Robust and Reliable Defect Control for Runge-Kutta Methods

W. H. ENRIGHT
University of Toronto
and
WAYNE B. HAYES
University of California, Irvine

The quest for reliable integration of *initial value problems* (IVPs) for *ordinary differential equations* (ODEs) is a long-standing problem in numerical analysis. At one end of the reliability spectrum are fixed stepsize methods implemented using standard floating point, where the onus lies entirely with the user to ensure the stepsize chosen is adequate for the desired accuracy. At the other end of the reliability spectrum are rigorous interval-based methods, that can provide provably correct bounds on the error of a numerical solution. This rigour comes at a price, however: interval methods are generally two to three orders of magnitude more expensive than fixed stepsize floating-point methods. Along the spectrum between these two extremes lie various methods of different expense that estimate and control some measure of the local errors and adjust the stepsize accordingly.

In this article, we continue previous investigations into a class of interpolants for use in Runge-Kutta methods that have a defect function whose qualitative behavior is asymptotically independent of the problem being integrated. In particular the point, in a step, where the maximum defect occurs as $h \to 0$ is known a priori. This property allows the defect to be monitored and controlled in an efficient and robust manner even for modestly large stepsizes. Our interpolants also have a defect with the highest possible order given the constraints imposed by the order of the underlying discrete formula. We demonstrate the approach on three Runge-Kutta methods of orders 5, 6, and 8, and provide Fortran and preliminary *Matlab* interfaces to these three new integrators. We also consider how sensitive such methods are to roundoff errors. Numerical results for four problems on a range of accuracy requests are presented.

Categories and Subject Descriptors: G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations; G.1.1 [**Numerical Analysis**]: Interpolation; G.1.2 [**Numerical Analysis**]: Approximation; G.4 [**Mathematical Software**]; J.2 [**Physical Sciences and Engineering**]

General Terms: Algorithms, Reliability

This research was supported in part by the Natural Sciences and Engineering Council of Canada, and by the Information Technology Research Centre of Ontario.

Authors' addresses: W. H. Enright, Department of Computer Science, University of Toronto, Toronto, M5S 3G4, Ont., Canada; email: enright@cs.toronto.edu; W. B. Hayes, Department of Computer Science, University of California, Irvine, Irvine, CA 92697-3435; email: wayne@ics.uci.edu.

Additional Key Words and Phrases: Initial value problems, ordinary differential equations, Runge-Kutta methods, interpolation

ACM Reference Format:

Enright, W. R. and Hayes, W. B. 2007. Robust and reliable defect control for Runge-Kutta methods. ACM Trans. Math. Softw. 33, 1, Article 1 (March 2007), 19 pages DOI = 10.1145/1206040.1206041 http://doi.acm.org/10.1145/1206040.1206041

1. INTRODUCTION AND MOTIVATION

Consider the *initial value problem* (IVP) for an *ordinary differential equation* (ODE)

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \tag{1}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0, \tag{2}$$

where $\mathbf{y}$ is an n-dimensional vector and $\mathbf{f}$ an n-dimensional vector-valued function. Standard forward error analysis (see for example Dahlquist and Björck [1974]) tells us that it is only possible, for a very restricted class of problems (strongly dissipative ODEs), for the numerical solution to remain uniformly close to the exact solution for a long time. However, *backward error analysis* (see, for example, Corless [1994]) shows that numerical solutions to IVPs often exactly solve a "nearby" problem. Here, *nearby* can mean either that the numerical solution follows an exact solution of (1) with slightly different initial conditions (see Hayes [2001] for a review), or that the numerical solution exactly solves a problem with the same initial condition but with a slightly perturbed (1). One form of the latter is *defect-based* backward error analysis, in which it can be shown that a continuous numerical solution $\tilde{\mathbf{y}}(t)$ exactly satisfies a problem of the form

$$\tilde{\mathbf{y}}'(t) = \mathbf{f}(t, \tilde{\mathbf{y}}(t)) + \delta(t), \tag{3}$$

$$\tilde{\mathbf{y}}(t_0) = \mathbf{y}_0, \tag{4}$$

where $\delta(t)$ is an n-dimensional vector-valued function called the *defect*. Of course, the form and size of the defect depends upon the $\mathbf{f}$ the algorithm used to compute the numerical solution $\tilde{\mathbf{y}}(t)$. In this article, we deal exclusively with explicit pth order Runge-Kutta methods. These methods have the advantage that, with just a few (sometimes zero) extra evaluations of $\mathbf{f}$ (called *function evaluations*), we can compute a polynomial interpolant between the discrete solution points that approximates the local solution to $O(h^p)$ or $O(h^{p+1})$ where h is the stepsize [Enright et al. 1986]. This sequence of polynomial interpolants forms a continuous piecewise polynomial solution approximation $\tilde{\mathbf{y}}(t)$. This piecewise polynomial can be differentiated giving $\tilde{\mathbf{y}}'(t)$, and then $\tilde{\mathbf{y}}(t)$ is by definition an exact solution to (3) where

$$\delta(t) \equiv \tilde{\mathbf{y}}'(t) - \mathbf{f}(t, \tilde{\mathbf{y}}(t)). \tag{5}$$

Note that in some cases $\tilde{\mathbf{y}}'(t)$ may not be defined at the meshpoints, but this can be accommodated in the analysis.

In generating the underlying discrete solution (which is then interpolated to form $\tilde{\mathbf{y}}(t)$), one cannot directly control the forward global error, but must be satisfied with attempting to control some measure of the *local error*. That is, if φ_h is the exact time-h solution operator—that is, $\varphi_h(\mathbf{y}_0)$ is the exact solution to (1, 2) at time t_0+h—and ϕ_h represents an approximate discrete solution computed by some algorithm using stepsize h, then the local error at $t = t_0 + h$ is

$$\Delta(h, \tilde{\mathbf{y}}_0) = \phi_h(\tilde{\mathbf{y}}_0) - \varphi_h(\tilde{\mathbf{y}}_0), \tag{6}$$

where $\tilde{\mathbf{y}}_0$ is the solution value at the beginning of the step. We attempt to ensure that $\|\Delta(h, \tilde{\mathbf{y}}_0)\| \leq \varepsilon$, where ε is the *desired maximum local error*. To control this error, one traditionally varies the stepsize in order to maintain the norm of the estimated local error near to the desired local error. In practice, reliably estimating the local error is relatively inexpensive. For example, if one is using a pth-order numerical method then the best that we can expect is that $\|\Delta(h, \tilde{\mathbf{y}}_0)\| \sim h^{p+1}$. We can then estimate the local error by comparing two integrations across the same interval using different stepsizes or different formulas. However, it is not difficult to show that this simple strategy can be deceived, in that both integrations can sometimes give similar but inaccurate results. When this happens, their difference is small, leading to a spuriously small estimate of the actual local error. Note that such an underestimate of the size of the local error will not necessarily result in an approximation with a large local error because other conservative heuristics governing the stepsize-choosing strategy will usually prevent such deceptions.

Another way to control local error is to control the defect directly, which is explicitly computable, at least pointwise as in (5) [Enright 1989a, 1989b, 1993]. In general, however, the defect can be a complicated function of both the problem and the numerical algorithm, making an estimate of the *maximum* defect across a timestep difficult and expensive to compute. One could, for example, sample the defect at many points across a timestep, and take the maximum of these as an estimate of the maximum defect. The likelihood of this being a severe underestimate of the maximum would depend on the number and location of the sample points, and would be very small as the number of sample points increased. However, this is very ad hoc and expensive since it requires extra evaluations of $\mathbf{f}$ at each point we wish to monitor the defect. Furthermore, without sufficient care in the selection of the interpolating scheme, it is possible to construct examples that deceive such an ad hoc scheme into thinking the maximum defect is smaller than it actually is, just as the above local error estimate can be deceived.

