BIT 31 (1991), 647–663.

# PARALLEL DEFECT CONTROL

W. H. ENRIGHT and D. J. HIGHAM*

*Department of Computer Science, University of Toronto, Canada, M5S 1A4*

## Abstract.

How can small-scale parallelism best be exploited in the solution of nonstiff initial value problems? It is generally accepted that only modest gains in *efficiency* are possible, and it is often the case that "fast" parallel algorithms have quite crude error control and stepsize selection components. In this paper we consider the possibility of using parallelism to improve *reliability* and *functionality* rather than efficiency. We present an algorithm that can be used with any explicit Runge-Kutta formula. The basic idea is to take several smaller substeps in parallel with the main step. The substeps provide an interpolation facility that is essentially free, and the error control strategy can then be based on a defect (residual) sample. If the number of processors exceeds $(p-1)/2$, where $p$ is the order of the Runge-Kutta formula, then the interpolant and the error control scheme satisfy very strong reliability conditions. Further, for a given order $p$, the asymptotically optimal values for the substep lengths are independent of the problem and formula and hence can be computed a priori. Theoretical comparisons between the parallel algorithm and optimal sequential algorithms at various orders are given. We also report on numerical tests of the reliability and efficiency of the new algorithm, and give some parallel timing statistics from a 4-processor machine.

*AMS(MOS) Subject classification:* 65L05.

*Key words:* Runge-Kutta, parallelism, defect, interpolation.

## 1. Introduction.

Recent analysis of parallel algorithms for nonstiff initial value problems has revealed a number of negative results. For example, parallelism offers no advantage when one-step methods are used to solve the standard linear test problem [25]. Although careful algorithm design can produce modest speedups [2, 18, 19, 29], it is generally accepted that parallelism can be exploited only to a limited extent [2, 20, 25]. Also, in certain cases, the speedup is obtained at the cost of relaxed error control. In this work, rather than attempting to improve on the "straight-line" efficiency of standard sequential algorithms, we explore the possibility of using

* Both authors were supported by the Information Technology Research Centre of Ontario, and the Natural Sciences and Engineering Research Council of Canada. Second author's current address: Department of Mathematics and Computer Science, University of Dundee, Dundee DD1 4HN, Scotland.

Received January 1991. Revised May 1991.

small-scale parallelism to improve functionality and reliability. In particular we concentrate on the provision of interpolation and defect control.

The defect (residual) in a continuous approximation to the solution of an initial value problem is a useful theoretical tool [17, 26, 28], and Enright [6] recently suggested using a defect sample as the basis of an error control mechanism. Such an approach is intuitively reasonable – if the defect is small then a "nearby" problem has been solved. Furthermore, standard differential inequalities [13, page 56] show that the global error satisfies a method-independent bound and hence, in this sense, routines that control the defect can be thought of as interchangeable black boxes. Defect control is also one way to ensure tolerance proportionality; that is, an asymptotically linear relationship between the global error and the accuracy tolerance [17, 26].

In the next section we introduce an algorithm that provides asymptotically reliable interpolation and defect control. The issue of selecting the free parameters in the algorithm is discussed in section 3, and the results of a numerical search for optimal parameters are given. Section 4 gives a comparison of the efficiency of the new schemes with that of existing defect control schemes. Numerical tests that confirm our theoretical predictions are summarized in section 5. Finally, in section 6, we draw some overall conclusions and discuss possible extensions to this work.

## 2. The defect control algorithm.

Given the nonstiff initial value problem

$$y'(x) = f(x, y(x)), \qquad y(a) = y_a \in \mathbb{R}^N, \qquad a \le x \le b, \tag{2.1}$$

an $s$-stage explicit Runge-Kutta formula advances the numerical solution $y_n \approx y(x_n)$ over a step of length $h\,(= h_n)$ to $x_{n+1} := x_n + h$ by forming

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i, \tag{2.2}$$

where

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_i &= f\left(x_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right), \qquad 2 \le i \le s. \end{aligned} \tag{2.3}$$

It is useful, and in certain applications necessary, to provide a continuous approximation $p_n(x_n + \tau h) \simeq y(x_n + \tau h)$ for $\tau \in (0, 1]$. Good quality approximations will satisfy $p_n(x_n) = y_n$, $p_n'(x_n) = f(x_n, y_n)$, $p_n(x_{n+1}) = y_{n+1}$ and $p_n'(x_{n+1}) = f(x_{n+1}, y_{n+1})$, so that the corresponding piecewise approximation is globally $C^1$. Many such interpolation schemes have been derived recently, largely in conjunction with low order Runge-Kutta formulas (see, for example, [3, 5, 8, 12, 15, 21, 23]). The scheme

that we propose below falls into the class defined by Shampine [22]; see also Gladwell [11] for a related approach.

The defect $\delta_n(x)$ in $p_n(x)$ is the amount by which $p_n(x)$ fails to satisfy the ODE (2.1), that is

$$\delta_n(x) := p_n'(x) - f(x, p_n(x)).$$

Given an interpolant $p_n(x)$ it is possible to monitor the size of the defect in $p_n(x)$ on each step. Enright [6] discussed an error control scheme that attempts to keep the defect less than some user-supplied tolerance at all points in the range of integration. Here the defect is sampled at a single point $x_n + \tau^* h$ on every step, where $\tau^*$ is a fixed point in (0, 1), and $\|\delta_n(x_n + \tau^* h)\|$ is used to approximate $\max_{[0,1]} \|\delta_n(x_n + \tau h)\|$. If the norm of the defect sample is too large then the step is re-taken with a smaller stepsize. In general the shape of the defect over each step depends strongly upon $f$ and $x_n$. A heuristic approach for choosing $\tau^*$ was outlined in [6], and it was found that the resulting scheme was able to control the defect quite successfully; typically the sample underestimates the maximum defect over the step by a factor of less than 10. Some special defect control schemes were presented in [14] and [16]. Here the interpolants were derived in such a way that there is an asymptotically correct sample point $\tau^*$ that is independent of $f$ and $x_n$. These "robust" schemes control the defect more reliably than those of [6], but they do so at the expense of a considerably higher cost per step. In this work we aim to show that by exploiting parallelism, the cost of robust defect control can be reduced to a highly competitive level.

