数理解析研究所講究録
第 841 巻 1993 年 232-245

Parallel Numerical Methods for Initial-Value Problems of ODEs

三井 斌友
(Taketomo MITSUI)
名古屋大学人間情報学研究科
(*Graduate School of Human Informatics, Nagoya University*)

後藤 彰
(Akira GOTO)
特許庁審査第五部
(*Electronic Image Devices Division, Japanese Patent Office*)

ABSTRACT

We propose a new A-stable 3-stage fourth order implicit Runge-Kutta (RK) formula that can be carried out in parallel. Our parallel algorithm, which may fall in the category of parallelism across stage, can be applied to large-scale linear ordinary differential equations. An actual implementation on parallel computers with 3 CPU's and shared memory shows that it attains 2.9 times speed-up for the equations of $y' = Ly$ over the usual sequential computation. Moreover, through some numerical experiments we introduce a quantity κ_{IRK} which can be seen as an index of degree of accuracy in the solution of the parallelized implicit RK method.

1 Parallelism in Numerical ODEs

Introduction of parallelism to numerical ODEs is a promising as well as a challenging issue. Since the development of computer architecture in various kind of parallelism makes it possible to implement the organizations of computation beyond the traditional sequential one, it should be very natural to think of their application to numerical methods for initial-value problems of ODEs, especially for large-scale ones. Indeed recent research works are focusing on this issue. For instance, a special international workshop was organized by A. BELLEN and M. ZENNARO at Grado, Italy in September, 1991 (its Proceedings will appear in a special issue of the journal *Applied Numerical Methods*).

What kind of parllelism can be carried into numerical ODEs? Of course, it depends on what algorithm would be implemented. For example, a *vector computer*, a computer equipped with vector arithmetic units, is highly applicable for the solution of simultaneous linear equations often ocurring in the stepping of implicit discrete variable methods. The solution of equations could be accelerated by the application of vector computers. Much work has been devoted in this direction.

However, what we are considering below is *not* of this type of parallelism. Rather we are interested in the parallelism proper in the discrete variable formulation for numerical ODEs. Although there is a difference of opinion between researchers, conceptually such parallelism might be classified in the following way.

(i) Parallelism across System

Suppose that the underlying system of ODEs consists of two parts which are weakly connecting with each other. Then one can proceed some steps to integrate each part independently with possibly different methods and stepsizes. This can be organized in parallel. Although such system of ODEs are seemingly popular in the real world, the trial in this direction is rare as far as the present authors are aware.

(ii) Parallelism across Time

This name, together with that of the next category, is motivated by the parallel solution of space-discretized time-dependent PDEs. That is, some or whole numerical values for the discrete points in space are assigned to one processor at the same time-level. Parallelism occurs across different time-levels. Hence it is often called *Parallelism across the Method.*

Typically this type of parallelism is carried out with *predictor-corrector methods.* Probably MIRANKER and LINIGER[10] were the first to exploit the method. Their idea can be explained in the following simple example. For the initial-value problem

$$\frac{dy}{dx} = f(x,y) \quad (a < x < b), \quad y(a) = y_0, \tag{1}$$

usual Adams PC pair of second order is given by

$$y^p_{n+1} = y^c_n + h[3f^c_n - f^c_{n-1}]/2, \quad y^c_{n+1} = y^c_n + h[f^p_{n+1} + f^c_n]/2. \tag{2}$$

Here the superscripts p and c are the indications of the predicted and the corrected values, respectively. Since the computation will proceed in the manner

$$\cdots \to y^p_{n+1} \to f^p_{n+1} \to y^c_{n+1} \to f^c_{n+1} \to \cdots,$$

it can be done only sequentially.

Instead the following pair of formulae enables a parallel calculation for y^c_n and y^p_{n+1}:

$$y^p_{n+1} = y^c_{n-1} + 2hf^p_n, \quad y^c_n = y^c_{n-1} + h[f^p_n + f^c_{n-1}]/2. \tag{3}$$

That is, a Milne-type predictor is introduced in the pair.

(iii) Parallelism across Space

Some discrete variable methods are known to be reformulated in the *block predictor-corrector method* which allows a parallelism. Suppose that Y_{n+1} represents a vector of block of s solution values of y, and that $F(Y_{n+1})$ represents the vector of block of corresponding derivatives. A block PC method could be given by

$$Y^p = A_1 \otimes Y_n + hL_1 \otimes F(Y_n), \; Y^c = A_2 \otimes Y_n + hL_2 \otimes F(Y^p), \tag{4}$$

where A_1, A_2, L_1 and L_2 are all matrices of dimension s. Note that an explicit Runge-Kutta method as well as a linear multistep method can be written in the form. We may therefore mix the integration formulae between the predictor and the corrector phases[2, 11].