The class of methods we address in this article are a class of *Continuous Runge-Kutta* (CRK) methods where the approximate solution is a piecewise polynomial, $\mathbf{z}(t)$, which has an associated defect whose magnitude is bounded by a small multiple of the tolerance. In addition the "shape" of the defect, defined over a particular step, asymptotically depends only on the method, and not the problem being integrated or the location of the step. We use the term *shape* in the sense that two functions have the same shape if one is a constant multiple of the other. The "shape" of an individual function can then be defined by that function scaled by its maximum magnitude. Such a method can be developed at a relatively modest increase in cost (over that of standard

Table I. Number of Stages Per Step for CRKs Using an Asymptotically Correct Defect Estimate

Authors Order→	5	6	8
Enright [1989a]	4–9	8–10	
Higham [1989]	8–9		
Present work	11	14	24
Enright and Higham [1991]	21	32	60

implementations of discrete methods) and they have several advantages. First, since the shape of the defect is asymptotically independent of the problem being integrated, we can precompute (at compile time) the point in the step at which the maximum defect is expected to occur (at least asymptotically); at run-time, a reliable estimate of the maximum defect can be obtained with a single evaluation at that point. Second, since the maximum defect, and thus the local error per unit step [Stetter 1978], can be bounded by a small multiple of the user-specified tolerance, the methods can be used and the accuracy understood without a user knowing anything about the underlying formula, or the notion of local error or stepsize. Convergence as the user-specified tolerance approaches zero can be proved by relating the maximum defect (which is now directly controlled) to the maximum local error per unit step [Stetter 1978]. Thus, interfaces and calling sequences can and have been developed which are method independent.

Since the methods are continuous, they are also useful when visualization or estimates of the global error are of use or when alternative methods are to be investigated for solving the same problem.

This class of direct defect control methods is rigorously justified as $h \to 0$ and, in this manuscript, we investigate the validity of this approach over a range of nonasymptotic stepsizes that are typical of those arising in the solution of real problems. Other approaches could also be used, but the point of this article is that the cost of developing effective methods need not be that great. The work of Enright [1989a] was preliminary and based on the most natural and inexpensive optimal-order interpolation schemes that could be used for defect control. Higham [1989] also considered the use of asymptotically correct estimates. Enright and Higham [1991] considered an expensive technique that assumes parallel processing is available. The work we present here is intermediate: although more expensive than the methods of Enright [1989a] and Higham [1989], the methods we present achieve robustness comparable to the parallel methods [Enright and Higham 1991] at a lower cost than the parallel algorithms would require if implemented sequentially. In particular, Table I presents a sample of the number of sequential function evaluations required by the different approaches. Furthermore, as we have noted, an interpolant is most suitable if the shape of its defect is relatively constant as the stepsize ranges over the values that arise in typical applications. As we will show, the defects of the CRKs we analyze and implement here have such a relatively constant shape even for modestly large stepsizes, a property which has not been previously recognized. (See, for example, Figure 10, which depicts the worst case of nonconformity seen in our problems.)

We also note that, to our knowledge, the three methods described in this article (of orders 5, 6, and 8) comprise the first user-friendly implementation of direct defect control. The availability of both Fortran and *Matlab* versions of these experimental integrators makes them candidates for wide distribution and use.

The reliability of both local error and defect control depend on an implicit assumption that truncation error dominates roundoff. At stringent accuracy requests, this assumption may not be valid. We test this assumption and conclude that special care is necessary to reduce the effects of roundoff when using high-order formulas. This is particularly true if the formula requires many floating-point operations per step.

2. METHODS

2.1 Continuous, Explicit Runge-Kutta Methods

We are primarily concerned with explicit Runge-Kutta methods. A traditional pth-order, explicit, s-stage, discrete Runge-Kutta formula is described by its *Butcher Tableau*,

$$\begin{array}{c|c} \mathrm{c} & \mathrm{A} \\ \hline & w^T \end{array} = \begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s-1} & \\ \hline & w_1 & w_2 & \dots & w_{s-1} & w_s \end{array}, \tag{7}$$

which represents the discrete Runge-Kutta formula

$$\varphi_{t_i+h}(\tilde{\mathbf{y}}_i) \approx \phi_{t_i+h}(\tilde{\mathbf{y}}_i) \equiv \tilde{\mathbf{y}}_{i+1} = \tilde{\mathbf{y}}_i + h\sum_{j=1}^{s} w_j \mathbf{f}(t_i + c_j h, \tilde{\mathbf{Y}}_j),$$

where $\tilde{\mathbf{Y}}_j$ is a jth stage estimate of the solution at time $t_i + c_j h$,

$$\tilde{\mathbf{Y}}_j = \tilde{\mathbf{y}}_i + h\sum_{r=1}^{j-1} a_{jr}\mathbf{f}(t_i + c_r h, \tilde{\mathbf{Y}}_r).$$

For convenience, we let

$$\mathbf{k}_j = \mathbf{f}(t_i + c_j h, \tilde{\mathbf{Y}}_j),$$

giving

$$\tilde{\mathbf{y}}_{i+1} = \tilde{\mathbf{y}}_i + h\sum_{j=1}^{s} w_j \mathbf{k}_j.$$

To define a continuous Runge-Kutta method associated with this discrete formula, we determine an interpolant $\mathbf{u}_i(t)$ over each step of the integration[1]

[1]We sometimes delete the subscript i from $\mathbf{u}_i(t)$ when the meaning is clear from the context.

(see, for example, Enright et al. [1986]),

$$\mathbf{u}_i(t) = \tilde{\mathbf{y}}_i + h \sum_{j=1}^{\bar{s}} b_j(\tau)\mathbf{k}_j, \quad t \in [t_i, t_{i+1}], \tag{8}$$

where $\tau = (t - t_i)/h$, $\tau \in [0, 1]$, and

$$b_j(\tau) = \sum_{k=1}^{p+1} \beta_{jk}\tau^k. \tag{9}$$

The extra $(\bar{s} - s)$ stages and the polynomial coefficients β_{jk} are not unique and can be determined according to several different criteria; for a detailed and comprehensive discussion, see Verner [1993]. The interpolants that interest us are optimal order (in that they agree with the local solution to order $p + 1$), and their derivatives (and hence their defects) are accurate to order p. The interpolant, $\mathbf{u}_i(t)$, can be altered to form a new "improved" interpolant, $\mathbf{v}_i(t)$, at a relatively small increase in cost. The idea is based on the observation that by recomputing some of the $\mathbf{k}$'s using $\mathbf{u}(t)$ one can derive interpolants $\mathbf{v}(t)$ whose defect is both $O(h^p)$ and whose shape asymptotically approaches a multiple of a constant polynomial, independent of the problem being integrated. This is accomplished by ensuring that, for those stages $\mathbf{k}_j$ that are not accurate to $O(h^{p+1})$ but which do contribute to the definition of $\mathbf{u}(t)$ (i.e., the corresponding polynomial $b_j(\tau)$ in (8) is nonzero), we replace $\mathbf{k}_j$ by

$$\tilde{\mathbf{k}}_j = \mathbf{f}(t_i + c_j h, \mathbf{u}(t_i + c_j h)).$$

With this replacement, the coefficients β_{jk} (defining $\mathbf{v}_i(t)$) are identical to the coefficients defining $\mathbf{u}_i(t)$.