The idea that we propose is to take several smaller steps in parallel with the main step using the same Runge-Kutta formula. More precisely, we choose an integer $k$, and take $k - 1$ parallel steps of length $\sigma_i h$ to give

$$y_{n+\sigma_i} \simeq y(x_n + \sigma_i h), \quad i = 1, 2, \ldots, k, \tag{2.4}$$

where $0 = \sigma_1 < \sigma_2 < \ldots < \sigma_k = 1$. The corresponding $f$-values $f_{n+\sigma_i} = f(x_n + \sigma_i h, y_{n+\sigma_i})$ will also be formed. We can now let $p_n(x)$ be the Hermite interpolating polynomial of degree $\leq 2k - 1$ to the data $\{x_n + \sigma_i h, y_{n+\sigma_i}, f_{n+\sigma_i}\}_{i=1}^{k}$.

A suitable choice for $k$ can be made by analyzing the local error and defect in $p_n(x)$. We write $p_n(x)$ in normalized form

$$p_n(x_n + \tau h) = \sum_{i=1}^{k} d_i(\tau) y_{n+\sigma_i} + h \sum_{i=1}^{k} e_i(\tau) f_{n+\sigma_i}, \tag{2.5}$$

where $d_i(\tau)$ and $e_i(\tau)$ are scalar polynomials in $\tau$. (For example, $d_1(\tau)$ satisfies $d_1(\sigma_1) = 1$, $d_1(\sigma_i) = 0$ for $i = 2, \ldots, k$, and $d_1'(\sigma_i) = 0$ for $i = 1, \ldots, k$.) Similarly, we let $Q_n(x)$ denote the corresponding interpolant to the exact local data:

$$Q_n(x_n + \tau h) = \sum_{i=1}^{k} d_i(\tau) u_n(x_n + \sigma_i h) + h \sum_{i=1}^{k} e_i(\tau) u_n'(x_n + \sigma_i h), \tag{2.6}$$

where $u_n(x)$, the local solution for the step, satisfies

$$u_n'(x) = f(x, u_n(x)), \qquad u_n(x_n) = y_n.$$

If a $p$th order Runge-Kutta formula is used, then standard truncation analysis (see, for example, [13], page 158) shows that the local error in $y_{n+1}$ has the form

$$\text{(2.7)} \qquad \text{le}_{n+1} := y_{n+1} - u_n(x_{n+1}) = h^{p+1} \sum_{j=1}^{\lambda_p} A_j F_j + O(h^{p+2}),$$

where the scalar truncation coefficients $A_j$ depend only on the Runge-Kutta formula, and the elementary differentials $F_j$ involve partial derivatives of $f$ evaluated at $(x_n, y_n)$. Also, the local error in the data in (2.4) has the general form

$$\text{(2.8)} \qquad y_{n+\sigma_i} - u_n(x_n + \sigma_i h) = \sigma_i^{p+1} \text{le}_{n+1} + O(h^{p+2}).$$

If $f$ satisfies a Lipschitz condition in $y$ then it follows from (2.7) and (2.8) that

$$\text{(2.9)} \qquad f_{n+\sigma_i} - u_n'(x_n + \sigma_i h) = O(h^{p+1}).$$

Subtracting (2.6) from (2.5) and using (2.8) and (2.9) we obtain the following expression for the *data error* in $p_n(x)$

$$\text{(2.10)} \qquad p_n(x_n + \tau h) - Q_n(x_n + \tau h) = \text{le}_{n+1} \sum_{i=1}^{k} \sigma_i^{p+1} d_i(\tau) + O(h^{p+2}).$$

A well-known result from interpolation theory (see, for example, [4, page 67]) shows that the *interpolation error* has the form

$$\text{(2.11)} \qquad Q_n(x_n + \tau h) - u_n(x_n + \tau h) = K h^{2k} \prod_{i=1}^{k} (\tau - \sigma_i)^2 + O(h^{2k+1}),$$

where $K$ depends on $f$ and $(x_n, y_n)$, but is independent of $\tau$ and $h$. Combining (2.10) and (2.11) we find that the local error in the interpolant satisfies

$$\text{(2.12)} \qquad p_n(x_n + \tau h) - u_n(x_n + \tau h) = O(h^{\min\{2k,\, p+1\}}).$$

Choosing $2k > p + 1$, the data error dominates in (2.12) and we have

$$\text{(2.13)} \qquad p_n(x_n + \tau h) - u_n(x_n + \tau h) = \text{le}_{n+1} \sum_{i=1}^{k} \sigma_i^{p+1} d_i(\tau) + O(h^{p+2}).$$

In this case we see that, asymptotically, the local error at any point within the step is related to the local error in $y_{n+1}$ in a problem-independent way. A similar analysis shows that with $2k > p + 1$ the defect satisfies

$$\text{(2.14)} \qquad \delta_n(x_n + \tau h) = \frac{\text{le}_{n+1}}{h} \sum_{i=1}^{k} \sigma_i^{p+1} d_i'(\tau) + O(h^{p+1}).$$

Hence the asymptotic shape of the defect is known a priori, and we have a "robust" defect control scheme available. Note that the expressions for the local error and defect in $p_n(x)$ will only have the simple forms (2.13) and (2.14) if the data

error dominates *and* the same Runge-Kutta formula is used for each parallel substep.

The interpolation scheme above is designed to be used on a machine with (at least) $k-1$ processors. Any Runge-Kutta formula of order $p < 2k-1$ can be used. In particular, if the extra processors are available, a "parallel" Runge-Kutta formula could be used as the basic formula [18, 20]. We see from the form of the defect in (2.14) that an asymptotically correct estimate of $\max_{[0,1]} \|\delta_n(x_n + \tau h)\|$ is given by sampling the defect at a point $\tau^*$ where the polynomial $\sum_{i=1}^{k} \sigma_i^{p+1} d_i'(\tau)$ achieves its maximum modulus over $[0, 1]$. Although it is possible to sample the defect at several points in parallel, our tests have shown that a single sample gives a reliable defect estimate, even at quite lax error tolerances. Hence, we have not pursued the use of parallel sampling to estimate the maximum defect.

