,t) \quad \text{and} \quad a(y,t) = \frac{\partial}{\partial t} f(y,t) \tag{3}$$

representing a square matrix and a column vector, respectively. The restrictions of these functions to the smooth solution trajectory $y(t)$, for fixed y_0, will be denoted by the same symbols without the y-argument, such that

$$J(t) \equiv [A(t), a(t)] = J(y(t), t) \equiv [A(y(t), t), a(y(t), t)]$$

are dependent on the time alone. The properties of the Jacobian trajectory $A(t)$ and the solution path $y(t)$ are closely related.

Like the components of $y(t)$, each entry of $J(t)$ is a transcendental function in t unless it represents a linearity in that the corresponding entry of $J(y, t)$ does not depend on the state vector y at all. When all entries of $J(t)$ are linear in that sense we have for arbitrary y

$$A(t) = A(y, t) \quad \text{and} \quad a(y, t) = A'(t)y + f'(0, t) \quad .$$

Then one can rewrite the initial value problem in the affine form

$$y'(t) = A(t)y(t) + f(0, t) \quad , \quad y(0) = y_0 \in \mathbb{R}^n \; .$$

We have infinitely many Taylor coefficients

$$J_j(t) \equiv \frac{\partial^j J(t)}{j! \partial t^j} = [A_j(t), a_j(t)]$$

which are all nontrivial, except in linear cases with a polynomial matrix function $A(t) = J(0, t)$. The sparsity of the $A_j(t)$ is increasing with respect to j in that

$$Zeros(A_j(t)) \subset Zeros(A_{j+1}(t)) \quad \text{and} \quad Zeros(a_j(t)) \subset Zeros(a_{j+1}(t)) \quad (4)$$

for all $j \geq 0$. Here, $Zeros(M(t))$ denotes the a set of index pairs for which the corresponding entry of a matrix path $M(t)$ vanishes identically. The sparsity pattern of the homogeneous Jacobian $A(t)$ is quite important for the efficiency of HOP methods.

In the nonlinear case the Jacobian $J(t)$ inherits stiffness (in the sense of poor Taylor series convergence over reasonable steps) from the components of $y(t)$ that it depends on. Conversely, stiffness of the solution is often discussed in terms of the eigenvalues $\lambda_i(t)$ of the Jacobian $J(t)$, under the tacit assumption that the corresponding eigen vectors do not rotate too rapidly. If the ratio between the largest and smallest modulus of any any eigenvalue is large the problem is considered stiff, and nonstiff otherwise. Rapid oscillations occur if some of the eigenvalues have large imaginary components. In general multi step codes based on the Backward Differentiation Formulas (BDF) are considered the most efficient methods for stiff problems, but the higher order formula are known to be unstable if some of the dominant eigenvalues are close to the imaginary line in the complex plane. Then, only lower order BDF, Rosenbrock [19], or higher order Implicit Runge Kutta (IRK) and HOP methods are applicable. The numerical experiments reported in Section 4 were conducted on a forced harmonic oscillator, for which the eigenvalues of the Jacobian can be easily selected in the complex plane.

Relations between the Series

The Taylor coefficients $y_j(t)$ and $J_j(t)$ of the solution and the Jacobian trajectories are, of course, intimately related. Some of these relations and identities can be used to evaluate and utilize the derivatives more efficiently. At first, let us adopt a strictly algebraic point of view and consider f simply as a map from $\mathbb{R}^m$ to $\mathbb{R}^n$ with $m = n + 1$. Given any smooth curve

$$y(t) \equiv \sum_{j=0}^{d-1} y_j(t-t_0)^j + \mathcal{O}(t-t_0)^d \in \mathcal{C}^d(\mathbb{R}, \mathbb{R}^n), \tag{5}$$

one obtains a smooth image

$$z(t) \equiv f(y(t),t) \;=\; \sum_{j=0}^{d-1} z_j(t-t_0)^j + \mathcal{O}(t-t_0)^d. \tag{6}$$

Since we have not yet imposed any differential relationship between $y(t)$ and $z(t)$ we may consider the coefficient vectors y_j as independent variables and obtain the resulting coefficient vectors z_j as functions

$$z_j \;=\; F_{t,j}(y_0, y_1, \ldots, y_j) \quad \text{with} \quad F_{t,j} : \mathbb{R}^{n\times(j+1)} \mapsto \mathbb{R}^n$$

for $j = 0, \ldots, d-1$. Each one of these vector functions $F_{t,j}$ is uniquely determined by the righthand side $f(y,t)$, the time t, and its order j. The partial derivatives of the $F_{t,j}$ with respect to the y_i for $0 \le i \le j$ are exactly the matrix coefficients discussed before, as shown in the following result from [4].

If f is d times continuously differentiable on some neighborhood of a point $(y_0, t_0) \in \mathbb{R}^{n+1}$, then for all $0 \le i \le j < d$,

$$\frac{\partial y_j}{\partial y_i} \;=\; A_{j-i} \;=\; A_{j-i}(y_0, y_1, \ldots, y_{j-i})$$

with A_i the i-th Taylor coefficient of the Jacobian of $f'(y(t),t)$ at $t = t_0$, i.e.