It has been shown [Enright 1989a, 1989b] that the leading term in the asymptotic expansion of the defect corresponding to $\mathbf{u}_i(t)$ satisfies

$$\delta(t) = G(\tau)h^p + O(h^{p+1}),$$

where

$$G(\tau) = q_1(\tau)F_1 + q_2(\tau)F_2 + \cdots + q_K(\tau)F_K. \tag{10}$$

The q_j are polynomials in τ that depend only on the method and the F_j are constants depending on both the method and the problem.

Now for a given problem and any method, as $h \to 0$, $G(\tau)$ approaches a constant polynomial (for each step i), and therefore we should observe plots of $\delta(\tau)$ versus τ approaching a unique polynomial as $h \to 0$. This polynomial will in general be very different for different problems and for different steps i, and this is why it is difficult to choose a fixed sample point τ_* that would give a robust estimate of the maximum defect across any step. For the polynomial $\mathbf{u}(t)$ we have used a value for τ_* that is not near any of the zeros of the $q_1, q_2, \ldots, q_K$ [Enright 1989b].

There is a special (but more expensive) class of interpolants (including the $\mathbf{v}_i(t)$ discussed a few paragraphs above) which have $K = 1$, and therefore the asymptotic expansion of the defect satisfies

$$\delta(t) = q_1(\tau)F_1h^p + O(h^{p+1}), \tag{11}$$

Table II. The Butcher Tableau of *Matlab*'s `ode45` (The nonboldfaced items (all but the last row of a_{ij} and the last entry of b_j) give the standard discrete tableau. The last row gives the extra evaluation from which *Matlab* builds an interpolant with associated local error that is O(h^5) and defect that is $O(h^4)$.)

0	0						
1/5	1/5	0					
3/10	3/40	9/40	0				
4/5	44/45	−56/15	32/9	0			
8/9	19372/6561	−25360/2187	64448/6561	−212/729	0		
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656	0	
1	**35/384**	**0**	**500/1113**	**125/192**	**−2187/6784**	**11/84**	**0**
	35/384	0	500/1113	125/192	−2187/6784	11/84	**0**

where $F_1 h^{p+1}$ is the leading term in the expansion of the discrete local error evaluated at the right endpoint, $\tau = 1$ [Enright 1989a]. That is, there is a direct asymptotic relationship between the discrete local error and the continuous defect. Since q_1 is independent of the problem, we should observe as $h \to 0$ that the shape of $\delta(t)$, and consequently the (norm of the) local maximum defect, is determined by the fixed polynomial q_1, independent of the problem and the step i. The value to use for τ_* is then the location of the maximum of $|q_1(\tau)|$, $\tau \in [0, 1]$.

For a certain class of linear problems [Enright 1989a, 1993], it is possible rigorously to bound the magnitude of the maximum defect with only a modest restriction on the size of the timestep. The point of this article is to demonstrate that one can develop an effective estimate of the magnitude of the maximum defect that is asymptotically justified for all problems and which also holds quite well for practical timesteps. A technique for computing the theoretical $q_1(\tau)$ for the 5/6 pair using *Maple* is described in the Appendix.

2.2 The 4/5 Pair

The Butcher Tableau for the well-known discrete 4/5 Runge-Kutta pair used by *Matlab*'s `ode45` is shown in Table II. The first six (nonboldfaced) rows represent the tableau and define $k_1, \ldots, k_6$ for the standard, discrete solution.

The standard (nonoptimal) $O(h^5)$ interpolating polynomial, $\mathbf{z}(t)$, (with a defect that is $O(h^4)$) used in `ode45` is defined by

$$\mathbf{z}(t) = \tilde{\mathbf{y}}_i + h \sum_{j=1}^{7} b_j(\tau)\mathbf{k}_j,$$

$$\begin{pmatrix} b_1(\tau) \\ b_2(\tau) \\ b_3(\tau) \\ b_4(\tau) \\ b_5(\tau) \\ b_6(\tau) \\ b_7(\tau) \end{pmatrix} = \begin{pmatrix} 1 & -183/64 & 37/12 & -145/128 \\ 0 & 0 & 0 & 0 \\ 0 & 1500/371 & -1000/159 & 1000/371 \\ 0 & -125/32 & 125/12 & -375/64 \\ 0 & 9477/3392 & -729/106 & 25515/6784 \\ 0 & -11/7 & 11/3 & -55/28 \\ 0 & 3/2 & -4 & 5/2 \end{pmatrix} \begin{pmatrix} \tau \\ \tau^2 \\ \tau^3 \\ \tau^4 \end{pmatrix}. \qquad (12)$$

To construct an $O(h^6)$ interpolant $\mathbf{u}(t)$, we add two more stages (see Enright [1989a, 1989b] for a justification of these particular choices),

$$\begin{aligned}\mathbf{k}_8 &= \mathbf{f}(t_i + .86h, \mathbf{z}(t_i + .86h)),\\ \mathbf{k}_9 &= \mathbf{f}(t_i + .93h, \mathbf{z}(t_i + .93h)),\end{aligned}$$

and then our $O(h^6)$ interpolant, $\mathbf{u}(t)$, has nine stages, with corresponding coefficients (8) defined by

$$\begin{pmatrix} b_1(\tau)\\ b_2(\tau)\\ b_3(\tau)\\ b_4(\tau)\\ b_5(\tau)\\ b_6(\tau)\\ b_7(\tau)\\ b_8(\tau)\\ b_9(\tau)\end{pmatrix} = \begin{pmatrix} 1 & -\frac{1708582621}{524156928} & \frac{1232939669}{262078464} & -\frac{1663764925}{524156928} & \frac{208375}{253952}\\ 0 & 0 & 0 & 0 & 0\\ 0 & \frac{499875}{94976} & -\frac{1618625}{142464} & \frac{871875}{94976} & -\frac{15625}{5936}\\ 0 & \frac{499875}{65536} & -\frac{1618625}{98304} & \frac{871875}{65536} & -\frac{15625}{4096}\\ 0 & -\frac{26237439}{6946816} & \frac{28319463}{3473408} & -\frac{45762975}{6946816} & \frac{820125}{434176}\\ 0 & \frac{43989}{28672} & -\frac{142439}{43008} & \frac{76725}{28672} & -\frac{1375}{1792}\\ 0 & -\frac{2291427}{100352} & \frac{3838251}{50176} & -\frac{8579075}{100352} & \frac{199625}{6272}\\ 0 & -\frac{47953125}{1078784} & \frac{74828125}{539392} & -\frac{155453125}{1078784} & \frac{78125}{1568}\\ 0 & \frac{8734375}{145824} & -\frac{14359375}{72912} & \frac{31234375}{145824} & -\frac{234375}{3038}\end{pmatrix} \begin{pmatrix}\tau\\ \tau^2\\ \tau^3\\ \tau^4\\ \tau^5\end{pmatrix}. \qquad (13)$$

Finally, to compute the interpolant $\mathbf{v}(t)$, we "recompute" $\mathbf{k}_8$ and $\mathbf{k}_9$, giving

$$\begin{aligned}\hat{\mathbf{k}}_8 &= \mathbf{f}(t_i + 0.86h, \mathbf{u}(t_i + 0.86h)),\\ \hat{\mathbf{k}}_9 &= \mathbf{f}(t_i + 0.93h, \mathbf{u}(t_i + 0.93h)).\end{aligned}$$

Let $\hat{\mathbf{k}}_j \equiv \mathbf{k}_j$ for those terms not recomputed, i.e., for $j = 1, \ldots, 7$. Then the asymptotically improved interpolant, which also has local error $O(h^6)$, is

$$\mathbf{v}(t_i + \tau h) = \tilde{\mathbf{y}}_i + h \sum_{j=1}^{9} b_j(\tau)\hat{\mathbf{k}}_j.$$

3. TESTING AND RESULTS

3.1 Testing Our Methods

We created improved interpolants and implemented the resulting modified methods for several integrators including the 4/5 pair that is `ode45` in *Matlab*, as well as methods implemented by the authors based on 5/6 and 7/8 RK formula pairs originally introduced by Verner [1993]. Each such method was extensively tested using the DETEST package [Pryce and Enright 1987], which assesses the performance of a method on a suite of 25 problems.