We mention that once a defect sample has been made, an asymptotically correct estimate of the local error $\mathrm{le}_{n+1}$ can be constructed, since from (2.14) we have

$$h \frac{\delta_n(x_n + \tau^* h)}{\sum_{i=1}^{k} \sigma_i^{p+1} d_i'(\tau^*)} = \mathrm{le}_{n+1} + O(h^{p+2}).$$

An estimate of the local error in the interpolant at any intermediate point could also be formed, using (2.13). We do not propose the use of this extra information in any direct way, since we are reluctant to forsake the $C^1$ continuity of the interpolant and the robust nature of the defect control. However, the information could be used in order to provide an inexpensive global error estimate by means of a defect correction strategy, along the lines of [27].

## 3. Optimal parameters.

Once the order, $p$, and the number of parallel substeps, $k-1$, have been specified, values must be chosen for $\sigma_2, \ldots, \sigma_{k-1}$. In this section we consider how this choice can be made. Similar strategies for choosing "optimal" parameters in a robust defect control algorithm were used in [14, 16].

Defining

$$g(\tau) := \sum_{i=1}^{k} \sigma_i^{p+1} \mathrm{d}_i(\tau), \tag{3.1}$$

we see from (2.14) that for any vector norm $\|\cdot\|$

$$\|\delta_n(x_n + \tau h)\| = \frac{\|\mathrm{le}_{n+1}\|}{h} |g'(\tau)| + O(h^{p+1}). \tag{3.2}$$

From the point of view of obtaining the smallest defect for a given stepsize, it is clear that a problem-independent asymptotically optimal scheme arises when $\mathrm{gpmax} := \max_{[0,1]} |g'(\tau)|$ is minimized.

It is possible to gain some insight into this minimax problem. First, since $g(0) = 0$ and $g(1) = 1$, the Mean Value Theorem shows that $g'(\xi) = 1$ for some $\xi \in (0, 1)$, giving the lower bound gpmax $\geq 1$. To get an upper bound on the optimum gpmax we consider the case where $\sigma_j \to 0$ for $j = 2, \ldots, k-1$. Using the Lagrange basis function

$$l_i(\tau) = \prod_{j=1, j\neq i}^{k} \frac{\tau - \sigma_j}{\sigma_i - \sigma_j},$$

we may write $d_i(\tau)$ in the form

$$d_i(\tau) = [1 - 2(\tau - \sigma_i)l_i'(\sigma_i)]l_i(\tau)^2.$$

(It is straightforward to verify that $d_i(\sigma_i) = 1$, $d_i(\sigma_j) = 0$ for $i \neq j$, and $d_i'(\sigma_j) = 0$.) Hence,

$$\begin{aligned} d_i'(\tau) &= -2l_i'(\sigma_i)l_i(\tau)^2 + [1 - 2(\tau - \sigma_i)l_i'(\sigma_i)]2l_i(\tau)l_i'(\tau) \\ &= -2\left\{\sum_{j=1, j\neq i}^{k} \frac{1}{\sigma_i - \sigma_j}\right\} \prod_{j=1, j\neq i}^{k} \left(\frac{\tau - \sigma_j}{\sigma_i - \sigma_j}\right)^2 \\ &\quad + 2\left[1 - 2(\tau - \sigma_i) \sum_{j=1, j\neq i}^{k} \frac{1}{\sigma_i - \sigma_j}\right] \prod_{j=1, j\neq i}^{k} \left(\frac{\tau - \sigma_j}{\sigma_i - \sigma_j}\right) \\ &\quad \times \left\{\prod_{j=1, j\neq i}^{k} \frac{1}{\sigma_i - \sigma_j}\right\} \sum_{r=1, r\neq i}^{k} \left\{\prod_{j=1, j\neq i, r}^{k} (\tau - \sigma_j)\right\}. \end{aligned} \tag{3.3}$$

Recall that $0 = \sigma_1 < \sigma_2 < \ldots < \sigma_{k-1} < \sigma_k = 1$. For a fixed $i \in \{1, 2, \ldots, k-1\}$ we see that there are $2k - 3$ factors of the form $1/(\sigma_i - \sigma_j)$, where $j \neq k$, in the two terms on the right-hand side of (3.3). It follows that for $j \in \{2, 3, \ldots, k-1\}$ if we let the sigmas tend to zero uniformly, in the sense that $\sigma_j \to 0$, $\sigma_j/\sigma_{j+1}$ remains bounded above by a number less than 1, and $\sigma_2/\sigma_{k-1}$ remains bounded below by a number greater than zero, then $\sigma_i^{p+1} d_i'(\tau) \to 0$, provided $p + 1 > 2k - 3$. Hence, from (3.1),

$$\lim_{\{\sigma_j\}_{j=2}^{k-1} \to 0} g'(\tau) = \lim_{\{\sigma_j\}_{j=2}^{k-1} \to 0} d_k'(\tau) = (2k-1)(2k-2)\tau^{2k-3}(1-\tau). \tag{3.4}$$

Elementary calculus then shows that this limit function has a unique maximum in $[0, 1]$ of

$$\text{gpmax} = (2k-1)\left(\frac{2k-3}{2k-2}\right)^{2k-3}, \text{ at } \tau^* = \frac{2k-3}{2k-2}. \tag{3.5}$$

Provided the order $p$ is chosen so that $2k - 3 < p + 1 < 2k$, this gives an upper bound on the optimum gpmax.

Table 1. *Parameters from numerical search.*

| | $\{\sigma_i\}_{i=1}^{k}$ | gpmax | $\tau^*$ | $\max_i\{\sigma_i \lvert d_i(\tau^*)\rvert, \lvert e_i(\tau^*)\rvert\}$ | $\max_i\{\sigma_i \lvert d_i'(\tau^*)\rvert, \lvert e_i'(\tau^*)\rvert\}$ |
|---|---|---|---|---|---|
| $k=4, p=5$ | 0, .2, .4, 1 | 4.1 | .88 | 4.0 | 37.3 |
| $k=4, p=6$ | 0, .2, .4, 1 | 3.9 | .88 | 4.2 | 35.6 |
| $k=5, p=7$ | 0, .2, .4, .7, 1 | 7.0 | .94 | 0.7 | 6.7 |
| $k=5, p=8$ | 0, .2, .4, .6, 1 | 6.7 | .93 | 3.5 | 56.4 |

We performed numerical searches for optimum parameter values for the cases $(k=4, p=5), (k=4, p=6), (k=5, p=7)$ and $(k=5, p=8)$. We used a grid search, refining the $\{\sigma_2, \ldots, \sigma_{k-1}\}$ grid in regions where the smallest gpmax appeared to reside. In each of the four cases our "solution" converged to the limiting values derived above. For $k=4$ this lead to gpmax $= 2.813$ and $\tau^* = .833$, and for $k=5$ it lead to gpmax $= 3.534$, and $\tau^* = .875$. Our numerical evidence strongly suggests that these minimax problems have infima given by the limiting case in (3.5).