$$f'(y(t),t) \;=\; \sum_{i=0}^{d-1} A_i(t-t_0)^i + O(t-t_0)^d.$$

It should be noted in particular that the matrix A_k, and similarly the vector a_k, depend only on the y_j with $j \le k$. This fact may appear rather obvious, but it has the interesting consequence that z_j is a linear function of all y_i with $j \ge i > j/2$. Let us suppose that we have selected k Taylor coefficients y_j with $j = 0, \ldots, k-1$, and then evaluated the derivatives

$$z_j = F_{t,j}(y_0, \ldots, y_j) \quad \text{and} \quad A_j = A_j(y_0, \ldots, y_j) \quad \text{for} \quad j = 0, \ldots, k-1,$$

and the incomplete extra terms

$$\hat{z}_j \equiv F_{t,j}(y_0, \ldots, y_{k-1}, \underbrace{0, \ldots, 0}_{n\times(j-k+1)}) \quad \text{for} \quad j = k, \ldots, 2k-1\,. \tag{7}$$

Now consider the possibility that only afterwards the next k coefficients y_i with $i = k, \ldots, 2k-1$ are somehow determined. Then the corresponding z_j can be computed from the $\hat{z}_j$ as linear updates according to the formula

$$z_j \;=\; F_{t,j}(y_0, \ldots, y_j) \;=\; \hat{z}_j + \sum_{i=k}^{j} A_{j-i} y_i \quad \text{for} \quad j < 2k \tag{8}$$

This simple relation holds because the z_j are linear with respect to the higher coefficients y_i with $i \geq k$. Hence we have doubled the number of known coefficients z_j from k to $2k$, without referring back to the original righthand side function f. The formula (8) becomes particularly useful when the y_{j+1} are functions of the z_j by way of the differential relation

$$z(t) = y'(t) \quad \Rightarrow \quad y_{j+1} = z_j/(1+j) \quad . \tag{9}$$

Then one can define the coefficient functions $y_j = G_{t,j}(y) : \mathbb{R}^n \mapsto \mathbb{R}^n$ by the recurrence

$$G_{t,j+1}(y) \;\equiv\; \frac{1}{(1+j)} F_{t,j}(y, G_{t,1}(y), \ldots, G_{t,j}(y)) \quad \text{for} \quad j = 0, \ldots, d-1 \tag{10}$$

Differentiating (10) with respect to $y \equiv y_0$ we find that the total derivatives

$$B_j \;=\; B_j(y) \;=\; \frac{dy_{j+1}}{dy} \;=\; G'_{t,j}(y)$$

can be evaluated according to the chain rule by the recurrence

$$\begin{aligned} B_j \quad &= \quad \frac{1}{(1+j)} \left[\frac{\partial F_{t,j}(y)}{\partial y_0} + \sum_{i=1}^{j} \left(\frac{\partial y_j}{\partial y_i} \right) \left(\frac{dy_i}{dy_0} \right) \right] \\ &= \quad \frac{1}{(1+j)} \left[A_j + \sum_{i=1}^{j} A_{j-i} B_{i-1} \right] \quad . \end{aligned} \tag{11}$$

Here we have initially $B_{-1} = I$ and $B_0 = A_0$ with $A_{-1} = 0$ as a formal convention.

In contrast to the A_j the B_j have a decreasing sparsity pattern in that as a consequence of (11)

$$Zeros(B_{j-1}) \supset Zeros(B_j) = Zeros(A_0^j) \quad , \tag{12}$$

where the equality holds under the reasonable assumption that the Jacobian $A(t)$ has a nonvanishing diagonal. Then (12) follows by induction from (4) and (11). Unless A_0 is (permuted) block triangular (in which case (1) has a closed subsystem), the inverse $A(t)^{-1}$ is structurally dense and it follows by its Neumann expansion that the matrices A_0^j and B_j must have the same property for sufficiently large j. The fill-in in the derivative matrices B_j is of particular concern to us because the HOP method of order $2d$ leads to an algebraic system, whose Jacobian is a weighted sum of the B_j for $j < d$.

Another connection between the vector- and matrix-coefficients can be established by considering the **derived ODE**

$$z'(t) \quad = \quad A(t)\, z(t) + a(t) \quad , \quad z_0 = y_0' \quad . \tag{13}$$

Here $z(t) \equiv y'(t)$ is still the state derivative and $[A(t), a(t)]$ are as originally defined in (3). This initial value problem is obtained by simply differentiating the original system once (totally) with respect to time. Identifying coefficients and using (9) to express the z_j in terms of the y_{j+1} one obtains yet another recursion, namely we have

$$y_{j+1} \quad = \quad \frac{1}{(j+1)j} \left[a_{j-1} + \sum_{i=1}^{j} A_{j-i}\; i\, y_i \right] \quad \text{for} \quad j > 0 \quad . \tag{14}$$

In particular, having computed the y_j for $j = 0, \ldots, d$ and the $J_j = [A_j, a_j]$ for $j = 0, \ldots, d-1$ one obtains one extra vector coefficient, namely y_{d+1}, from (14) with $j = d$ almost for free. Note however, that the corresponding derivative matrix $B_d = dy_{d+1}/dy$ cannot be computed without evaluating the as yet unknown matrix A_d. In our implementation of the HOP method the extra term y_{d+1} is used for error estimation.

In stiff cases the conditioning of the $B_j \approx A_0^j/(1+j)!$ deterioates very rapidly with j. To combat this problem our pilot implementation currently employs rescaling by a factor τ so that the numerical scheme is effectively applied to the differential equation $y' = \tau f(y, \tau t)$.

Costs and Benefits of AD

With varying emphasis automatic differentiation has pursued two related goals. One is to optimize the way in which the chain rule is applied mathematically and the other is to implement the whole process in software that is as user-convenient as possible. Here, we can only give an indication of the mathematical aspects relating to the particular problem at hand.