In the model of space-discretized PDEs, this parallelism means numerical values for different time-levels on a single space-discrete point are assigned to one processor. Thus the name of *Parallelism across the System* is often given to this category.

(iv) Parallelism across Step

It might cause a confusion of names, but the *waveform relaxation methods*, which now attract much attention of the experts in numerical ODEs, can be said to derive a parallelism across step. As they are also called *Picard-Lindelöf iteration*, their another organization may be considered as a parellelism across methods[5].

(v) Parallelism across Stage

This is particular in the Runge-Kutta methods. Since the main drawbacks in RK lie on the number of function evaluations per step, parallelism across stage may extend a new perspective for RKs. The authors like to put emphasis on this topic and will give some more detailed explanations in the next section.

Since we have a large-scale system in mind and it often exhibits stiffness, the *stability* as well as the *accuracy* of the method should be still main elements to be analyzed.

2 Implicit Runge-Kutta Formula for Parallel Computation

From now on we will be concerned with parallelizable implicit Runge-Kutta methods, for they are easy to implement on a multiprocessor machine with the shared-memory structure. Taking the possible stiffness of ODEs into account, a good stability should be attained by the targeted implicit Runge-Kutta (IRK) methods.

Consider a large-scale linear system of ODEs

$$C(t)\frac{d}{t}\boldsymbol{u} + A(t)\boldsymbol{u} = \boldsymbol{f}. \tag{5}$$

In a particular case, Eq.5 becomes to $\boldsymbol{y}' = L\boldsymbol{y}$, where L is a constant matrix. KARAKASHIAN and RUST[8, 9] took such linear equations in mind and made a parallelizable 2-stage third order IRK method of collocation type. What we propose below is along their guideline to make a new parallelizable A-stable 3-stage fourth order IRK formula, and to define its parallel algorithm for equations with variable coefficients $\boldsymbol{y}' = L(t)\boldsymbol{y} + \boldsymbol{g}(t)$, too.

2.1 Family of 3-Stage Fourth Order IRK Formulae

To solve a system of ODEs in a general form as

$$\frac{d}{dt}\boldsymbol{y} = \boldsymbol{f}(t, \boldsymbol{y}), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0$$

numerically, an implicit Runge-Kutta formula with stepsize h gives the value $y_{n+1} \approx y(t_{n+1})$ by a single step integration from $y_n \approx y(t_n)$ in the following way.

$$y_{n+1} = y_n + h\sum_{l=1}^{p} \beta_l k_l, \qquad k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{p} \alpha_{ij} k_j\right). \tag{6}$$

The matrix and vector notations for the formula parameters

$$A = (\alpha_{ij}), \qquad b = (\beta_l), \qquad c_i = \sum_{j=1}^{p} \alpha_{ij}$$

are conventional.

As can be seen above, Eq.6 is not fit for parallelism in its naïve form, because during the single step integration, every k_i is functionally dependent to all other k_j's. Thus we have restricted ourselves to get a parallel implementation of IRK for the linear ODEs Eq.5 or the constant coefficient case. In the latter case, IRK reduces to

$$y_{n+1} = y_n + h\sum_{\ell=1}^{p} \beta_\ell k_\ell, \qquad k_i = L\left(y_n + h\sum_{j=1}^{p} \alpha_{ij} k_j\right). \tag{7}$$

It is well known that a *collocation type* IRK formula can be identified either by the pair $\{A, b\}$ or by the fractions $\{c_1, c_2, \ldots, c_p\}$ provided all c_i's are distinct. Since we are seeking for a 3-stage fourth order IRK formula of collocation type, the formula parameters should satisfy the 7 equations

$$\begin{cases} S_1 &= \lambda_1 + \lambda_2 + \lambda_3 \\ S_2 &= \lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1 \\ S_3 &= \lambda_1\lambda_2\lambda_3 \end{cases}, \tag{8}$$

$$\begin{cases} S_1 &= \dfrac{c_1 + c_2 + c_3}{3} \\ S_2 &= \dfrac{c_1c_2 + c_2c_3 + c_3c_1}{6} \\ S_3 &= \dfrac{c_1c_2c_3}{6} \end{cases}, \tag{9}$$

$$\frac{1}{24} - \frac{1}{6}S_1 + \frac{1}{2}S_2 - S_3 = 0. \tag{10}$$

Here $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues of the matrix A. Eventually we can derive that the degree of freedom is two for the above system of nonlinear equations with six unknowns c_1, c_2, c_3, $\lambda_1, \lambda_2, \lambda_3$. Therefore λ_1 and c_1 may become to the free parameters.