In practice, of course, it is undesirable to have the $\sigma_i$ too close together. Intuitively, we would like to use information that is well spaced out across the step from $x_n$ to $x_{n+1}$. Also, as we will discuss later, there are potential rounding error difficulties in the evaluation of $p_n(x)$ and $p_n'(x)$ associated with close $\sigma_i$ values. For this reason we decided to perform a restricted numerical search where the sigmas were spaced out by at least 0.1. After some experimentation with the resulting defect control algorithms on some test ODEs, we found that a 0.2 spacing gave a reasonable compromise between efficiency (small gpmax) and susceptibility to rounding errors. The parameters that we chose for the four cases are given in Table 1. The table also gives the corresponding gpmax and $\tau^*$ values. It can be seen that with $k=4$ the gpmax values are within a factor 1.5 of the limiting value of 2.8. For $k=5$, with $p=7$ and $p=8$, gpmax comes within a factor 2 of the 3.5 limit. The last two columns in Table 1 relate to the sensitivity of the defect sample to rounding errors. They will be discussed in more detail at the end of this section.

In Figures 1 and 2 we plot the polynomials $g'(\tau)$ for the coefficients in Table 1. The polynomials are normalized so that $|g'(\tau^*)| = 1$. These figures show the asymptotic shape of the defect over each step. The shapes for the limiting cases $\sigma_i \to 0$ are also included. For comparison, Figure 3 shows the defect arising from an actual integration. Here the $(k=4,\ p=6)$ mode was used on a two-component Fehlberg test problem with a tolerance of TOL $= 10^{-2}$ (see [10] for more details). We chose (arbitrarily) to plot the defect over the step from $x_{19}$ to $x_{20}$.

Equation (2.13) shows that the behavior of the local error in the interpolant is determined by $g(\tau)$. These polynomials are plotted in Figures 4 and 5. In each case we have the pleasing result that $|g(\tau)| < 1$ for $\tau \in (0, 1)$. This means that the asymptotic local error at intermediate points in the step is always less than that at the end of the step, so we can regard the interpolant as being at least as accurate as the underlying formula.

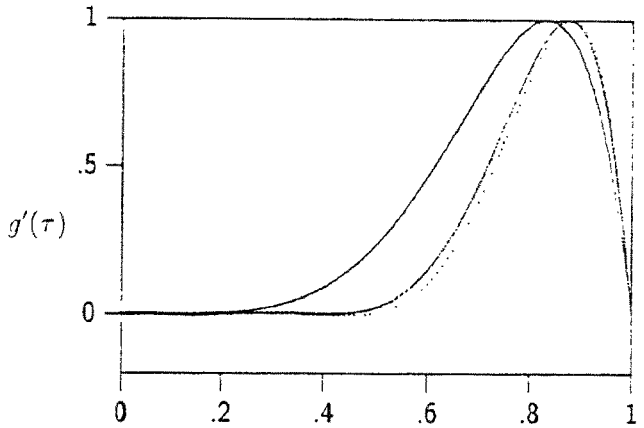


Fig. 1. Normalized defect polynomials, $k = 4$: dotted line is $p = 5$, dashed line is $p = 6$, solid line is limiting case.

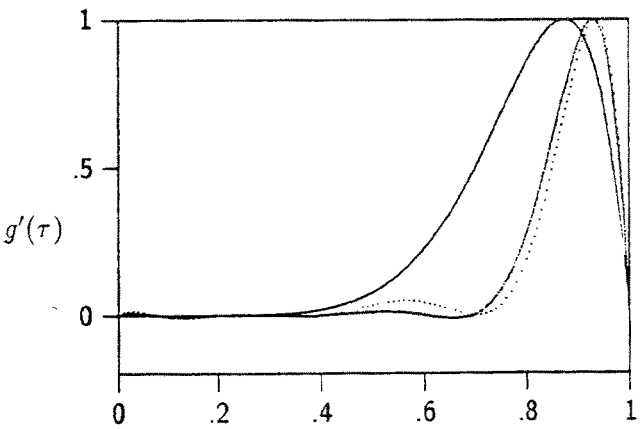


Fig. 2. Normalized defect polynomials, $k = 5$: dotted line is $p = 7$, dashed line is $p = 8$, solid line is limiting case.

We mention that by choosing $\{\sigma_i\}_{i=2}^{k-1}$ to make gpmax small we are making the algorithm efficient in the sense of defect versus cost (where cost is measured in terms of the total number of $f$ evaluations, or the total number of steps.) We are therefore using the defect in $p_n(x)$ as our measure of accuracy. An alternative way to measure the efficiency, which is perhaps more relevant when other types of error control are being studied, is to look at the global error versus the cost. In this latter sense, it can be argued that varying $\{\sigma_i\}_{i=2}^{k-1}$ will not affect the efficiency. To see this, suppose that we implement the algorithm with two different sets of $\{\sigma_i\}$ parameters, set A and set

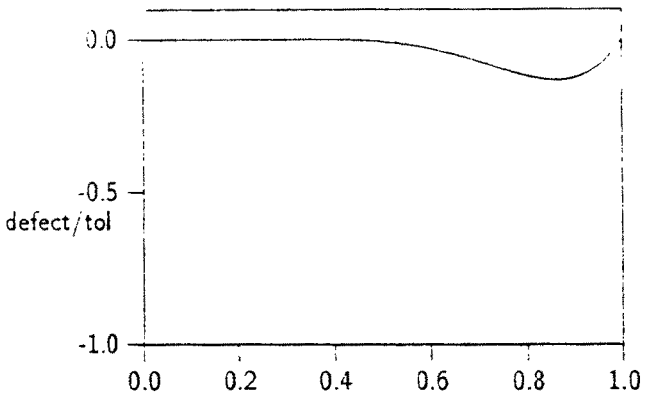


Fig. 3. Components of the defects ($k = 4, p = 6$) method on the Fehlberg problem with TOL $= 10^{-2}$, over the step from $x_{19}$ to $x_{20}$.