The Taylor coefficients $y_j = G_{t,j}(t)$ are usually evaluated by automatic differentiation tools in the same way as by hand. Starting from $y_0 = y$ the higher coefficients $y_{j+1} = z_j/(1+j)$ are obtained one at a time. To this end truncated Taylor polynomials of degree j are propagated through the procedure for evaluating the righthand side, yielding z_j as the highest coefficient of the resulting polynomial. Each one of these sweeps costs $c + j^2/2$ or $c + j$ computational units, depending on whether the Taylor coefficients of intermediate quantities are recomputed each time or kept in storage from the previous sweep. A computational unit represents the number of arithmetic operations and the memory space to evaluate the righthand side once, usually by executing a subroutine in a high-level programming language. Consequently, the total computational effort for computing the first $d+1$ Taylor coefficients in d sweeps is $(c+d)d^2/6$ or $(c+d)d/2$ units. The constants c represent the overhead of transferring the Taylor coefficients from and to memory, an effort which may be a significant or even dominant part of the computation when d is relatively small. After discussing the evaluation of the corresponding $J_j = [A_j, a_j]$, we will show how the number of sweeps can be reduced from d to $\log_2(d+1)$ using the identity (8). One the basis of fast convolution algorithms one can calculate the first d Taylor coefficients vectors at a cost of order $\mathcal{O}(d\ln(d))$ units, thus avoiding the quadratic or cubic complexity growth incurred by simple polynomial arithmetic. However, the cross over point is far outside the range of the single digit d that are normally of interest in ODE solving.

Unfortunately, currently available automatic differentiation software is not very suitable for evaluating the Jacobians $A_j = \partial z_j/\partial y_0$ and the vectors $a_j = \partial z_j/\partial t$ efficiently. This combined task constitutes a problem of nested differentiation, where the scaled time derivatives $y_{j+1} = x_j/(1+j)$ are differentiated once more with respect to the $m = n + 1$ vector (y, t). In most ODE problems the time parameter enters in a special way, not like one of the spatial components. Moreover, there are often other structural properties like linearity, sparsity and partial separability that could and should be exploited automatically. There currently exists no software that can automatically detect and exploit such desirable properties to the extent they are present, but that will hopefully change in the future. Until then the HOP methods will only in special circumstances be run-time competitive with derivative-free state of the art integrators.

In our package ADOL–C [8] the evaluation of the J_j for $j < d$ by the reverse mode of automatic differentiation costs roughly n times as many units as the evaluation of the underlying y_j for $j \leq d$ with storage of intermediates. Essentially, the same complexity applies for the dense forward mode, which was already advocated by Wanner [22] in the context of numerical ODE solving. ADOL–C yields the sparsity pattern for the partial Jacobians A_j, which is then used by the routine that accumulates them to the total Jacobians B_j. However, the sparsity structure is not yet exploited in the differentiation process itself, which could yield substantial savings [1].

It is hoped that the cost penalty n in going from the y_j to the J_j can be drastically reduced by a combination of the forward and reverse mode [7] that also takes account of the special role played by the time. Naturally, better methods for computing Jacobians will benefit many methods for the numerical analysis of nonlinear problems. An achievable improvement is the detection and suitable representation of jumps, kinks and higher order derivative discontinuities, which occur in many practical righthand sides.

One scheme that has been validated experimentally is the doubling method based on (8). Starting with y_0, we may evaluate $y_1 = z_0$ and the incomplete coefficient $\hat{z}_1$ as defined in (7) for $k = 1$, during a first forward sweep. Subsequently, we can perform a reverse sweep to obtain the Jacobian A_0, which substituted into (8) yields the complete value for $2y_2 = z_1 = \hat{z}_1 + A_0 y_0$. Hence, we have a total of 3 Taylor coefficients after one combined forward/reverse sweep. Next time we propagate $y_3 = z_2/3$ and $\hat{z}_3, \hat{z}_4, \hat{z}_5$ for

$k = 3$. The subsequent reverse sweep yields A_1 and A_2, which allow us to complete the values of z_3, z_4 and z_5, so that we now have all y_j with $j < 7$ correct. Continuing this procedure we find that the number of known y_j grows from k to $2k + 1$ on every combined forward and reverse sweep. Starting from $k = 1$ we obtain the rapidly growing sequence $3, 7, 15, 31, \ldots$, so that just a small number of i sweeps are enough to obtain $d = 2^i - 1$ vectors y_j, and about half as many Jacobians J_j. One might then use an extra combined sweep to obtain the remaining Jacobians J_j for $d/2 < j < d$ or decide to make do without them. The latter strategy might work well for the HOP method, where the Jacobians are *only* used for prediction and correction, without actually affecting the solution points. Even when structure is exploited optimally, the cost of evaluating the A_j up to a particular order d is certain to dominate that of evaluating the corresponding y_j. Therefore using only half as many should roughly quarter the computational effort per step.

3 The HOP methods with rational Predictor

While the classical Taylor series method implemented with AD [2] can be quite efficient on nonstiff problems, we have to find an implicit variant for the more interesting stiff case. Most one-step methods for the numerical solution of systems of ordinary differential equations can be interpreted as quadratures for the identity

$$y(t_+) = y(t_0) + \int_{t_0}^{t_+} y'(t)\, dt \quad . \tag{15}$$

In the given time interval $[t_0, t_+]$ one replaces the function $y'(t) = f(y(t), t)$ by some approximation, which depends parametrically on values of y' and its derivatives at nodes in $[t_0, t_+]$. We will consider Hermite interpolants, i.e. polynomials that interpolate $y'(t)$ and a certain number of its derivatives at the two endpoints t_0 and t_+. The resulting quadrature formulas discovered by Obreshkov [14, 15] contain also the values of $y(t)$ itself at the endpoints. Therefore, it is sufficient to annotate quantities by the subscripts 0 and + to indicate whether they belong to the left or right end of the interval.