In addition, we closely studied the following one-dimensional problems taken (along with their names) from DETEST. These were chosen because they have closed-form solutions which facilitate easy evaluation of both local and global

Table III. The Meanings of Each Letter Used in the Names of Curves in Several Figures to Follow (Note that not all variations were used in all tests. For example, compensated summation was not necessary for the method `ode45`, and so we do not show that variation for `ode45`.)

(Ss)	Error/defect evaluated only at τ_* [S], or the max over $\tau \in [0..1]$ [s]. Ideally, we would like τ_* to be near the point of maximum defect.
(Cc)	Use compensated summation when summing Butcher Tableau? C=yes, c=no.
(Bb)	Precomputed accurate values of $b_j(\tau)$ of (9) and its derivative at $\tau = \tau_*$? B=yes, b=no.
(Ee)	Reduce roundoff error in (9) and its derivative by ordering terms in increasing absolute value at run-time? E=yes, e=no.
(Nn)	Reduce roundoff error in (9) and its derivative by compile-time ordering by increasing absolute value of the coefficients β_{jk}? N=yes, n=no.
(Qq)	Use QUAD precision to evaluate (9) and its derivative? Q=yes, q=no.

solutions:

$$\text{Problem A1: } y' = -y. \tag{14}$$

$$\text{Problem A2: } y' = -y^3/2. \tag{15}$$

$$\text{Problem A4: } y' = \frac{y}{4}\left(1 - \frac{y}{20}\right). \tag{16}$$

In addition, we used problem D3, which is the planar, two-body Kepler problem, which has four components: position x, y, and velocity v_x, v_y. The Kepler problem also has a closed-form solution.

For each problem and method above, we studied the effect of several variations. Since we want the magnitude of the defect at τ_* to be a good approximation of the maximum defect, we compare the two. Since the coefficients in the Butcher Tableau of the discrete Runge-Kutta formula and the coefficients defining the interpolant can both be quite large, we investigated the effect of *compensated summation*, which attempts to decrease the effects of roundoff error by performing all sums using an algorithm that partially extends precision beyond the hardware floating point precision (see, e.g., Higham [2002]). We also studied the effect of ordering the terms β_{jk} in (9) and its derivative (used in computing the defect) both by term size at run-time, and by coefficient size at compile-time. Finally, to compare against the best answer we can expect (when roundoff error dominates), we compared each of these variations to one that uses QUAD precision in all internal calculations. These variations (related to roundoff) are summarized in Table III.

Note that for a vector-valued **f** such as that associated with the Kepler problem, the defect is a also vector-valued function that changes from step to step. After some preliminary tests on many problems across many steps and accuracy requests, we found that looking at a single component of the defect on the first step of an integration reflected the typical situation that arose on each step of a system. For the Kepler problem, we also looked only at the first component of the vector, after verifying that all components were similar.

3.2 U-Shaped Curves

A useful tool to visualize both the order of an integration scheme and the effect of roundoff is the so-called *U-shaped curve*, or U-curve for short, in which we

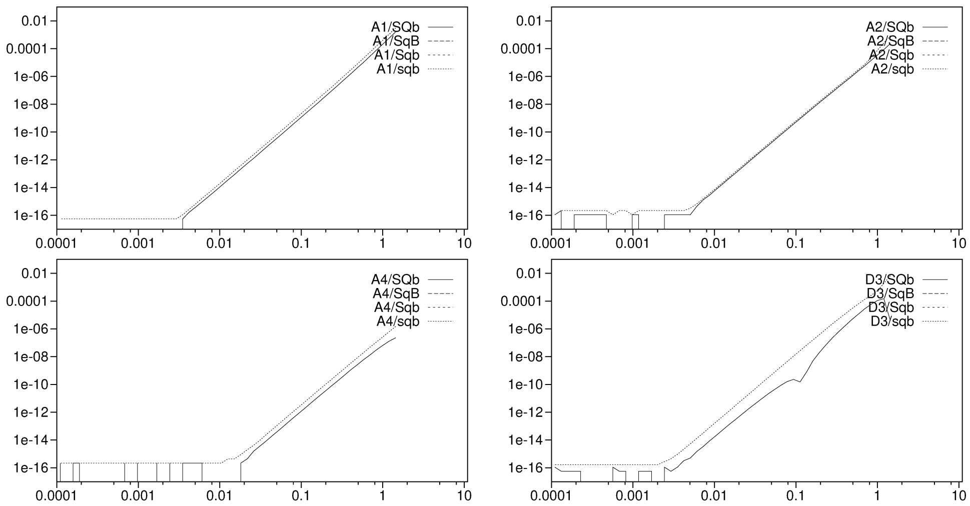

Fig. 1. Magnitude of the local error as a function of stepsize for the standard $O(h^5)$ interpolant, $\mathbf{z}(t)$, used by the *Matlab* integrator `ode45`. Each figure is for a different problem, A1, A2, A4, and D3, respectively. In all cases the slope of the curve (at least above the machine precision of 1e-16) is 5.0, which corresponds to the expected local order of the method. We plot the U-curves as computed in several different ways for comparison. The three letters of the name of the curve correspond to the options explained in the text and enumerated in Table III.

plot either the discrete local error or the maximum observed defect associated with a single step versus the stepsize. (Such a curve can be determined for any step of the integration. After some preliminary computations, we found that the curves for a given problem were generally qualitatively similar for all steps. As a result we have chosen to report the curves corresponding to the first step of the integration.) We compute the local error by comparing the approximate solution with the exact local solution for each problem. The slope of the U-curve on a log-log plot gives the observed order. In Figure 1 we plot the U-curve of the local error for the standard $O(h^5)$ interpolant $\mathbf{z}(t)$ used by *Matlab*'s `ode45` for our four problems. The value of τ_* for this method, with the interpolant as given in (12), is $\tau_* = 0.23$. (This value is chosen simply to avoid zeros of the polynomials $q_1, \ldots, q_K$ (10).) We plot the U-curves as computed in several different ways for comparison. The first three letters of the name of the curve in the legend correspond to the options described in Table III. As can be seen, for problems A1 and A2, the local error at τ_* (although we would not be able to measure it in practice) closely approximates the maximum error, but for problems A4 and D3 it is a significant underestimate. This is not surprising since τ_* is chosen based on attempting to control the defect rather than the local error (and the maximum local error usually occurs at the right endpoint, $\tau = 1$).