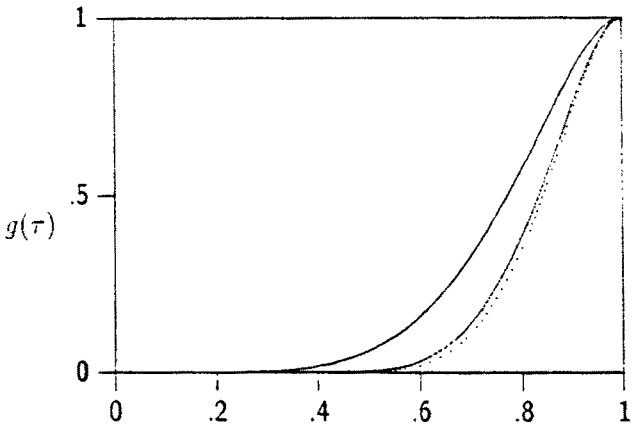


Fig. 4. Local error polynomials, $k = 4$: dotted line is $p = 5$, dashed line is $p = 6$, solid line is limiting case.

B, and let the maxima of the corresponding defect polynomials satisfy $|g'_A(\tau_A^*)| = C\,|g'_B(\tau_B^*)|$. Let us also make the idealized assumptions that the higher order terms in (3.2) are negligible, and that the algorithms are implemented in such a way that the stepsizes used make the maximum defect norm exactly equal to the user-supplied tolerance on each step. Then it follows that set A with a tolerance of TOL will generate exactly the same stepsize sequence, and hence the same discrete solution, as set B with a tolerance of TOL/$C$. In other words, both algorithms deliver

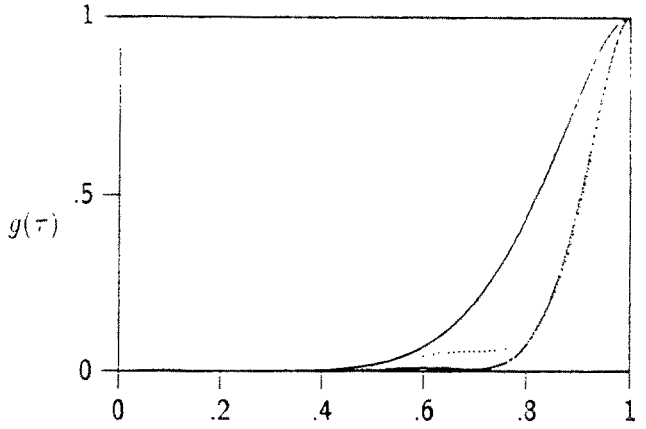


Fig. 5. Local error polynomials, $k = 5$: dotted line is $p = 7$, dashed line is $p = 8$, solid line is limiting case.

the same (discrete) global error for the same cost. We believe, however, that the first measure of efficiency (defect versus cost) is the more realistic way to compare defect control algorithms, since after a single integration a user can infer the size of the defect, but not the global error.

We conclude this section by discussing the form in which the interpolant and its derivative should be evaluated. Although each of the $k - 1$ parallel steps generates $s - 1$ new samples of $f$, we really need only two vectors of information, such as $y_{n+\sigma_i}$ and $f_{n+\sigma_i}$. Equation (2.5) could be used, along with

$$\frac{d}{dx} p_n(x_n + \tau h) = \frac{1}{h} \sum_{i=1}^{k} d_i'(\tau) y_{n+\sigma_i} + \sum_{i=1}^{k} e_i'(\tau) f_{n+\sigma_i}. \tag{3.6}$$

However, this formulation is likely to introduce unnecessary rounding errors; the sum $\sum_{i=1}^{k} d_i'(\tau) y_{n+\sigma_i}$ involves quantities of order $y$ but has a result of order $h$. It follows that the rounding errors in (3.6) will be $O(h^{-1})$. A more stable approach is to use the data $\Phi_i$ and $f_{n+\sigma_i}$, where $\Phi_i$ is the increment that satisfies

$$y_{n+\sigma_i} = y_n + \sigma_i h \Phi_i.$$

(Note that $\Phi_i$ is some linear combination of $f$ values.) The interpolant and its derivative may then be written

$$p_n(x_n + \tau h) = y_n + h \sum_{i=1}^{k} [\sigma_i d_i(\tau) \Phi_i + e_i(\tau) f_{n+\sigma_i}], \tag{3.7}$$

$$\frac{d}{dx} p_n(x_n + \tau h) = \sum_{i=1}^{k} [\sigma_i d_i'(\tau) \Phi_i + e_i'(\tau) f_{n+\sigma_i}]. \tag{3.8}$$

By dealing directly with the $f$ values we avoid the $h^{-1}$ rounding errors associated

with (3.6). However, if some of the coefficients in $\{\sigma_i d_i(\tau), e_i(\tau)\}$ or $\{\sigma_i d_i'(\tau), e_i'(\tau)\}$ are large in modulus then there is still a danger of significant rounding errors in the summations in (3.7) or (3.8). This is particularly important at the point $\tau = \tau^*$, since the defect sample must be made sufficiently small in order for the integration to proceed. Unnecessary rounding errors in the formation of the defect, $p_n'(x_n + \tau^* h) - f(x_n + \tau^* h, p_n(x_n + \tau^* h))$, will artificially limit the range of tolerances at which the algorithm can successfully operate. In Table 1 we include the maximum values of $\{\sigma_i |d_i(\tau^*)|, |e_i(\tau^*)|\}$, and $\{\sigma_i |d_i'(\tau^*)|, |e_i'(\tau^*)|\}$. We found that with the sigmas more closely spaced the algorithm "failed" (in the sense that it was unable to produce a sufficiently small defect sample) at larger tolerance values. This is to be expected, since as $\sigma_i \to 0$ for $i = 2, \ldots, k-1$ the corresponding $d_i(\tau)$ and $d_i'(\tau)$ become arbitrarily large (see (3.3)).

## 4. Theoretical comparisons.

In this section we compare our parallel defect control technique, which we will denote PDEF, with two other defect control techniques that have been proposed. One of these alternative techniques is inexpensive and is based on the use of a defect estimate that is not asymptotically valid and the other is relatively expensive and is based on direct control of the local error-per-unit-step.