Derivation of the HOP methods

Rather than pursuing the Hermite interpolation directly, one can derive the HOP methods much faster in the following way due to Wanner [20]. Consider for any pair (q, p) of natural numbers the polynomial

$$P(s) \;\equiv\; s^p(1-s)^q/(p+q)! \quad .$$

The salient property of $P(t)$ is that its first $(p-1)$ derivatives vanish at $s = 0$ and its first $(q-1)$ derivatives vanish at $s = 1$. The remaining derivatives can be seen to have the values

$$P^{(p+q-j)}(0) = \frac{(-1)^j}{j!} c_j^{q,p} \quad \text{and} \quad P^{(p+q-j)}(1) = \frac{1}{j!} c_j^{p,q} \quad .$$

where

$$c_j^{q,p} \equiv \frac{q\,(q-1)\cdots(q-j+1)}{(q+p)\cdots(q+p-j+1)} \leq 1$$

and $c_j^{p,q}$ correspondingly with p and q interchanged. Now it can be verified, through repeated integration by parts, that for any smooth function $g(s)$ on $[0,1]$

$$\int_0^1 P(1-s) g^{(q+p+1)}(s) ds = \sum_{j=0}^{q} (-1)^j c_j^{q,p} g_j(1) - \sum_{j=0}^{p} c_j^{p,q} g_j(0) \quad ,$$

where $g_j(s) = g^{(j)}(s)/j!$ denotes the j-th Taylor coefficient as before. For the particular choice $g(s) \equiv y(t_0 + sh)$ with $h = t_+ - t_0$ we find that $g^{(q+p+1)}(s) = \mathcal{O}(h^{q+p+1})$ and hence the integral on the left is also of order $(p+q+1)$ in h. Thus we obtain with $g_j(0) = h^j y_j(t_0)$ and $g_j(1) = h^j y_j(t_+)$ the relation

$$\sum_{j=0}^{q} (-h)^j c_j^{q,p} y_j(t_+) = \sum_{j=0}^{p} h^j c_j^{p,q} y_j(t_0) + \mathcal{O}(h^{q+p+1}) \quad . \tag{16}$$

Now consider $y(t_0) = y_0$ as initial condition over the current step. Then the point

$$y_{1/2}^{q,p} \equiv \sum_{j=0}^{p} h^j c_j^{p,q} y_j(t_0) = \sum_{j=0}^{p} h^j c_j^{p,q} G_{0,j}(y_0)$$

is well defined and may be interpreted as a midpoint between y_0 and the next point $y(t_+)$ on the analytical trajectory. The latter point now may be approximated by the hopefully unique solution y_+ of the vector equation

$$H^{q,p}(y_+) \equiv \sum_{j=0}^{q} (-h)^j c_j^{q,p} G_{t_+,j}(y_+) - y_{1/2}^{q,p} = 0 \quad . \tag{17}$$

This system of nonlinear algebraic system defines the (q,p) HOP method. Clearly, the choices $(0,1), (1,0)$ and $(1,1)$ yield the explicit or implicit Euler method and the trapezoidal scheme, which is identical to the midpoint rule if the points $y_{1/2}^{1,1}$ are viewed as main iterates.

Since y_0 is a solution for $h = t_+ - t_0 = 0$, there exists a unique path of solutions, as long as the Jacobian

$$R^{q,p} \equiv \frac{\partial H^{q,p}(y)}{\partial y} = \sum_{j=0}^{q} (-h)^j c_j^{q,p} G'_{+,j}(y)$$

is nonsingular. Since with $B_j = G'_{+,j}$ as defined in (11) at $t = t+$

$$R^{q,p} \;=\; I + \sum_{j=1}^{q} (-h)^j c_j^{q,p} B_{j-1} \quad , \tag{18}$$

this regularity assumption must hold for small h and there are solutions $y_+ \approx y(t_0 + h)$ on some interval $[0, \bar{h}]$. Moreover, since the neglected integral term was of order $(p + q + 1)$ in h, we obtain the local discretization error

$$y_+ - y(t_+) \;=\; \mathcal{O}(h^{p+q+1}) \quad .$$

Naturally, the error constant on the right is dependent on the size of the Taylor coefficient $y_{p+q+1}(t)$ within the interval $t_0 \le t \le t_+ = t_0 + h$.

Extending the the terminology of [10] we refer to the numerical integrators defined by (17) as Hermite-Obreshkov-Padé methods. Padé enters into the picture, because for homogeneous ODEs with constant coefficient matrix A_0, we have $G_{t,j}(y) = A_0^j y/j!$ and $B_j = A_0^{j+1}/(1+j)!$, so that $R^{q,p}$ is a matrix polynomial in A_0 and y_+ can be expressed as

$$y_+ \;=\; [R^{q,p}]^{-1} \left[\sum_{j=0}^{p} h^j c_j^{p,q} A_0^j / j! \right] y_0 \quad .$$

This is exactly the (q, p) Padé approximant to the analytical solution trajectory $y(t_0 + h) = \exp(hA_0)y_0$. It is well known [5] that integrators with this transfer function in the linear case are A-stable if and only if $p \le q \le p + 2$ and L-stable if $p < q \le p + 2$. While these properties are highly desirable for the numerical solution of stiff problems, one would also wish to obtain stability results on wider classes of dissipative functions. Unfortunately, these have not yet been found for the HOP methods.

