Computers Math. Applic. Vol. 23, No. 10, pp. 65–72, 1992
Printed in Great Britain.

0097-4943/92 $5.00 + 0.00

SOLUTION OF ORDINARY DIFFERENTIAL INITIAL VALUE PROBLEMS ON AN INTEL HYPERCUBE

L. LUSTMAN, B. NETA, W. GRAGG
Naval Postgraduate School, Department of Mathematics
Monterey, CA 93943, U.S.A.

(Received January 1991)

Abstract—Schemes for the solution of *linear* initial or boundary value problems on a hypercube were developed by Katti and Neta [1] and tested and improved by Lustman, Neta and Katti [2]. Among other procedures for parallel computers, fully implicit Runge-Kutta methods were discussed by Jackson and Norsett [3] and Lie [4].

Here, we develop a method based on extrapolation to the limit, which is useful even for nonlinear problems. Numerical experiments show excellent accuracy when low order schemes are combined with polynomial extrapolation.

1. INTRODUCTION

In this paper, we discuss the numerical solution, on a parallel computer, of a system of first order ordinary differential equations with initial data. Fully implicit Runge-Kutta methods were discussed by Jackson and Norsett [3] and Lie [4]. Lie assumes that each processor of the parallel computer has vector capabilities. Katti and Neta [1] have developed schemes for the solution of *linear* initial value and boundary problems. These schemes were tested and improved by Lustman *et al.* [1].

Here, we consider the solution of Initial Value Problems (linear or not), based on extrapolation to the limit. The system is solved independently by each processor, using different step sizes, then the results are combined by extrapolation to obtain higher accuracy.

In the next section, we describe the ordinary differential equation solution schemes and the extrapolation procedure. Section 3 will detail the parallel algorithm. Numerical experiments are summarized in the last section.

2. ODE-INTEGRATION AND EXTRAPOLATION

There are numerous schemes for the solution of first order IVPs:

$$\begin{aligned} y'(x) &= f(x, y(x)), \\ y(a) &= y_a, \end{aligned} \tag{1}$$

where y and f are vector valued functions and y_a is a vector of initial values. See Fatunla [5], Lambert [6], Gear [7], Shampine and Gordon [8] and others for numerical methods for ordinary differential equations. See also Deuflhard [9] for review of extrapolation methods.

We demonstrate our idea with two schemes of low order, one of which possesses an asymptotic local error expansion in even powers of h (the mesh size). The following are used:

1. Euler's method:

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{2}$$

This research was conducted for the Office of Naval Research and was funded by the Naval Postgraduate School.

2. The modified midpoint rule, due to Gragg [10,11], in the form:

$$\begin{aligned} z_{\frac{1}{2}} &= y_0 + \frac{h}{2} f(x_0, y_0), \\ y_1 &= y_0 + h f(x_{\frac{1}{2}}, z_{\frac{1}{2}}), \\ z_{n+\frac{1}{2}} &= z_{n-\frac{1}{2}} + h f(x_n, y_n), \qquad n = 1, 2, \ldots, \\ y_{n+1} &= y_n + h f(x_{n+\frac{1}{2}}, z_{n+\frac{1}{2}}). \end{aligned} \tag{3}$$

Euler's scheme is explicit, first order, and its local truncation error is given by:

$$y_n - y(nh) = A_1 h + A_2 h^2 + A_3 h^3 + \cdots . \tag{4}$$

It can be shown that each stage of extrapolation will lead to one order of h increase in accuracy. Gragg's scheme is explicit, second order, and its local truncation error is given by:

$$y_n - y(nh) = B_2 h^2 + B_4 h^4 + \cdots . \tag{5}$$

Thus, each extrapolation stage leads to a gain in accuracy of 2 orders in h. There are also implicit schemes with local truncation error containing only even powers, but their use with extrapolation is discouraged (Fatunla [5]), as these expansions are valid only if fully converged solutions are obtained at each integration step. Thus, Gragg's method, which has the advantage of being explicit, is the scheme we have decided to employ.