In [6] and [7] the defects corresponding to a class of interpolants associated with an $s$-stage, $p$th order Runge-Kutta formula were investigated. It was shown that for any locally $O(h^{p+1})$ interpolant, $\hat{p}_n(x_n + \tau h)$, derived by adding extra stages to the underlying Runge-Kutta formula, the local error in the interpolant satisfies

$$\hat{p}_n(x_n + \tau h) - u_n(x_n + \tau h) = h^{p+1} \sum_{j=1}^{\lambda_p} q_j(\tau) F_j + O(h^{p+2}), \tag{4.1}$$

where the $q_j(\tau)$ are polynomials in $\tau$, of degree at most $p$, depending only on the coefficients of the underlying formula and the extra stages. From (4.1) it can be shown that the associated defect satisfies

$$\hat{\delta}_n(x_n + \tau h) = h^p \sum_{j=1}^{\lambda_p} q_j'(\tau) F_j + O(h^{p+1}). \tag{4.2}$$

The emphasis in [6] was to use interpolants, $\hat{p}_n(x_n + \tau h)$, which required as few total stages as possible. Specific interpolants were analysed for $4 \le p \le 7$, and, by considering the $q_j'(\tau)$ for $j = 1, 2, \ldots, \lambda_p$, an appropriate sample point $\hat{\tau}^*$ for determining an approximation

$$\|\hat{\delta}(x_n + \hat{\tau}^* h)\| \simeq \max_{[0,1]} \|\hat{\delta}(x_n + \tau h)\|, \tag{4.3}$$

was identified. Since the $F_j$ in (4.2) are problem-dependent it is impossible to guarantee that a pre-chosen sample point $\hat{\tau}^*$ will give a good approximation (indeed,

in the worst case, cancellation in $\sum_{j=1}^{s} q_j'(\hat{\tau}^*)F_j$ can make the defect sample arbitrarily small). Nevertheless the numerical results in [6] show that the resulting technique performs surprisingly well. This type of defect error control will be denoted RDEF (for "relaxed" defect control). We point out that the defect in the PDEF algorithms has the form (4.2), but in this case all polynomials $q_j'(\tau)$ are the same, up to a constant factor, and hence there is no chance of misleading cancellation in the sum.

The second defect control technique we will consider is actually a new interpretation of a standard error control technique. Consider an error-per-unit-step implementation of an order $(p, p+1)$, $s$-stage, explicit Runge-Kutta formula pair where the lower order formula is used to advance the solution. The higher order formula determines an approximation $\tilde{y}_{n+1}$ that is used only to obtain an asymptotically correct estimate of the local error per unit step;

$$\frac{y_{n+1} - \tilde{y}_{n+1}}{h} \simeq \frac{\text{le}_{n+1}}{h}. \tag{4.4}$$

In [7] it is observed that this technique can be interpreted as a defect control strategy for an interpolant $\tilde{p}_n(x_n + \tau h)$ that can be defined in terms of a locally $O(h^{p+2})$ interpolant, $q_n(x_n + \tau h)$, as follows

$$\tilde{p}_n(x_n + \tau h) := q_n(x_n + \tau h) - \tau(\tilde{y}_{n+1} - y_{n+1}). \tag{4.5}$$

Note that $\tilde{p}_n(x_n + \tau h)$ interpolates $y_n$, $f(x_n, y_n)$ and $y_{n+1}$ and the resulting piecewise polynomial will be in $C^0[a, b]$ but not in $C^1[a, b]$. It follows from (4.5) that the associated defect satisfies

$$\tilde{\delta}_n(x_n + \tau h) = \frac{y_{n+1} - \tilde{y}_{n+1}}{h} + O(h^{p+1}). \tag{4.6}$$

Clearly $\|y_{n+1} - \tilde{y}_{n+1}\|/h$ provides an asymptotically correct estimate of the maximum defect norm, although forming the corresponding interpolant would be relatively expensive since it is defined in terms of an $O(h^{p+2})$ interpolant, whereas the discrete solution is only $p$th order. We will refer to this defect control technique as SDEF (for strict defect control).

In Table 2 we tabulate the cost per step of these three defect control schemes. The cost is measured in terms of the number of derivative evaluations per step required to implement the technique for a given order. For PDEF and RDEF this includes the cost required to construct the associated interpolant. For SDEF one can estimate the defect without forming the interpolant so we report only the cost of forming $y_{n+1}$ and $\tilde{y}_{n+1}$ (although we also report in parentheses the extra cost of forming the interpolant, $\tilde{p}_n(x_n + \tau h)$). For PDEF the cost reported is the "parallel cost" – a collection of $f$ evaluations that can be performed in parallel are counted only once. This requires the existence of $k - 1$ processors where $k > (p+1)/2$. For an $s$-stage, $p$th order Runge-Kutta formula this parallel cost is always $s + 1$. (We

assume here that a "sequential" Runge-Kutta formula is used; if more processors are available then the cost could be reduced to $p + 1$, see for example [18].)

Table 2. *Minimum cost per step for defect control techniques.*

| order | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|
| RDEF | 6 | 9 | 11 | 15 | 20 |
| SDEF | 6 (2) | 8 (3) | 10 (5) | 13 (7) | 16 (9) |
| PDEF | 5 | 7 | 8 | 10 | 12 |

Table 3. *Limiting speedup factors for PDEF.*

| order | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|
| $R_{RD}$ | 1.20 | 1.29 | 1.38 | 1.50 | 1.67 |
| $R_{SD}$ | 1.60 | 1.57 | 1.88 | 2.00 | 2.08 |
| $R_{PD}$ | 1.80 | 2.71 | 2.75 | 3.70 | 3.75 |
| breakeven | 83% | 78% | 73% | 67% | 60% |

Note that the reported costs are the minimum known values for a particular order $p$. Since the PDEF technique applies to all Runge-Kutta formulas the associated cost is one more than the minimum number of stages required to obtain order $p$. For RDEF and SDEF, interpolants have been extensively investigated only for restricted classes of formulas (or formula pairs). The values reported for SDEF are based on using a standard formula pair to determine $y_{n+1}$ and $\tilde{y}_{n+1}$ and therefore the cost associated with order $p$ is the lowest known cost for an order $(p, p+1)$ pair. For RDEF schemes, the tabulated costs for orders 7 and 8 reflect recent work of Verner [30].