Since we have in (18) an explicit representation of the Jacobian $R^{q,p}$ in terms of the B_j, whose evaluation was discussed at length in Section 2, it is clear that we wish to solve (17) by Newton's method. As we have noted, the sparsity of the B_j is decreasing with j. Consequently, $R^{q,p}$ has the sparsity of the last term B_{q-1}, which can be exploited during the computation of the Newton correction $\Delta\tilde{y}$ by solving the linear system

$$R^{q,p} \tilde{\Delta} y = -H^{q,p}(\tilde{y}_+) \quad ,$$

where $\tilde{y}_+$ represents the current approximation to y_+. All the usual tricks of nonlinear equations solving for stabilizing the iteration and reducing the number of Jacobian evaluations or factorizations can be applied. However, the best remedy for uncertain and slow convergences is usually a good initial approximation $\tilde{y}_+$.

Prediction and Error Estimation

The HOP methods promise small discretization errors and good stability, even for comparatively large steps. In order to realize this potential advantage, we need a good predictor because even stabilized Newton's methods will otherwise be unable to locate a solution of the the nonlinear algebraic system (17). Since polynomial predictors are well known to be unsuitable for stiff systems, we attempt to derive a predictor that is a rational function of the available data. The prediction problem does not arise if the ODE (1) and consequently the algebraic system (17) are linear, in which case Newton's method converges in a single step form any initial guess. Since the derived ODE (13) is always affine we can use it to generate a good predictor as follows.

After convergence to the current point y_0 during the previous time-step, we know not only the Taylor coefficients $y_j = G_{0,j}(y_0)$ for $j = 1, \ldots q$, but also the approximations

$$\tilde{A}(t) \equiv \sum_{j=0}^{q-1} A_j (t-t_0)^j \quad \approx \quad A(t) = \frac{\partial}{\partial y} f(y(t), t) \quad ,$$

and

$$\tilde{a}(t) \equiv \sum_{j=0}^{q-1} a_j (t-t_0)^j \quad \approx \quad a(t) = \frac{\partial}{\partial y} f(y(t), t) \quad .$$

Substituting these approximations into (13) we obtain the approximating derived ODE

$$\tilde{z}'(t) = \tilde{A}(t)\tilde{z}(t) + \tilde{a}(t) \quad , \quad \tilde{z}(t_0) = y_0' \tag{19}$$

Since the coefficient functions have been altered by a perturbation of order h^q the solution $\tilde{z}(t_+)$ at $t_+ = t_0 + h$ differs from $z(t_+)$ by a discrepancy of order h^{q+1}. The numerical integration of (19), by the (q,p) HOP method with the local order $q+p+1 \geq q+1$ yields an approximate solution $\tilde{z}_+ \approx \tilde{z}(t_+)$ with

$$\tilde{z}_+ = \tilde{z}(t_+) + \mathcal{O}(h^{q+p+1}) = z(t_+) + \mathcal{O}(h^{q+1}) \, .$$

The computation of $\tilde{z}_+$ involves the calculation of

$$\tilde{A}_j \equiv \left. \frac{\partial^j \tilde{A}(t)}{\partial t^j} \right|_{t=t_+} \quad \text{and} \quad \tilde{a}_j \equiv \left. \frac{\partial^j \tilde{a}(t)}{\partial t^j} \right|_{t=t_+}$$

by a so-called Taylor shift and the forming and factoring of the resulting $\tilde{B}_j$ and $\tilde{R}^{q,p}$ according to (11) and (18).

We are left with the task of deriving an equally good estimate $\tilde{y}_+ \approx y(t_+)$ from the approximate slope $\tilde{y}_+' = \tilde{z}_+$ just computed. To this end, we first apply the recurrence (14) for the derived ODE (13) with $J_j = [A_j, a_j]$ replaced by

$$\tilde{J}_j \equiv [\tilde{A}_j, \tilde{a}_j] = J_j + \mathcal{O}(h^{q-j}) \quad \text{for} \quad j < q \quad .$$

It then follows by induction, starting from $\tilde{z}_0 \equiv \tilde{z}_+$, that the resulting approximate Taylor coefficients $\tilde{z}_j$ differ from the exact coefficients $z_j(t_+) = (1+j)y_{j+1}(t_+)$ by an error of size $\mathcal{O}(h^{q+1-j})$.

Setting $\tilde{y}_{j+1} \equiv \tilde{z}_j/(1+j)$, we have now obtained $\mathcal{O}(h^{q+2})$ approximations to all terms occurring in the defining equation (17), except for the one involving $y_+ \approx y(t_+)$. Bringing this unknown term over to the lefthand side we find that the predictor

$$\tilde{y}_+ \;\equiv\; \sum_{j=0}^{p} h^j c_j^{p,q} y_j(t_0) \;-\; \sum_{j=1}^{q} (-h)^j c_j^{q,p} \tilde{y}_j$$

satisfies

$$\tilde{y}_+ \;=\; y_+ + \mathcal{O}(h^{q+2}) \;=\; y(t_+) + \mathcal{O}(h^{q+1+\min(1,p)}) \quad .$$

Except for the implicit Euler method and other HOP schemes with $p = 0$, the predictor $\tilde{y}_+$ is therefore an $\mathcal{O}(h^{q+2})$ approximation to both the numerical solution y_+ and the local analytical solution $y(t_+)$.