Figure 2 is similar to Figure 1 except we plot the U-curves of the defect rather than the local error. The defect is computed based on evaluation of (5) for the standard $O(h^5)$ interpolant. The U-shaped curves for both interpolants $\mathbf{u}(t)$ and $\mathbf{v}(t)$ (not shown) are qualitatively similar, except the order of the error and defect (and hence the slopes of the U-curves) are each higher by 1 than

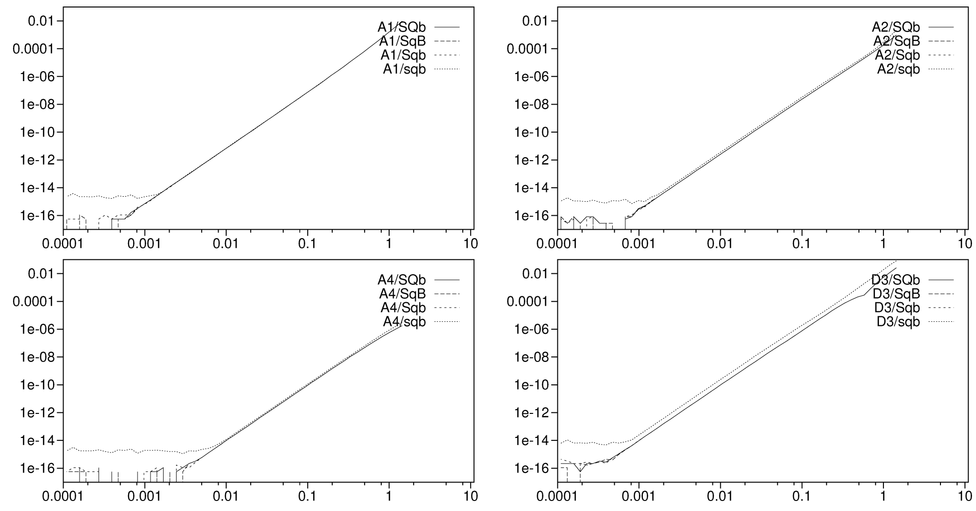

Fig. 2. Plots similar to Figure 1, except this time we plot the magnitude of the defect. The slopes of the curves are all about 4.0, because the defect of the *Matlab* `ode45` polynomial interpolant is $O(h^4)$. Note that for problems A1, A2, and A4, the defect at τ_* (curve `Sqb`) is virtually identical to the maximum defect across $\tau \in [0, 1]$ (curve `sqb`); however, for problem D3, the maximum defect occurs elsewhere and so our estimate at τ_* is a too low by about a factor of 3. Note also that roundoff affects the computation of the maximum defect near limiting precision, so that the leveling of the `sqb` curves near 1e-14 is due to roundoff. If we precompute accurate coefficients for the polynomial at τ_* (curve `SqB`), we eliminate these roundoff effects, allowing computation of the defect at τ_* as accurately as if we had used QUAD precision (curve `SQb`). No compensated summation was used.

in Figures 1 and 2. Note that for problem D3, the defect at τ_* is a significant underestimate of the maximum defect for both the standard $O(h^5)$ interpolant $\mathbf{z}(t)$ (shown) and for $\mathbf{u}(t)$ (not shown).

Figure 3 plots the shape of the defect of all four problems across $\tau \in [0, 1]$ for a typical stepsize (normalized so they all have the same magnitude). It clearly demonstrates that the shape of the defect—and in particular its maximum value—is different for the four problems for both the standard interpolant $\mathbf{z}(t)$, and for $\mathbf{u}(t)$. Thus there is no single choice of τ_* that will reliably estimate the maximum defect for all problems for these interpolants. The figure also demonstrates that the shape of the defect computed for $\mathbf{v}(t)$ is very similar across all four problems, and that the maximum occurs very close to the expected value of $\tau_* = 0.23$. This allows accurate, robust estimation of the maximum defect using only one function 0evaluation at τ_*.

3.3 The 5/6 Pair

The tableau for the 5/6 pair, along with the derivation of the $O(h^6)$ interpolant was described in Verner [1993], and implemented and investigated in Enright [1993]. There are eight stages to define this discrete explicit method. One extra function evaluation results in an interpolant of $O(h^6)$ accuracy, and an additional two evaluations define an optimal order interpolant, $\mathbf{u}(t)$, of $O(h^7)$ accuracy. Recomputing these final three stages (giving a total of six additional

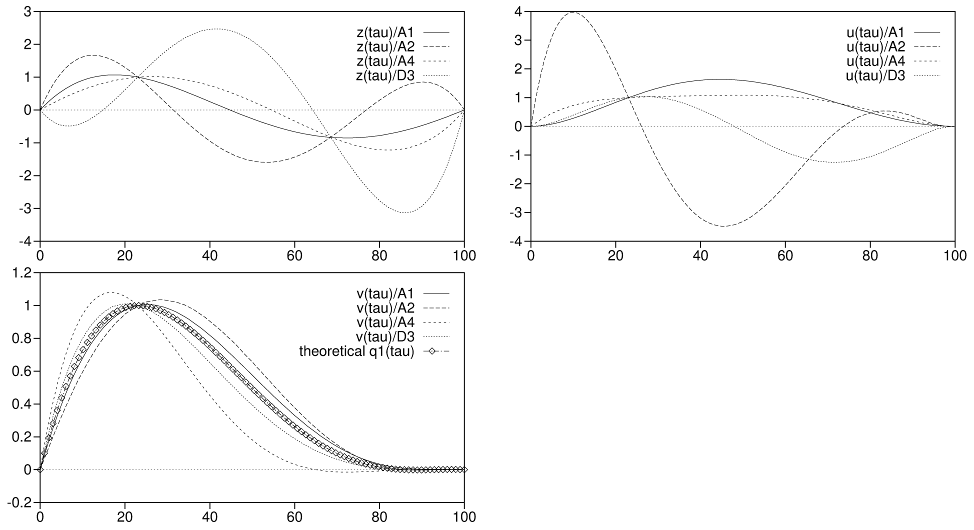

Fig. 3. The shape of the 4/5 pair defect curves for all four problems and three interpolants across an entire step from $\tau = 0$ to $\tau = 1$. The horizontal axis is labeled by percentage across the timestep, and the curves are normalized so that they are all unity at $\tau_* = 0.23$. The top-left plot is for the standard $O(h^5)$ interpolant $\mathbf{z}(t)$ of *Matlab*'s `ode45`, while the right plot is for $\mathbf{u}(t)$ of (13), which has local error $O(h^6)$. Note that the maxima (in magnitude) occur at different places for different problems, and that the max defect is often several times larger in magnitude than that at τ_*; in some cases (not shown) it was observed to be 40 times larger. The bottom plot is the for the interpolant $\mathbf{v}(t)$, which has $O(h^6)$ local error. Its defect has a very similar shape (for a typical stepsize) for all four problems, so that the defect at τ_* is very close to the maximum defect.

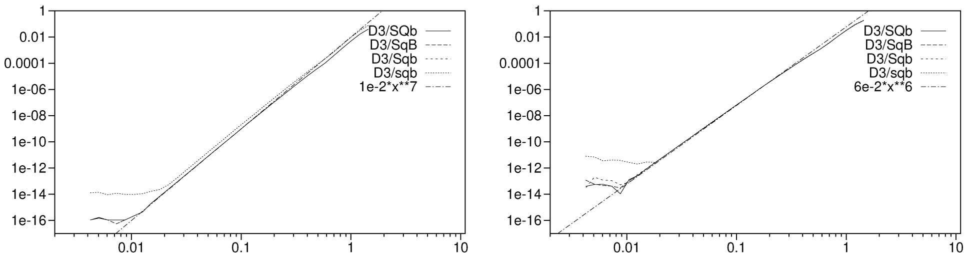

Fig. 4. U-shaped-curves for the 5/6 pair interpolant $\mathbf{u}(t)$ operating on the Kepler problem (problem D3); local error is on the left, and defect on the right. The letters describing the curves are from Table III. Also plotted are straight lines demonstrating that the local error of the improved interpolant is 7th order and the defect 6th order, respectively. As seen for the 4/5 pair, roundoff has a significant effect when interpolating at points other than at τ_*, and pre-computing the accurate polynomial at τ_* virtually eliminates roundoff at τ_*, giving results as good as if QUAD had been used. Similar effects are seen for the other problems.