To quantify the performance of parallel numerical methods it is customary to define and measure a "speedup" factor. One such measure is the ratio of execution time of a single processor implementation of PDEF to that of a multiprocessor implementation of PDEF. We will denote such a measure $R_{PD}$. A second more natural measure is the ratio of execution time of a single processor implementation of the "best" available competing sequential technique, in this case either RDEF or SDEF, to that of a multiprocessor implementation of PDEF. We will denote the corresponding measures $R_{RD}$ and $R_{SD}$ respectively.

If we assume that the execution time for each step is dominated by the time associated with derivative evaluations we can predict the expected speedups for each of the above measures from the values in Table 2. For $k = \lceil p/2 + 1 \rceil$ and $(k-1)$ processors the predicted value for $R_{PD}$ is $(s(k-1)+1)/(s+1)$ while the predicted values of $R_{RD}$ and $R_{SD}$ are just the ratios of the corresponding costs per step. These ratios are reported in Table 3. In determining these measures of speedup we are

assuming that the interpolant is required on each step and therefore the cost associated with SDEF is the cost of the basic defect control scheme plus the cost of forming the interpolant.

From the values of $R_{RD}$ in Table 3 we observe that in addition to the improved reliability that PDEF was designed to provide, significant improvements in efficiency can also be realised. Since it is perhaps unrealistic to expect perfect parallelism in the algorithm, we also give the "breakeven" percentage; that is, the percentage of the theoretically attainable parallel speedup needed for PDEF to be at least as efficient as RDEF. The numerical results of the next section are intended to verify that the expected improvements in reliability and efficiency can be achieved in practice.

Table 4. *Percentage of optimal speedup on wave problem.*

| $K1$ | 1 | 3 | 5 | 7 |
|---|---|---|---|---|
| $k = 4, p = 6$ | 85.3% | 94.6% | 96.7% | 96.6% |
| $k = 5, p = 8$ | 90.4% | 93.8% | 94.4% | 96.0% |

## 5. Numerical tests.

We implemented the PDEF algorithm with $k = 4, p = 6$ and $k = 5, p = 8$. For the underlying Runge-Kutta formulas we used a seven-stage, sixth order formula and a twelve-stage eighth order formula identified by Sharp and Smart [24]. These formulas were derived as the higher order members of embedded pairs, and we believe that they are among the most efficient of their type. (However, as mentioned earlier, special parallel Runge-Kutta formulas could be used to improve efficiency on suitable architectures.) We used the $\{\sigma_i\}_{i=2}^{k-1}$ and $\tau^*$ values of Table 1. The infinity norm of the defect was controlled, and the stepsize was varied according to standard asymptotically-based criteria (see [6] or [14]). A comprehensive collection of test results can be found in [10]. Here, due to space limitations, we present a summary of these results.

Our PDEF algorithm was designed under the assumption that $f$ evaluations dominate the overall cost of the integration. Since the algorithm is quite coarse-grained (no communication between processors is required until each Runge-Kutta step has been completed) and well load balanced (each Runge-Kutta step takes approximately the same amount of time to complete), a near optimal level of parallelism should be attainable when $f$ is expensive to evaluate. We are therefore concerned with the two questions

- How close can we come to the optimal $R_{PD}$ speedup of section 4 when $f$ is expensive?
- Can we achieve the "breakeven" level of parallelism when $f$ is inexpensive?

We implemented the algorithm on a Silicon Graphics Inc. Power Iris 4D/240S-64 machine with a 4-processor shared memory architecture. The parallelism in the algorithm takes place in one loop of the form.

*do* $i = 2, k$

$\{$*Advance from* $(x_n, y_n)$ *over a step of length* $\sigma_i h$ *to produce* $\Phi_i$ *and* $f_{n+\sigma_i}$.$\}$

On the SGI machine this is easily handled in Fortran using the C$DOACROSS compiler directive. If compiled normally the directive is ignored (treated as a comment statement), while compilation with the "-mp" option allows the compiler to generate code that parallelizes the do loop. Hence a single Fortran program can be used to study the effect of parallelism on the execution time for the algorithm.

The algorithm was tested on a perturbed nonlinear wave group problem given by equation (4.2) of [1], with parameters $K0 = 40$, $K1 = 3$, $\varepsilon = .025$, $\alpha = .251$, integrated over the range $[0, 500]$. In this case the cost of evaluating $f$ depends on the parameter $K1$, which corresponds to the width of the waveband being computed. The cost is dominated by the term $[2K1(2K1 + 1)(4K1 + 1)/3 + (2K1 + 1)^2]M$, where $M$ is the cost of three complex multiplications plus one complex conjugation. Results for $K1 = 1, 3, 5, 7$ are given in Table 4. (For these tests, in order to reduce the overall cpu time used, we restricted the range of integration for the larger $K1$ values.) We see that the speedups are always reasonable and become close to optimal as the expense increases. Other tests confirmed that even when $f$ involves very few floating point operations the algorithm exceeds the breakeven level, and for more expensive functions the level exceeds 98% of its limiting value.

Further tests were performed to compare the reliability and efficiency of the PDEF algorithm with that of the corresponding relaxed version, RDEF, mentioned in section 4. The testing made use of a modified version of the nonstiff DETEST package [9]. The modifications were designed to allow a more comprehensive assessment of methods based on defect control. The detailed results for the twenty-five test problems, which can be found in [10], were in agreement with the analysis of the previous sections.

## 6. Conclusions and extensions.

Our analysis and numerical investigations have shown that small-scale parallelism can be used to improve both the efficiency and reliability of initial value methods based on explicit Runge-Kutta formulas. We are also hopeful that ideas similar to those introduced here could be successfully applied to multistep methods and to methods for stiff differential equations.

As mentioned earlier our approach also provides a $(p + 1)$st order approximation to the local solution that could be used as the basis of a (completely parallel) global error estimation strategy. We plan to pursue this idea in a separate investigation.

Another natural extension that uses two extra processors is outlined in [10]. Here the aim is to advance part of the way to $x_n + h$ when $\|\delta_n(x_n + \tau^* h)\|$ is too large, and similarly to advance beyond $x_n + h$ (with no extra computation) when $\|\delta_n(x_n + \tau^* h)\|$ is much smaller than TOL. The resulting scheme is likely to be more efficient since there would be fewer rejected steps and the average stepsize should be larger. Also the "tolerance proportionality" may be improved, since rather than simply aiming for a defect that is less than TOL on each step, an attempt is made to keep the defect close to TOL in norm.