In any case, one Newton correction will reduce the residual to $\mathcal{O}(h^{2q+4})$, which is at least three orders below the local truncation error level for the A-stable schemes where $p \leq q$. However, the mere order of the predictor is not the main consideration, since it could also be achieved with polynomial expansion or extrapolation. The key advantage is that the predictor is rational in the function and derivative data, due to the inversion of the approximate Jacobian $\tilde{R}^{q,p} \approx R^{q,p}$. It has been shown in [4], that in the linear case with constant coefficients the predictor is exact in that $\tilde{y}_+ = y_+$ so that no correction is necessary at all. This means in particular, that the predictor by itself defines a linearly implicit scheme that is L-stable or least A-stable depending on whether $p < q$ or $p = q$ with $p \geq q-2$ understood. In our numerical experiments we found that even on (possibly time-varying) linear problems, correction steps were rarely taken, whereas on nonlinear problems a single correction iteration was typical. This desirable effect may be partly due to the short step sizes resulting from following, rather conservative, procedure for estimating discretization errors before and after a step has been taken.

The approximate information available to the predictor yields according to (14) also an approximation $\tilde{y}_{q+1} = y_{q+1} + \mathcal{O}(h)$, which we have not used so far. Thus we can compute the predicted residual

$$\tilde{r}^{p+1,q+1} \;\equiv\; \sum_{j=0}^{q+1} (-h)^j c_j^{q+1,p+1} \tilde{y}_j \,-\, y_{1/2}^{q+1,p+1} \tag{20}$$

of the $(p+1, q+1)$ scheme at the predicted point $\tilde{y}_+$. Now we need an approximation $\tilde{R}^{q+1,p+1}$ to the Jacobian $R^{q+1,p+1}$ of this higher order scheme, in order to compute the Newton-step $[\tilde{R}^{q+1,p+1}]^{-1}\tilde{r}^{p+1,q+1}$ as an estimate for the discrepancy between $\tilde{y}_+$ and the actual solution $y(t_+)$. For simplicity one may want to use the matrix $\tilde{R}^{q,p}$, which has already been formed and factorized. On very stiff problems that may not be a good idea since $R^{q+1,p+1}$ involves the matrix B_q which has the leading term A_0^q whereas B_{q-1}

is only of order A_0^{q-1}. Therefore, it may make sense to compute $\tilde{B}_q$ with $\tilde{A}_q$ set to zero since the evaluation of A_q just for error estimation would be too expensive. The resulting $\tilde{R}^{q+1,p+1}$ will then be exact for constant coefficient problems so that the a priori error estimate discussed so far will coincide with the following a posteriori error estimate.

After the corrector iteration has reached a point $\tilde{y}_+$ with an acceptably small residual $H^{q,p}(\tilde{y}_+)$ of the nonlinear system (17) one usually wishes to obtain some a posteriori estimate of the discretization error. This can be computed just like the a priori estimate by evaluating $r^{q+1,p+1}$ as in (20) but with the $\tilde{y}_j$ replaced by the $G_{+,j}(\tilde{y}_+)$. By preconditioning with a suitable approximation to $R^{q+1,p+1}$ one obtains an approximate Newton-step whose size may serve as an estimate of the discretization error. Since it is known to be of order h^{p+q+1}, the step size h can be reduced by the $(p+q+1)$-st root of a desired error reduction factor should the estimate exceed the user specified tolerance. On the other hand, if the estimate is significantly below the tolerance, the trial value of h for the next time step can be chosen as an appropriate multiple of the current value. While this procedure is standard in the field of numerical ODEs, we also can test and possibly adjust the new step size on the basis of the priori error estimate, before any extra evaluations have taken place. In our experience this effort pays off in that very few steps need to be rejected, without the step-sizes being excessively small.

4 Numerical Experiments and Conclusion

The purpose of our experimentation is to demonstrate that the higher order information obtained in form of the vector and matrix coefficients can be put to good use in the subdiagonal HOP schemes. It was expected that long steps with few corrections could be taken on stiff problems, including the ones with highly oscillatory solutions, where higher order BDF codes can be expected to run into difficulties. These notions were confirmed in a sizable number of numerical test runs, some which are reported in [4]. While the number of time-steps and corrections is generally quite low, the effort per step is of course significantly larger than for derivative-free methods, and in most cases the run-times of our code are not yet competitive.

To highlight the properties of the HOP methods we have studied their behavior on the damped and forced oscillator problem

$$y'' + \beta(y')y' + c^2 y \;=\; \alpha \sin(\omega t)$$

for fixed Hook's constant c^2 with $c > 0$, forcing amplitude and frequency (α, ω), and nonnegative friction coefficient $\beta(\zeta)$. In the linear case $\beta(\zeta) = 2b$ with constant $b \geq 0$ the Jacobian has the eigenvalues

$$\lambda \;=\; -b + \sqrt{b^2 - c^2}\;.$$

In case of subcritical damping where $b < c$ the eigenvalues are complex conjugate, have modulus $|\lambda| = c$, and they always form the angle

$$\phi = \operatorname{acos}(\min(b/c, 1)) \tag{21}$$

with the negative real axis. Provided $b > 0$ the solutions converge from all initial conditions eventually to the limit-cycle $y(t) = \tilde{\alpha}\sin(\omega t - \delta)$ with the amplitude $\tilde{\alpha} = \alpha/\sqrt{(c^2-\omega^2)^2+(2b\omega)^2}$ and the phase delay $\delta = \tan^{-1}(2b\omega/(c^2-\omega^2))$. Presumably, one wishes to track these oscillations with some degree of accuracy so that one might consider a step size h as reasonable if $h\omega$ is a sizable fraction of π.