function evaluations over the original eight) defines the interpolant, $\mathbf{v}(t)$. Figure 4 plots the U-curves for the interpolant $\mathbf{u}(t)$ for this 5/6 pair on the Kepler problem. Unlike the 4/5 pair, the defect at τ_* (right figure) is only a slight underestimate of the maximum defect across the timestep, at least for these four problems. As Figures 5, 6, and 7 demonstrate, however, the shape of the defect across a step for the $O(h^6)$ interpolant or for $\mathbf{u}(t)$ is still a strong

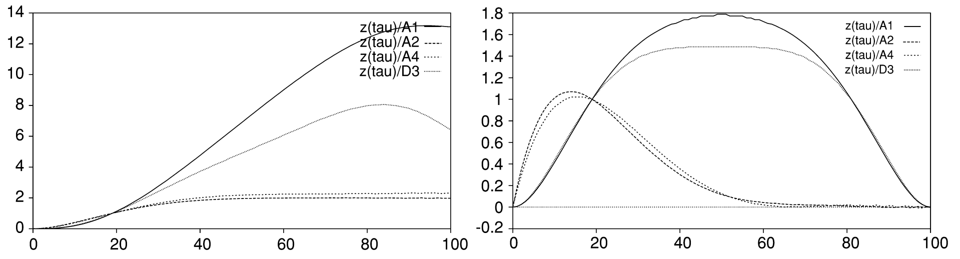

Fig. 5. Shape of the local error (left) and defect (right) across a step for the $O(h^6)$ interpolant for the 5/6 pair, operating on all four problems.

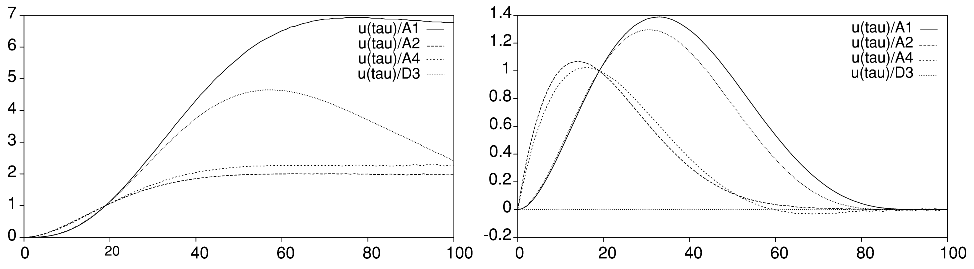

Fig. 6. Same as Figure 5, but for the $O(h^7)$ interpolant $\mathbf{u}(t)$.

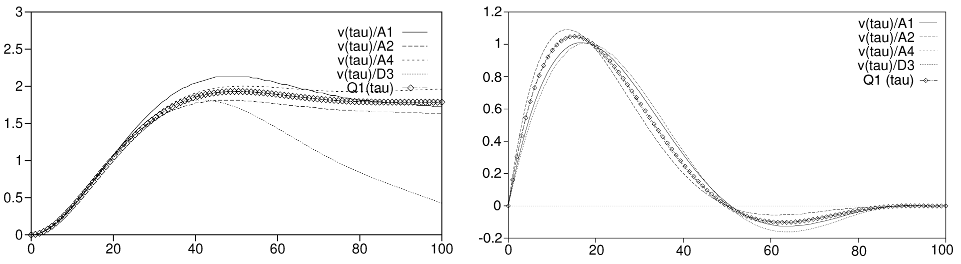

Fig. 7. Same as previous two figures but for the $O(h^7)$ interpolant $\mathbf{v}(t)$ (whose defect is $O(h^6)$), plus the theoretical $Q_1(\tau)$ and $q_1(\tau)$. Note the shape of the defect is almost problem independent, and corresponds well to the shape of the theoretical defect $q_1(\tau)$.

function of the problem, and so the sample at τ_* could potentially be worse for problems other than these four. (The τ_* for $\mathbf{u}(t)$ was 0.1943, based on avoiding zeros of the interpolating polynomial applied to linear problems, as described previously.) Figures 5, 6, and 7 plot the computed local error and defect across a single step for the 5/6 pair operating on all four problems. As with the 4/5 pair, we see that the shape of the defect for $\mathbf{v}(t)$ is almost problem independent, allowing a single evaluation of the defect at τ_* to give a reliable estimate of the maximum defect across the timestep. In Appendix A, we include *Maple* code for the derivation of $q_1(\tau)$, whose shape corresponds to that of the defect of $\mathbf{v}(t)$, and whose maximum occurs at $\tau_* \approx 0.1509$.

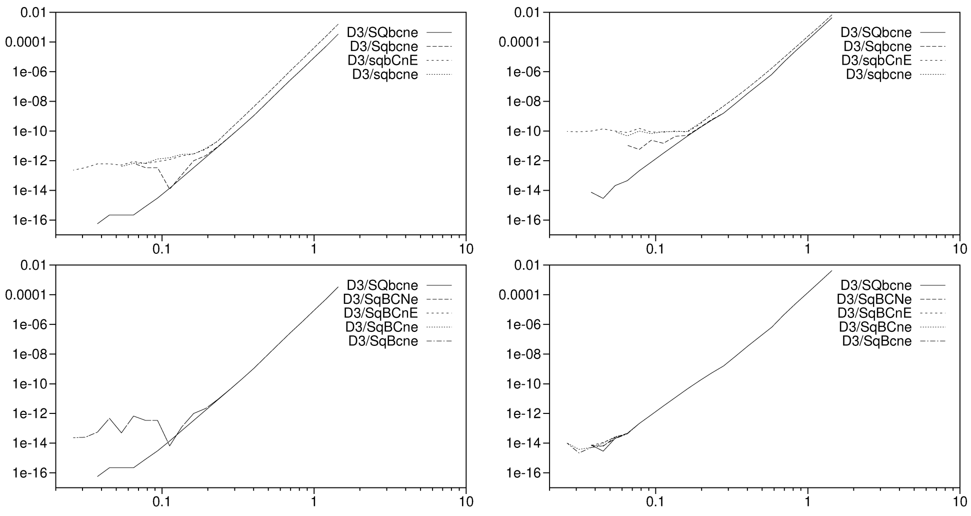

Fig. 8. U-shaped-curves for the 7/8 pair interpolant $\mathbf{u}(t)$ operating on the Kepler problem (problem D3). The letters describing the curves are from Table III. Local error is in the left two figures, defect in the right two figures. The top two figures compare QUAD at τ_* to all cases computing the maximum error and defect across $\tau_* \in [0, 1]$, *without* precomputing accurate values of the polynomial coefficients. The bottom two figures compare QUAD at τ_* to all cases computing the error and defect at τ_* *with* precomputing accurate polynomials $b_j(\tau_*)$ of (9).

3.4 The 7/8 Pair

The derivation of the tableaus for several 7/8 pairs was described in Verner [1993]. We have chosen one member of this family of Runge-Kutta formulas. (The actual formula pair chosen can be identified by specifying the abcissa vector, c. We do this in Appendix B.) Figure 8 reports several U-shaped curves associated with this 7/8 pair corresponding to problem D3. Local error is on the left, defect is on the right. The top two figures compare the QUAD-computed value of the error and defect at $\tau_* \approx 0.92$ to the maximum error and defect across the step, respectively. Two broad observations are clear from the top two plots: first, the error and defect at τ_* is a significant underestimate of the maximum across the step (which explains the gap between the QUAD and all other curves in both top diagrams); and second, there is significant roundoff error associated with evaluation of (9) and its derivative when the timestep is smaller than about 0.2 (which is why the top curves level off well above machine precision). Various methods of compensated summation (represented by the letters C and E in the labels, described in Table III) have surprisingly little effect in decreasing the roundoff in evaluating (9). The significant roundoff demonstrated in the top left figure (local error) results from evaluation of (9), which has several large magnitude ($\approx 5 \times 10^4$) polynomial coefficients for the 7/8 pair; the problem is exacerbated in the right figure (defect), where we evaluate the derivative of that polynomial, which has even larger magnitude coefficients (up to 4×10^5).