In this context a straightforward calculation leads to solutions of the equations Eqs.8, 9 and 10 as follows:

$$c_2, c_3 = \frac{H \pm \sqrt{K}}{D}. \tag{11}$$

Here H, K and D are polynomials of c_1 and λ_1 with their total degrees 4, 8 and 3, respectively. Furthermore

$$\lambda_2, \lambda_3 = \frac{U \pm \sqrt{V}}{W}, \tag{12}$$

where U, V and W are again polynomials of c_1 and λ_1 with total degrees 4, 8 and 3, respectively. It should be remarked that the condition $K > 0$ is necessary for the collocation type formula, whereas the parallel implementation within the real number arithmetic imposes the condition $V > 0$.

Once we get the distinct real eigenvalues λ_j, we can take advantage of the similarity transformation of A

$$A = S^{-1}\Lambda S, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3), \quad S = (s_{ij}) \tag{13}$$

for parallel computation.

2.2 *A-Stability Consideration*

As usual in the collocation type IRK, we define the polynomial $N(t)$ by

$$N(t) \equiv \sum_{j=0}^{3} \frac{(-1)^j S_j t^{3-j}}{(3-j)!}, \tag{14}$$

where S_j is the elementary symmetric polynomial of λ_1, λ_2 and λ_3 of j-th degree.

Let $P(z)$ and $Q(z)$ be the following.

$$\begin{aligned} P(z) &\equiv N(0)z^3 + N'(0)z^2 + N''(0)Z + N^{(3)}(0), \\ Q(z) &\equiv N(1)z^3 + N'(1)z^2 + N''(1)Z + N^{(3)}(1). \end{aligned}$$

The stability function $R(z)$ of the collocation type RK formula becomes[1] to $R(z) = Q(z)/P(z)$. Thus we can compute so-called E-polynomial as

$$E(y) \equiv \mid Q(iy) \mid^2 - \mid P(iy) \mid^2,$$

which is a rational function with respect to c_1 and λ_1. We omit its functional form here, but take notice that the denominator turns out to square of a polynomial. If $E(y) \leq 0$ for any y, then our formula is A-stable whenever $\lambda_i > 0$. Since the equation

$$R(\infty) \equiv \lim_{z \to \infty} R(z) = -\frac{(1-c_1)(1-c_2)(1-c_3)}{c_1 c_2 c_3}$$

holds in our case, we are interested in the case of $c_1 = 1$ for L-stability.

2.3 *The Region of Parameters Giving an A-Stable Formula*

We are going to determine the region of parameters (c_1, λ_1) where they satisfy the conditions $K > 0$ (collocation type), $V > 0$ (parallelizability), $\lambda_i > 0$ and $E(y) < 0$ (A-stability). In fact, the conditions $\lambda_i > 0$ are equivalent to $U^2 - V > 0$ and $WU > 0$. On the c_1-λ_1 plane, superposing all these conditions, in Figure 1 we get the desired region of parameters (the hatched region).

Since the straight line $c_1 = 1$ does not have any intersection with the region in Fig. 1, we get the following.

Theorem 1 *Any collocation IRK formula of our type cannot be L-stable.*

Moreover, in Fig. 1 we observe that the condition for A-stability isn't compatible with those of $V > 0$ for $0 < c_1 < 1$, in short there are no parallelizable A-stable formulae for these c_1.

However the region in Fig. 1 implies much possibility to select the pair (c_1, λ_1). We will fix them through numerical investigation in the following section.

3 Parallel Algorithm for the Constant Coefficient Case

At first our parallel algorithm is applied to solve a linear system of ODEs with the constant coefficient case $y' = Ly$. Through this case we will tune up the value of parameters λ_1 and c_1 within the allowable region in the previous section.

3.1 Parallel Algorithm

The heat equation on the interval $[0, 1]$ with the conduction constant $1/c$ subject to the initial condition $\sin \pi x$ and the Dirichlet boundary condition turns out to the m-dimensional linear ODEs with tridiagonal constant matrix through usual centered differencing. This is our model equation.