In case of subcritical damping the particular solution will be superimposed by sinusoidal oscillations with the eigenfrequency $\sqrt{(c^2-b^2)}$. The amplitude of these homogeneous solutions depends in the beginning on the initial condition and tends asymptotically to zero, except in the frictionless case $b = 0$. Nevertheless, discretization and round-off errors or other perturbations are likely to reinject small amplitudes of these eigenoscillations. Since their Taylor series converges quite slowly if $h|\lambda| > 1$, we will consider the problem as stiff whenever

$$\omega \ll |\lambda| = \max(c, \sqrt{|2\,b^2 - c^2|})\,.$$

This condition applies like (21), even in the supercritical and critical cases $b > c$ and $b = c$, respectively. The numerical results for the linear case are listed in Table 1. Here we compare the BDF code form the NAG library with our HOP implementation of order $(4, 3)$ and $(9, 8)$. All runs were conducted with a tolerance of 10^{-6} over the interval $0 \le t \le 5\pi$. The initial point was computed reasonably close to the periodic solution by integrating first over ten periods of the forcing term.

The number pairs in the table entries represent the function and Jacobian evaluations. For BDF this means simply evaluating the right hand side y_1 or its Jacobian J_0, respectively. In case of the HOP methods this requires the evaluation of all y_j with $j \le q{+}1$ or all J_j with $j < q$, respectively. Hence the comparison is certainly not fair in the run time sense and the HOP methods should be penalized at the very least with a factor of q. The results in Table 1 confirm the theoretical expectation that the HOP methods are less effected than the BDF code by eigenvalues close to the imaginary axis, which are reported above the diagonal of the table. It is also not very surprising that the larger the friction term b is the more ground higher order $(9, 8)$ scheme loses to the lower order $(4, 3)$ scheme. Looking at the undamped cases in the first row, we see that the $(9, 8)$ and the $(4, 3)$ scheme take per eigenperiod almost exactly one and six steps, respectively, whereas the BDF methods takes many more. Nevertheless, the steps are automatically selected so that $h|\lambda|$ is of order 1 when $b = 0$. Hopefully, this should not apply if the original conditions are chosen such that the amplitude of the fast oscillations is below the tolerance. When $b > 0$ the steps get gradually

	$b\backslash c$	0	10	10^2	10^3	10^4
BDF		707/40	1627/1411	16894/866	278500/18939	2894415/119793
(3, 4)	0	167/86	231/225	1502/1502	15002/15002	150002/150002
(8, 9)		52/37	49/49	252/252	2503/2503	25002/25002
BDF		623/37	424/34	4970/665	–/–	1243557/107029
(3, 4)	10	185/82	211/211	248/248	657/656	4931/4929
(8, 9)		48/48	51/45	52/52	176/176	1348/1345
BDF		588/38	593/38	2509/416	–/–	–/–
3, 4)	10^2	184/184	185/185	161/161	152/152	620/618
(8, 9)		44/44	44/44	44/43	61/61	329/328
BDF		678/46	810/72	752/60	–/–	–/–
(3, 4)	10^3	128/128	136/136	123/123	88/88	99/98
(8, 9)		69/69	62/62	62/62	49/49	233/233

Table 1: Linearly Damped Forced Oscillator

larger as the fast oscillations are damped out and the total number of steps grows clearly sublinear as a function of c.

It is also worth noting that for our HOP implementations, the number of Jacobian and function evaluations is practically the same, which means that the predicted point was accepted without any correction on most iterations. To explain this effect we note that with linear damping the problem has a constant Jacobian $A(t) = A_0$. However, because the forcing term is non–polynomial the predictor is not quite exact. More specifically, the discrepancy between the predicted point and the numerical solution is proportional to the error between $\sin(\omega t)$ and the first q terms of it's Taylor expansion over a step of size h. Since, the steps are selected such that $h\omega \ll 1$ this discrepancy is usually quite small. In contrast we find on the nonlinear problem reported in Table 2 that a single correction iteration is required on most time-steps.

The choice $\beta(\zeta) = 2b\zeta^2$ as a nonlinear damping term makes the problem much harder. Now the Jacobian $J(t)$ paths inherits *stiffness* from the solution trajectory, in the sense that its Taylor coefficients J_j become very large and ill-conditioned. Despite the scaling mentioned in the previous section, exponent over- or under-flow leads to some numerical difficulties, which have not yet been satisfactorily resolved. This may be part of the reason why the higher order $(9, 8)$ scheme takes more steps than the $(4, 3)$ scheme when p and q are equal to 10,000. Again, the BDF code requires consistently many more function and Jacobian evaluations , and in some case the HOP methods were actually faster in terms of total run-time. In view of the fact that the test problem is very small and the $A_j(t)$ have only one non-vanishing component it is not appropriate to draw any conclusions regarding the actual efficiency of the HOP methods on practical problems. In any case, the

	$b\backslash c$	0	10	10^2	10^3	10^4
BDF		1448/102	1943/120	26130/1771	270454/18665	2774188/2157651
$(4,3)$	10	262/260	349/194	2605/1934	18282/17439	148592/148559
$(9,8)$		117/64	160/86	1206/821	9712/971	84751/49183
BDF		2535/189	1756/108	24427/1715	250063/191529	2533661/176178
$4,3)$	10^2	405/252	279/263	2346/1642	16134/14980	128149/127661
$(9,8)$		167/110	129/79	1170/609	9964/5470	64750/52627
BDF		3258/2761	256/199	22495/1639	226464/169932	2298893/166129
$(4,3)$	10^3	500/423	334/307	2200/1293	13136/122681	77632/76434
$(9,8)$		215/158	167/125	1081/605	8053/5832	57044/51279
BDF		3681/315	3718/310	18913/14142	190018/12988	2090526/163104
$(4,3)$	10^4	531/418	530/400	1649/1294	11111/10044	35894/34971
$(9,8)$		271/212	295/219	984/637	6793/5276	58216/46703

Table 2: Nonlinearly Damped Forced Oscillator

natural competitors are the IRK methods, especially on problems with rapid oscillations that cannot simply be damped out artificially.