The bottom two figures in Figure 8 compare the QUAD-computed value of the error and defect at τ_* to the double-precision-computed values at τ_* with

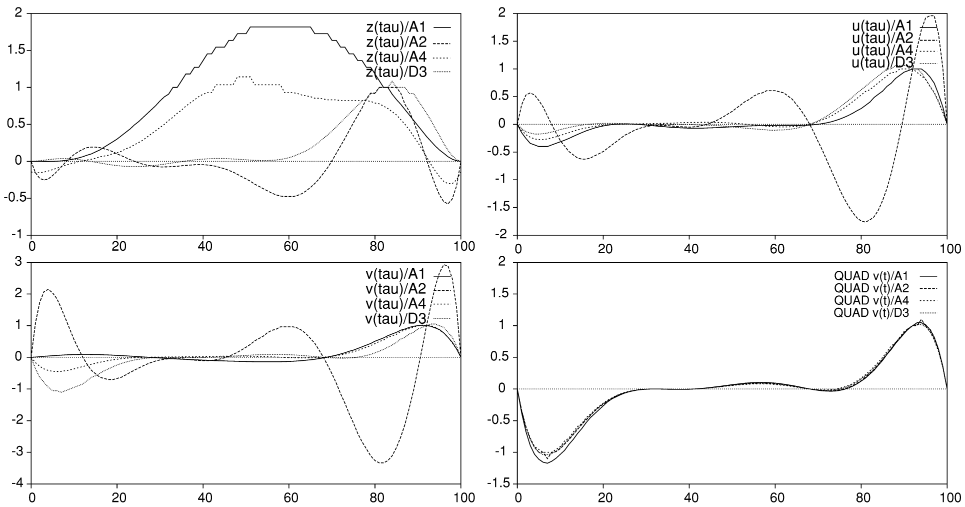

Fig. 9. Shape of the defect across a typical step for the interpolants of the 7/8 pair operating on all four problems. The interpolants are the nonoptimal $O(h^8)$ interpolant (top-left), $\mathbf{u}(t)$ (top-right), and $\mathbf{v}(t)$ (bottom-left) using stepsizes typical of double-precision integration. On the bottom-right, we replot the shape of $\mathbf{v}(t)$, using a smaller stepsize giving QUAD-precision accuracy for all four problems, to demonstrate the limiting behavior of $\mathbf{v}(t)$.

precomputed accurate values for the polynomials $b_j(\tau_*)$, including again various methods of compensated summation. These figures demonstrate that (i) precomputing accurate values of polynomials b'_j at τ_* (lines labeled with a capital B), which are needed to compute the estimate of the maximum defect, recovers effective QUAD precision (bottom-right figure), and (ii) again none of the types of compensated summation significantly improve the solution (both figures). In fact, the local error (bottom-left figure) is virtually identical for all the non-QUAD cases, demonstrating that compensated summation has surprisingly little effect. Similar results are seen for the other problems. DETEST results also confirmed this conclusion, in that no significant change in DETEST results is observable when compensated summation is added.

The shapes of the defects across a step for the 7/8 interpolants are plotted in Figure 9. Ironically, the 7/8 pair is so accurate that even when the local error and defect are close to the machine precision for problem A2, the stepsize is too large to reflect the asymptotic behavior of the optimal interpolant, although for the other three problems we see that the limiting polynomial is being approached. In the bottom-right figure, we resort to QUAD-precision integrations with defect tolerance set to `1e-25` in order to force all problems to have a small enough stepsize to demonstrate the limiting behavior of the defect associated with $\mathbf{v}(t)$, and in Figure 10, we demonstrate how this limiting polynomial is approached as the stepsize decreases for problem A2. This also provides a lesson: if the problem is too "easy" to integrate, then defect measurements at τ_* may be inappropriate even for the improved interpolant, because the stepsize is not small enough for the limiting behavior to be exhibited.

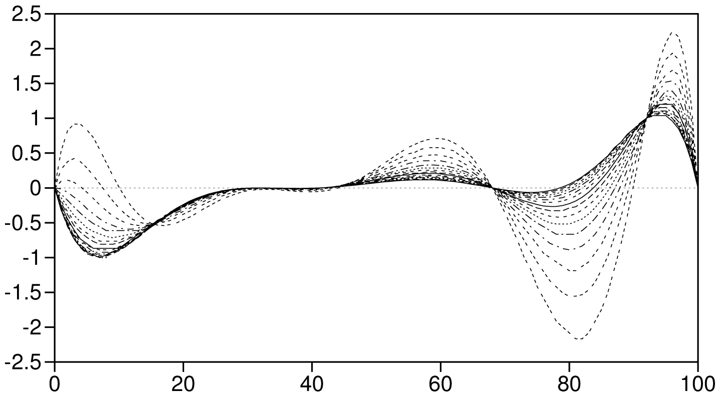

Fig. 10. How the shape of A2's defect curve for the 7/8 pair converges on the asymptotic shape as the timestep decreases. Sixteen stepsizes are plotted, ranging from 0.4823 decreasing successively by a factor of 1.2 down to 0.0313.

We note that QUAD precision versions of the interpolant and derivative are available to the user in our package, which is especially useful for the 7/8 pair. A full QUAD precision version of the 7/8 pair is also provided.

4. DISCUSSION

When using defect control, it is misleading to say we are using a "pair" of RK formulas, since we do not use the lower-order formula in our estimates. This allows us to avoid some of the function evaluations related only to the lower order of the pair, but of course we generally add more than we save. These methods tend to be more reliable and between about 30–70% more expensive than traditional implementations of the same formula pair. Note that, although we could be much more precise regarding the relative cost per step of the different implementations, they use different error control strategies and therefore the number of steps to solve a problem might be quite different. It should also be acknowledged that the sensitivity of the eighth-order method to roundoff error, although typical of high-order methods using defect control, only affects performance at very stringent tolerances (accuracy requests near the unit roundoff level). In general our double precision implementation of the eighth-order method is more effective than the sixth-order method at moderate to stringent tolerances.

Note that our analysis and our limited testing have shown that, without a rigorously justified technique for estimating the maximum defect (as is available with the improved interpolants, $\mathbf{v}(t)$), the estimate can be a severe underestimate. As this will likely only occur rarely and only on isolated steps, it will not necessarily affect the reliability of the integration. (In most step-selection schemes, constraints on the change in stepsize from step to step will reduce the potential impact of such an isolated deceived estimate.)

We inadvertently observed the effect of errors in the coefficients of the tableau on the accuracy of the integrations. In particular, near the beginning of this investigation our 7/8 pair had its coefficients computed using only double-precision accuracy and, because of ill-conditioning, they were accurate to only seven to eight significant figures. We later recomputed the coefficients in QUAD precision, giving coefficients accurate to 16 digits. We then tested both versions using DETEST (assessing the performance of the defect-based error control) and found very little difference in performance. Only at the most stringent error tolerances was there any difference (in some cases) and these differences were not significant. In particular, when a failure occurs in the original version, it occurs for the version with more accurate coefficients at the same point or very close by. In some cases, there is a bit of improvement in the accuracy obtained or in the number of steps to achieve a given accuracy, but overall the reported summary statistics are not significantly different. This could be a result of the fact that satisfying the order conditions exactly may not be as important as satisfying a small residual. We know that the original version had a small residual for the order conditions that we checked (even though the accuracy of the coefficients was low).