Finally we emphasize that given any Runge-Kutta formula the error control technique presented here only increases the number of sequential stages by one. It therefore has an efficiency that is at least comparable with any other parallel explicit Runge-Kutta error control technique. Moreover, only a single formula is required, rather than an embedded pair, and a reliable interpolant is produced. For these reasons we believe that robust defect control is a natural choice when explicit Runge-Kutta formulas are to be implemented in a small-scale parallel environment.

**Acknowledgements.**

We thank Philip Sharp for providing a Fortran specification of the wave problem used in section 5 and Nick Higham for commenting on this manuscript.

REFERENCES

[1] Bryant, P. J.: *Nonlinear wave groups in deep water*, Studies in Applied Mathematics (1979), 1–30.
[2] Burrage, K.: *Solving nonstiff IVPs in a transputer environment*, Manuscript (1990), CMSR, University of Liverpool, U.K.
[3] Cash, J. R.: *A block 6(4) Runge-Kutta formula for nonstiff initial value problems*, ACM Trans. Math. Soft. 15 (1989), 15–28.
[4] Davis, P. J.: *Interpolation and Approximation*, Dover, New York, 1975.
[5] Dormand, J. R. and Prince, P. J.: *Runge-Kutta triples*, Comp. and Maths. with Appls. 12 (1986), 1007–1017.
[6] Enright, W. H.: *A new error-control for initial value solvers*, Applied Maths. and Computation 31 (1989), 288–301.
[7] Enright, W. H.: *Analysis of error control strategies for continuous Runge-Kutta methods*, SIAM J. Numer. Anal. 26 (1989), 588–599.
[8] Enright, W. H., Jackson, K. R., Nørsett, S. P. and Thomsen, P. G.: *Interpolants for Runge-Kutta formulas*, ACM Trans. Math. Soft. 12 (1986), 193–218.
[9] Enright, W. H. and Pryce, J. D.: *Two FORTRAN packages for assessing initial value methods*, ACM Trans. Math. Soft. 13 (1987), 1–27.
[10] Enright, W. H. and Higham, D. J.: *Parallel defect control*, Technical Report No. 237/90, Department of Computer Science, University of Toronto, Canada (1990).
[11] Gladwell, I.: *Initial value routines in the NAG library*, ACM Trans. Math. Soft. 5 (1979), 386–400.
[12] Gladwell, I., Shampine, L. F., Baca, L. S. and Brankin, R. W.: *Practical aspects of interpolation in Runge-Kutta codes*, SIAM J. Sci. Stat. Comput. 8 (1987), 322–341.
[13] Hairer, E., Nørsett, S. P. and Wanner, G.: *Solving Ordinary Differential Equations I*, Springer-Verlag, Berlin, 1987.

[14] Higham, D. J.: *Robust defect control with Runge-Kutta schemes*, SIAM J. Numer. Anal. 26 (1989), 1175–1183.
[15] Higham, D. J.: *Highly continuous Runge-Kutta interpolants*, Technical Report No. 220/89, Department of Computer Science, University of Toronto, Canada (1989); to appear in ACM Trans. Math. Softw.
[16] Higham, D. J.: *Runge-Kutta defect control using Hermite-Birkhoff interpolation*, Technical Report No. 221/89, Department of Computer Science, University of Toronto (1989); to appear in SIAM J. Sci. Stat. Comput.
[17] Higham, D. J.: *Global error versus tolerance for explicit Runge-Kutta methods*, Technical Report No. 233/90, Department of Computer Science, University of Toronto, Canada (1990); to appear in IMA J. Numer. Anal.
[18] Houwen, P. J. van der and Sommeijer, B. P.: *Variable step iteration of high-order Runge-Kutta methods on parallel computers*, Report NM-R8817, Centre for Mathematics and Computer science, Amsterdam, 1988.
[19] Houwen, P. J. van der and Sommeijer, B.P.: *Block Runge-Kutta methods on parallel computers*, Report NM-R8906, Centre for Mathematics and Computer science, Amsterdam, 1989.
[20] Jackson, K. R. and Nørsett, S. P.: *The potential for parallelism in Runge-Kutta methods.* Part 1: *RK formulas in standard form*, Technical Report No. 239/90, Department of Computer Science, University of Toronto, Canada (1990).
[21] Owren, B. and Zennaro, M.: *Derivation of efficient continuous explicit Runge-Kutta methods*, Technical Report No. 240/90, Department of Computer Science, University of Toronto, Canada (1990).
[22] Shampine, L. F.: *Interpolation for Runge-Kutta methods*, SIAM J. Numer. Anal. 22 (1985), 1014–1027.
[23] Shampine, L. F.: *Some practical Runge-Kutta formulas*, Math. Comp. 46 (1986), 135–150.
[24] Sharp, P. W. and Smart, E.: Private communication, (1990).
[25] Skeel, R. D. and Tam, H.-W.: *Potential for parallelism in explicit linear methods*, Working Document, Department of Computer Science, University of Illinois at Urbana-Champaign (1989).
[26] Stetter, H. J.: *Considerations concerning a theory for ODE-solvers*, in *Numerical Treatment of Differential Equations*: Proc. Oberwolfach, 1976 (Eds. R. Bulirsch, R. D. Grigorieff and J. Schröder), Lecture Notes in Mathematics 631, Springer, Berlin, (1978) 188–200.
[27] Stetter, H. J.: *Global error estimation in Adams PC-codes*, ACM Trans. Math. Soft. 5 (1979), 415–430.
[28] Stewart, N. F.: *Certain equivalent requirements of approximate solutions of* $x' = f(t, x)$, SIAM J. Numer. Anal. 7 (1970), 256–270.
[29] Tam, H.-W.: *Parallel methods for the numerical solution of ordinary differential equations*, Report No. UIUCDCS-R-89-1516, Department of Computer Science, University of Illinois at Urbana-Champaign (1989).
[30] Verner, J. H.: *On the derivation of higher order interpolants*, Seminar presented at "The 1990 Conference on the Numerical Solution of ODEs", Helsinki, Finland, June 1990.