Summary and Conclusions

For all righthand side functions defined by evaluation procedures, Taylor coefficients of the solution and Jacobian trajectory can be obtained with high accuracy and at a reasonable cost. Further improvements in automatic differentiation methodology promise a significant reduction in computational cost and the systematic detection and treatment of derivative discontinuities, where one-sided Taylor-coefficients can still be defined and computed. This methodology is applicable to boundary value problems, differential algebraic equations and many other nonlinear problems in scientific computing.

In the second part of the paper the higher order derivative data were utilized in the predictor and corrector of Hermite-Obreshkov-Padé methods. Comparatively large steps with acceptable discretization error and at most one correction iteration could be realized on stiff problems with rapidly attenuating and/or highly oscillatory solutions. It is not yet clear whether the well-known A- and L-stability results can be extended to nonlinear test functions. Even though it is known that the higher order HOP methods are not symplectic [11], it is expected that the time-reversible (q,q) schemes perform reasonably well on Hamiltonian systems.

Acknowledgements

Much of the material in this paper is based on joint research with George Corliss, Petra Henneberger, Gabriela Kirlinger, Florian Potra, and H. J.

Stetter. The results of this collaboration will be published in the manuscript [4], which benefited greatly from comments by Ian Gladwell. The numerical results were obtained by Petra Hennerberger with a code that has been developed over the years with the help of George Corliss and several students.

References

[1] Brett Averick, Jorge Moré, Christian Bischof, Alan Carle, and Andreas Griewank. Computing large sparse Jacobian matrices using automatic differentiation. *to appear in SIAM Journal on Scientific Computing*, 1993.

[2] Y. F. Chang and G. Corliss, Solving ordinary differential equations using Taylor series, *ACM Trans. Math. Software,* 8(1982), 114–144.

[3] Andreas Griewank and George Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Applications.* SIAM, Philadelphia, Penn., 1991.

[4] G. F. Corliss, A. Griewank, P. Henneberger, G. Kirlinger, F. A. Potra, H. J. Stetter, High-Order Stiff ODE Solvers via Automatic Differentiation and Rational Prediction, Manuscript, submitted for publication, 1995.

[5] B. L. Ehle, A-stable methods and Padé approximations to the exponential, *SIAM J. Math. Anal.* 4(1973), 671–680. Manuscript, submitted for publication, 1995.

[6] Andreas Griewank. The chain rule revisited in scientific computing, I-II. *SIAM News*, May/July 1991.

[7] Andreas Griewank and Shawn Reese, On the calculation of Jacobian matrices by the Markowitz rule. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 126–135. SIAM, Philadelphia, Penn., 1991.

[8] Andreas Griewank, David Juedes, and Jean Utke, ADOL-C, a package for the automatic differentiation of algorithms written in C/C++, *ACM Transactions on Mathematical Software*, to appear, 1995. First version submitted in 1991.

[9] Andreas Griewank. Automatic Directional Differentiation of Nonsmooth Composite Functions , to appear in Proceedings of Seventh French-German Conference on Optimization, Lecture Notes in Economics and Mathematical Systems, Springer Verlag.

[10] E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II,* Springer-Verlag, Berlin, 1991.

[11] E.Hairer, A.Murua, J.M.Sanz-Serna. *The non-existence of symplectic multi-derivative Runge-Kutta methods*, BIT 34 (1994), 80-87.

[12] R. Lohner, Einschließung der Lösung gewöhnlicher Anfangs-und Randwertaufgaben und Anwendungen, Dissertation, Karlsruhe 1988.

[13] Fredrick Munger. *Applications of Definor Algebra to Ordinary Differential Equations*, After Math Press, Instructor's Edition, 1990.

[14] N. Obreshkov, Neue Quadraturformeln, Abh. Preuss. Akad. Wiss. Math. Nat. Kl., **4**,(1940).

[15] N. Obreshkov, Sur le quadrature mecaniques (Bulgarian, French summary), Spisanie Bulgar. Akad. Nauk, **65**, 191-289(1942).

[16] H. Padé, Sur la représentation approchée d'une fonction par des fractions rationelles, Thesis, Ann. de l'Éc. Nor. (3), 9(1892).

[17] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*, Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana,(1980).

[18] H. J. Stetter, Validated solution of initial value problems for ODE, in Computer Arithmetic and Self Validating Numerical Methods, *Proceedings SCAN Basel 1989*, 171–187 (1990).

[19] Karl Strehmel und Rüdiger Weiner. *Linear–implizite Runge–Kutta–Mathoden und ihre Anwendung*, Teubner–Texte zur Mathematik, Stuttgart · Leipzig, 1992.

[20] G. Wanner. On the integration of stiff differential equations. Technical Report, October 1976, Université de Genéve Section de Mathematique, 1211 GENEVE 24th, Suisse.

[21] G. Wanner. STIFFI, A Program for Ordinary Differential Equations. Technical Report, October 1976, Université de Genéve Section de Mathematique, 1211 GENEVE 24th, Suisse.

[22] G. Wanner. *Integration gewöhnlicher Differentialgleichungen*, Hochschultaschenbücher–Verlag, Bibliographisches Institut, Mannheim/Zürich, 1969.