5. CONCLUSIONS

In this article, we have investigated several explicit, continuous Runge-Kutta methods in which the maximum defect across a step can be reliably and efficiently monitored and controlled. We have analyzed interpolants whose defect, in the limit of small stepsize, has a predicted shape dependent only on the integration method and not the problem being integrated. Although methods of similar functionality have been proposed in the past, ours is different in that the order of the interpolant is optimal relative to the order of the discrete formula. This results in a method in which the error control is both efficient and theoretically justified for all sufficiently smooth problems.

APPENDIX

A. DERIVATION OF $q_1(\tau)$ FOR THE 5/6 PAIR

The shape of the local error curve for the asymptotically justified interpolant $\mathbf{v}(t)$ for the 5/6 pair is denoted $Q_1(\tau)$. Its derivative, $q_1(\tau) = Q_1'(\tau)$, defines the shape of the defect curve, where $Q_1(\tau)$ is the unique polynomial of degree 6 satisfying

$$Q_1(1) = 1,$$

$$Q_1'(0) = Q_1(0) = Q_1'(1) = Q_1'(0.5) = Q_1'(0.9) = Q_1'(0.95) = 0,$$

where $\tau = 0.5,\ 0.9,$ and 0.95 are the points at the last stages of the interpolant that are recomputed as described earlier.

Note that if we have

$$Q_1(x) = b_0 + b_1 x^1 + \cdots + b_6 x^6$$

then the linear system that determines the b_i's (with the constraints defined above at 0 and 1) is simple to write. For the constraints at 1/2, 9/10, and 19/20, the linear equation contains powers of these points and corresponds to a Vandermond matrix. Using *Maple* to solve for $Q_1(\tau)$ for the 5/6 pair, we have

```
Q1:=x->b0+b1*x+b2*x^2+b3*x^3+b4*x^4+b5*x^5+b6*x^6;
 q1:=x->b1+2*b2*x+3*b3*x^2+4*b4*x^3+5*b5*x^4+6*b6*x^5;
 solve({q1(0)=0,Q1(1)=1,Q1(0)=0,q1(1)=0,q1(1/2)=0,q1(9/10)=0,q1(19/20)=0},
     {b0,b1,b2,b3,b4,b5,b6});
```

Maple then gives

```
b0 = b1 = 0, b2 = 513/17, b3 = -1766/17, b4 = 2478/17, b5 = -1608/17,
b6 = 400/17.
```

Solving for the zeros of $q_1'(t)$ allows us to determine that the maximum of $|q_1(x)|$ for $x \in [0, 1]$ occurs at $x \approx 0.15087$.

B. PRELIMINARY *MATLAB* VERSIONS OF THE INTEGRATORS

Our implementations of these methods are in Fortran with an interface modeled after DVERK [Hull et al. 1976]. To implement them in *Matlab*, we used *Matlab*'s MEX interface that allows Fortran programs to be directly interfaced with Matlab code.

The mapping between *Matlab*'s ODE `options` datatype and the options in our current Fortran implementations is not one-to-one; the only options that map cleanly are the suggested initial timestep `HSTART` and the maximum allowed timestep `HMAX`. Although our Fortran version can specify a smallest allowable timestep and the maximum number of steps, the *Matlab* `option` has no facility for specifying these. The tolerance specifications are also different. *Matlab* has both `RelTol` and `AbsTol`, while our Fortran implementations have only one tolerance, but five different ways to interpret it.

We have implemented an interface that is as similar to `ode23` and `ode45` as we can reasonably make it, except for events. We currently have three new integrators called `ode45d`, `ode56d`, and `ode78d`.

Both the Fortran and *Matlab* implementations are available from the authors on request, including a QUAD precision version of the 7/8 pair. For identification purposes (cf. Section 3.4), the **c** vector we use for the 7/8 pair is the one called `CVSS8` in Enright [1993], whose value is $c =$ [0, 139/2500, 473499/4616000, 1420497/9232000, 1923/5000, 923/2000, 769/5000, 8571/10000, 0.95052227954989854326408074615621 3, 3611/5000, 15/16, 1, 1, 1, 0.311017763495386386392741731882 91, 0.407, 0.103, 41/125, 0.41, 0.313, 0.681].

ACKNOWLEDGMENTS

We thank Jacek Kierzenka for help with Matlab MEX files, and Philip Sharp for routines to compute coefficients. This research was conducted at the Fields Institute for Research in Mathematical Sciences in Toronto during the Thematic Year on Numerical Computing.

REFERENCES

Corless, R. M. 1994. Error Backward. *Contemp. Math. 172*, 31–62.

Dahlquist, G. and Björck, Å. 1974. *Numerical Methods*. Automatic Computation Series. Prentice-Hall, Englewood Cliffs, NJ.

ENRIGHT, W. H. 1989a. Analysis of error control stategies for continuous Runge-Kutta methods. *SIAM J. Numer. Analys. 26*, 3, 588–599.

ENRIGHT, W. H. 1989b. A new error-control for initial value solvers. *Appl. Math. Comp. 31*, 288–301.

ENRIGHT, W. H. 1993. The relative efficiency of alternative defect control schemes for high-order continuous Runge-Kutta formulas. *SIAM J. Numer. Analys. 30*, 5, 1419–1445.

ENRIGHT, W. H. AND HIGHAM, D. J. 1991. Parallel defect control. *BIT 31*, 647–663.

ENRIGHT, W. H., JACKSON, K. R., NØRSETT, S. P., AND THOMSEN, P. G. 1986. Interpolants for Runge-Kutta formulas. *ACM Trans. Math. Softw. 12*, 193–218.

HAYES, W. B. 2001. *Rigorous Shadowing of Numerical Solutions of Ordinary Differential Equations by Containment*. Ph.D. dissertation. Department of Computer Science, University of Toronto, Toronto, Ont., Canada. Available on the Web as `http://www.cs.toronto.edu/NA/reports.html#hayes-01-phd`.

HIGHAM, D. J. 1989. Robust defect control with Runge-Kutta schemes. *SIAM J. Numer. Analys. 26*, 5, 1175–1183.

HIGHAM, N. J. 2002. *Accuracy and Stability of Numerical Algorithms*, 2nd ed. SIAM, Philadelphia, PA.

HULL, T. E., ENRIGHT, W. H., AND JACKSON, K. R. 1986. User's guide for DVERK—a subroutine for solving non-stiff ODEs. Tech. Rep. UT-DCS-100. Department of Computer Science, University of Toronto. Source code available from the authors, or online from `http://www.netlib.org/` in the ODE directory.

PRYCE, J. AND ENRIGHT, W. 1987. Two FORTRAN packages for assessing initial value methods. *ACM Trans. Math. Softw. 13*, 1–27.

STETTER, H. J. 1978. Considerations concerning a theory for ODE-solvers. In *Numerical Treatment of Differential Equations* (Proc. Oberwolfach 1976). Lecture Notes in Mathematics, vol. 631. Springer, Berlin/Heidelberg, Germany, 188–200.

VERNER, J. H. 1993. Differentiable interpolants for high-order Runge-Kutta methods. *SIAM J. Numer. Analys. 30*, 5, 1446–1466.

Received February 2005; revised October 2005; accepted March 2006