Polynomial and rational extrapolation will be used. Suppose p processors have generated the values $y(x_i)$ using various steps h_r. We consider the set of points:

$$\{h_r, y(x_i, h_r) | r = 0, 1, \ldots, p-1; \quad i = 1, 2, \ldots, M\}, \tag{6}$$

where x_i is the mesh point at which all the processors have obtained a numerical solution to our problem. At each point x_i, it is possible to define a polynomial $\Pi_{p-1}(h)$ of degree $p-1$ or a rational function $R_{\mu,\nu}(h)$ (where μ is the degree of the numerator, and ν is the degree of the denominator), which satisfy:

$$\Pi_{p-1}(h_r) = R_{\mu,\nu}(h_r) = y(x_i, h_r). \tag{7}$$

Aitken [12] and Neville [13] each independently proposed a scheme whereby the polynomial is generated recursively. A table is constructed having as first column

$$T_{r0} = y(x_i, h_r).$$

The s^{th} column, consisting of entries T_{rs} is computed by:

$$T_{rs} = T_{r+1,s-1} + \frac{T_{r+1,s-1} - T_{r,s-1}}{\left(\frac{h_r}{h_{r+s}}\right)^{\gamma} - 1}, \qquad s = 1, 2, \ldots, p-1; \quad r = 0, 1, \ldots, p-s. \tag{8}$$

Here, $\gamma = 2$ if T_{r0} has an asymptotic error expansion in powers of h^2, and $\gamma = 1$ otherwise.

"Experience has shown that extrapolation based on rational functions is superior to polynomial extrapolation particularly in the neighborhood of a singularity" (Fatunla [5]). Bulirsch and Stoer [14] adopted the rational function

$$R_{\mu,\nu}(h) = \frac{\sum_{j=0}^{\mu} \alpha_j h^{\gamma j}}{\sum_{j=0}^{\nu} \beta_j h^{\gamma j}}, \tag{9}$$

where

$$\mu = \left[\frac{p-1}{2}\right], \tag{10}$$

and

$$\nu = p - 1 - \mu. \tag{11}$$

The formula used to construct the table is then

$$T_{r,-1} = 0, \tag{12}$$

$$T_{r0} = y(x_i, h_r), \quad r = 0, 1, \ldots, p-1, \tag{13}$$

$$T_{rs} = T_{r+1,s-1} + \frac{T_{r+1,s-1} - T_{r,s-1}}{\left(\frac{h_r}{h_{r+s}}\right)^{\gamma}\left[1 - \frac{T_{r+1,s-1}-T_{r,s-1}}{T_{r+1,s-1}-T_{r+1,s-2}}\right] - 1}, \tag{14}$$

$$s = 1, 2, \ldots, p-1, \quad r = 0, 1, \ldots, p-s-1.$$

Wuytack [15,16] constructed T_{rs} in a more efficient way (fewer arithmetic operations).

3. PARALLEL PROCEDURE

In this section, we describe the implementation of our idea to solve the system of initial value problems (1) on an INTEL hypercube. Let the r^{th} processor, out of p, solve the same problem (1) using step size h_r, where

$$h_r = \frac{p}{r}H, \quad r = 1, 2, \ldots, p, \tag{15}$$

and any numerical scheme (same one for all processors). Then, clearly, all processors have a solution to some accuracy (depending on the processor) at all points

$$x_j = a + (j-1)pH. \tag{16}$$

This choice of h_r is taken to have an almost balanced load. These values are extrapolated (by either a polynomial or a rational function) to obtain the solution at those points to much higher accuracy. For example, if Euler's scheme is used, one can get a solution at these points x_j to $O(h^p)$. Gragg's method will yield a solution accurate to $O(h^{2p})$.

Note that the first processor is using the largest step size and therefore, it will finish first. The second one will finish soon after. As soon as the second is done, the first can start computing the elements in the first column of the extrapolation table. The second processor works on the next column as soon as the first two entries in the previous column are ready. In Figure 1, we indicate which processor works on which column.

	Computed by Processor	Step Size	Processor 1	2	3	4	5	6	7
T_{00}	1	$8H$							
			T_{10}						
T_{01}	2	$4H$		T_{20}					
			T_{11}		T_{30}				
T_{02}	3	$\frac{8}{3}H$		T_{21}		T_{40}			
			T_{12}		T_{31}		T_{50}		
T_{03}	4	$2H$		T_{22}		T_{41}		T_{60}	
			T_{13}		T_{32}		T_{51}		T_{70}
T_{04}	5	$\frac{8}{5}H$		T_{23}		T_{42}		T_{61}	
			T_{14}		T_{33}		T_{52}		
T_{05}	6	$\frac{4}{3}H$		T_{24}		T_{43}			
			T_{15}		T_{34}				
T_{06}	7	$\frac{8}{7}H$		T_{25}					
			T_{16}						
T_{07}	8	H							

Figure 1.

Note that in order to minimize communication, we let processor i be the i^{th} processor as given by Gray code.

4. NUMERICAL EXPERIMENTS

In this section, we list some of the systems solved and compare the accuracy obtained by Euler's and Gragg's methods when combined with either polynomial or rational extrapolation. In each case, we used p processors, $p = 2, 3, \ldots, 8$.

The first example is

$$y' = y \sin x, \quad 0 < x < 5, \tag{17}$$

$$y(0) = e^{-1}. \tag{18}$$

The exact solution is

$$y_e(x) = e^{-\cos x}. \tag{19}$$

In Table 1, we summarize the results of our experiment with Euler's method and extrapolation using p processors. It is clear that the results using polynomial extrapolation are much better. The accuracy ($||y_h - y_e||_0$, where y_h is the result after extrapolation) is increasing with the number of processors. The results using rational extrapolation are not as good and not improving after using 5 processors.

To measure the order of the method, we have computed the solution y_h and $y_{h/2}$ (the solution after extrapolation with step $h/2$ instead of h). The columns entitled 'error reduction' in each table report the error quotients 'coarse/fine.' In Table 1, this quantity was always below the theoretical value of 2^p. With rational extrapolation, the results are disappointing.

Table 1. Euler's scheme.

	POLYNOMIAL			RATIONAL		
Processor	coarse (1/4)	fine (1/8)	error reduction	coarse (1/4)	fine (1/8)	error reduction
2	1.55-02	4.54-03	3	5.15-03	1.26-03	4
3	1.17-03	1.78-04	7	3.52-04	4.03-05	9
4	6.71-05	5.23-06	13	9.04-04	4.23-05	21
5	3.05-06	1.22-07	25	1.11-04	2.11-06	53
6	1.19-07	2.42-09	49	6.38-06	1.05-06	6
7	3.92-09	4.06-11	97	1.26-05	5.67-05	0
8	1.14-10	5.01-13	228	2.11-04	5.29-05	4

The results using Gragg's method are given in Table 2.

Table 2. Gragg's method.

	POLYNOMIAL			RATIONAL		
Processor	coarse (1/4)	fine (1/8)	error reduction	coarse (1/4)	fine (1/8)	error reduction
2	3.06-06	1.88-07	16	6.72-06	4.16-07	16
3	7.72-09	1.17-10	66	9.99-09	1.54-10	65
4	1.66-11	6.20-14	268	2.06-10	2.87-12	72
5	3.01-14	3.99-15	8	1.59-10	2.18-13	729
6	4.73-15	2.92-15	2	6.04-12	1.08-13	56
7	4.81-15	1.20-14	0	4.37-13	1.44-14	30
8	3.36-14	2.98-14	0	1.22-08	7.60-10	16

Note that the accuracy using Gragg's method is higher. Machine accuracy (double precision used) has been reached with five processors if polynomial extrapolation is used, but seven processors are required when rational extrapolation is employed. The error quotient is close to

the theoretical one (4^p) until machine accuracy is reached. Again, rational extrapolation was disappointing.

The second example is a system of N equations

$$y_j' = \frac{j\, y_j\, y_{j+1}}{x^{j+2}}, \quad j = 1, 2, \ldots, N-1 \quad 6 < x < 10,$$
$$y_N' = \frac{N\, y_N\, y_1}{x^2}, \tag{20}$$

subject to the initial condition:

$$y_j(6) = 6^j, \quad j = 1, 2, \ldots, N. \tag{21}$$

The exact solution is

$$y_j = x^j \quad j = 1, 2, \ldots, N. \tag{22}$$

The results are summarized in Tables 3 (Euler) and 4 (Gragg) for $N = 4$.

Table 3. Euler's scheme.

	POLYNOMIAL			RATIONAL		
Processor	coarse (2)	fine (1)	error reduction	coarse (2)	fine (1)	error reduction
2	1.33-01	5.70-02	2	6.95-02	2.56-02	3
3	3.89-02	1.07-02	4	1.03-01	2.97-03	35
4	1.06-02	1.88-03	6	2.69-03	3.54-04	8
5	2.78-03	3.02-04	9	1.39-03	4.61-04	3
6	6.82-04	4.40-05	16	3.99-03	1.69-05	236
7	1.55-04	5.79-06	27	8.73-05	1.34-05	7
8	3.24-05	6.94-07	47	5.35-03	1.54-03	3

Table 4. Gragg's method.

	POLYNOMIAL			RATIONAL		
Processor	coarse (2)	fine (1)	error reduction	coarse (2)	fine (1)	error reduction
2	7.51-03	7.18-04	10	5.00-03	4.79-04	10
3	4.56-04	1.27-05	36	1.39-04	1.12-06	124
4	2.06-05	1.59-07	130	8.22-06	1.40-07	59
5	6.89-07	1.45-09	475	7.29-07	1.25-08	58
6	1.80-08	1.03-11	1748	3.61-06	1.25-08	289
7	3.85-10	5.38-14	7156	8.62-09	1.60-10	54
8	7.10-12	2.55-14	278	1.15-05	7.35-07	16

The same conclusions can be reached for this system.

The next example is from orbital mechanics.

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -\frac{y_1}{r^3}, \\ y_3' &= y_4, \\ y_4' &= -\frac{y_3}{r^3}, \qquad 0 < x < 4, \end{aligned} \tag{23}$$

$$y(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{24}$$

where $r^2 = y_1^2 + y_3^2$.

The exact solution is

$$y_e(x) = \begin{bmatrix} \cos x \\ -\sin x \\ \sin x \\ \cos x \end{bmatrix}. \tag{25}$$

The results are given in Tables 5 (Euler) and 6 (Gragg). The accuracy of Euler's method is much lower, and extrapolation didn't improve the results much.

Table 5. Euler's scheme.

	POLYNOMIAL			RATIONAL		
Processor	coarse (.4)	fine (.2)	error reduction	coarse (.4)	fine (.2)	error reduction
2	5.48-01	3.31-01	2	8.96-01	7.80-01	1
3	3.06-01	1.42-01	2	2.56+00	5.58-01	5
4	1.64-01	5.68-02	3	4.13+00	5.80-01	7
5	8.45-02	2.09-02	4	2.67+00	1.15+01	0
6	4.13-02	7.12-03	6	5.30-01	3.72-01	1
7	1.91-02	2.23-03	9	5.17-01	7.75-01	1
8	8.38-03	6.50-04	13	4.00-01	2.72+00	0

Table 6. Gragg's method.

	POLYNOMIAL			RATIONAL		
Processor	coarse (.4)	fine (.2)	error reduction	coarse (.4)	fine (.2)	error reduction
2	4.14-03	2.95-04	14	2.02-03	2.43-02	1
3	7.44-05	1.38-06	54	1.57+00	1.73-04	9075
4	8.52-07	4.04-09	211	3.31-03	1.67-05	198
5	6.72-09	8.12-12	828	2.96-04	1.36-06	218
6	3.95-11	2.20-14	1750	1.93-05	8.96-08	215
7	1.80-13	8.43-14	2	1.79-06	1.51-08	119
8	1.59-13	1.50-13	1	5.87-05	3.40-06	17

We conclude with a linear system

$$y' = Ay, \quad 0 < x < 4,$$

$$y(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{26}$$

where $A = (a_{ij})$ is a symmetric tridiagonal matrix whose elements are

$$a_{ii} = -2, \quad i = 1, 2, \ldots, N, \tag{27}$$

$$a_{i,i+1} = a_{i+1,i} = 1, \quad i = 1, 2, \ldots, N-1.$$

This system results from approximating the one-dimensional heat equation

$$u_t = u_{xx}. \tag{28}$$

The results are summarized in Table 7. The error quotient is much better than 2^p when using polynomial extrapolation. The accuracy, though, is not very high, and in linear systems, it is cheaper and more accurate to use the idea developed by Lustman *et al.* [2].

Table 7. Euler's scheme.

	POLYNOMIAL			RATIONAL		
Processor	coarse (1/2)	fine (1/4)	error reduction	coarse (1/2)	fine (1/4)	error reduction
2	7.88-01	2.16-02	36	8.21-01	5.78-02	14
3	3.75-01	2.98-03	126	3.02-02	4.98-03	6
4	1.27-01	6.49-04	196	5.86-02	3.40-03	17
5	3.40-02	1.18-04	288	2.02-02	3.01-04	67
6	7.82-03	1.69-05	463	5.36-03	5.89-04	9
7	1.62-03	2.01-06	806	1.64-03	2.27-05	72
8	3.11-04	2.07-07	1502	5.06-03	1.10-03	5

To show the benefit of this parallel algorithm, we have measured the speedup defined by

$$S = \frac{T_p(1)}{T_p(p)}, \tag{29}$$

where $T_p(i)$ is the execution time required when using i processors. This is the most common formulation of speedup.

As defined, the speedup should ideally be directly proportional to p, the number of processors used. The efficiency, defined as

$$E = \frac{S}{p}, \tag{30}$$

provides a quantitative measure of how closely the observed speedup approaches the ideal result. We have measured the execution time required for Gragg's method with polynomial extrapolation to solve (20)–(22) on an eight processor hypercube versus the time required by one processor to execute the same task, using $h = .1$. We found that

$$T_p(1) = 759,$$

$$T_p(8) = 152; \tag{31}$$

thus,

$$S = 4.96,$$

$$E = .62. \tag{32}$$

When the amount of work is increased (by taking $h = .05$), the speedup and efficiency are

$$S = \frac{1509}{247} = 6.08,$$

$$E = .76\ . \tag{33}$$

It is clear that our previous algorithm for linear systems (Lustman *et al.*, [2]) is more efficient and should be used when solving linear problems.

References

1. C.P. Katti and B. Neta, Solution of linear initial value problems on a hypercube, Technical Report NPS-53-89-001, Naval Postgraduate School, Monterey, CA, pp. 1–7, (1988).
2. L. Lustman, B. Neta and C.P. Katti, Solution of linear systems of ordinary differential equations on an INTEL hypercube, *SIAM Sci. Statis. Comp.* **12**, 1480–1485 (1991).
3. K.K. Jackson and S.P. Norsett, Parallel Runge-Kutta methods, (unpublished report).
4. I. Lie, Some Aspects of Parallel Runge-Kutta Methods, Institutt for Numerisk Mathematikk, Universitetet i Trondheim, Norway, pp. 1–39, (1987).
5. S.O. Fatunla, *Numerical Methods for Initial Value Problems in Ordinary Differential Equations*, Academic Press, Boston, (1988).

6. J.D. Lambert, *Computational Methods in Ordinary Differential Equations*, J. Wiley, New York, (1973).
7. C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs, NJ, (1971).
8. L.F. Shampine and M.K. Gordon, *Computer Solution of Ordinary Differential Equations—The Initial Value Problem*, W.H. Freeman & Co., San Francisco, (1975).
9. P. Deuflhard, Recent progress in extrapolation methods for ordinary differential equations, *SIAM Rev.* **27**, 505–535 (1985).
10. W.B. Gragg, Repeated extrapolation to the limit in the numerical solution of ordinary differential equations, Ph.D. Dissertation, UCLA, Los Angeles (1964).
11. W.B. Gragg, On extrapolation algorithms for ordinary initial value problems, *SIAM J. Num. Anal.* **2**, 384–403 (1965).
12. A.C. Aitken, On interpolation by iteration of proportional parts, *Proc. Edinburgh Math. Soc.* **2**, 56–76 (1932).
13. E.H. Neville, Iterative interpolation, *J. Indian Math. Soc.* **20**, 87–120 (1934).
14. R. Bulirsch and J. Stoer, Numerical treatment of ordinary differential equations by extrapolation methods, *Numer. Math* **8**, 1–13 (1966).
15. L. Wuytack, A new technique for rational extrapolation to the limit, *Numer. Math.* **17**, 215–221 (1971).
16. L. Wuytack, Extrapolation to the limit by using continued fraction extrapolation, *Rocky Mountain J. Math.* **4**, 395–397 (1974).