The linear case reduces the IRK stepping Eq.6 to the following:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{l=1}^{3}\beta_l \boldsymbol{k}_l, \qquad \boldsymbol{k}_i = L\left(\boldsymbol{y}_n + h\sum_{j=1}^{3}\alpha_{ij}\boldsymbol{k}_j\right), \tag{15}$$

Taking Eq.13 into account and introducing new varaibles

$$\boldsymbol{u}_i \equiv L^{-1}\left(s_{i1}\boldsymbol{k}_1 + s_{i2}\boldsymbol{k}_2 + s_{i3}\boldsymbol{k}_3\right), \qquad \boldsymbol{v}_i^n \equiv \left(s_{i1} + s_{i2} + s_{i3}\right)\boldsymbol{y}_n, \tag{16}$$

then we arrive at

$$\begin{pmatrix} I - h\lambda_1 L & 0 & 0 \\ 0 & I - h\lambda_2 L & 0 \\ 0 & 0 & I - h\lambda_3 L \end{pmatrix}\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_1^n \\ \boldsymbol{v}_2^n \\ \boldsymbol{v}_3^n \end{pmatrix}, \tag{17}$$

$$\begin{pmatrix} \boldsymbol{v}_1^{n+1} \\ \boldsymbol{v}_2^{n+1} \\ \boldsymbol{v}_3^{n+1} \end{pmatrix} = \left(1 - \boldsymbol{b}^t A^{-1}\boldsymbol{e}\right)\begin{pmatrix} \boldsymbol{v}_1^n \\ \boldsymbol{v}_2^n \\ \boldsymbol{v}_3^n \end{pmatrix} + \begin{pmatrix} \gamma_{11}\boldsymbol{u}_1 + \gamma_{12}\boldsymbol{u}_2 + \gamma_{13}\boldsymbol{u}_3 \\ \gamma_{21}\boldsymbol{u}_1 + \gamma_{22}\boldsymbol{u}_2 + \gamma_{23}\boldsymbol{u}_3 \\ \gamma_{31}\boldsymbol{u}_1 + \gamma_{32}\boldsymbol{u}_2 + \gamma_{33}\boldsymbol{u}_3 \end{pmatrix}, \tag{18}$$

$$(\gamma_{ij}) = \Gamma = S\left(\boldsymbol{e}\boldsymbol{b}^t\right)A^{-1}S^{-1}, \qquad \boldsymbol{e} = (1, 1, 1)^t.$$

Applying the above formulae, we can give our parallel algorithm (IRK34) as follows:

(i) Compute $\boldsymbol{v}_i^0$ by initial value $\boldsymbol{y}_0$ and Eq.16.

(ii) Iterate the following processes for $n = 0, 1, 2, \cdots$.

  (a) Solve Eq.17 in parallel.

  (b) Compute $\boldsymbol{v}_i^{n+1}$ from Eq.18 in parallel.

(iii) Transform $\boldsymbol{v}_i^n$ into $\boldsymbol{y}_n$ by Eq.16.

Figure 2 shows the flowchart of this algorithm. It is suitable to the shared-memory multiprocessor parallel computers such as CARY Y-MP or CONVEX240.

3.2 Tuning of the Parameters

We solved the model equation for $0 \leq t \leq 16$ with the time-step $h = 1/4$ by our parallel algorithm varying the parameters λ_1 and c_1. Problem parameters were $c = 100\pi^2, m = 5000$. The results are listed in Table 1, which shows that the smaller the value κ_{IRK} is, the more accurate the solution becomes. Here we define

$$\kappa_{\mathrm{IRK}} \equiv (\text{the minimum eigenvalue of the matrix} A) \left(1 - \boldsymbol{b}^t A^{-1} \boldsymbol{e}\right)^2$$

The following Theorem suggests this phenomena.

Theorem 2 *Given two linear systems of equations by*

$$(I - h\lambda_1 L)\boldsymbol{x} = \boldsymbol{b},$$

$$(I - h\lambda_2 L)\boldsymbol{x} = \boldsymbol{b}.$$

Let κ_1 and κ_2 be the condition numbers of above equations, respectively. Suppose L is a symmetric negative definite matrix and the eigenvalues of L are all distinct. If $0 < \lambda_1 < \lambda_2$ then $\kappa_1 < \kappa_2$. Namely, there is a positive constant C, which depends on h and the eigenvalues of L, satisfying the inequality

$$\kappa_2 - \kappa_1 \leq C\,(\lambda_2 - \lambda_1)\,.$$

Specifically, if the eigenvalues of L are the same, then $C = 0$, in short $\kappa_1 = \kappa_2$.

Application of this Theorem to $(I - h\lambda_i L)\,\boldsymbol{u}_i = \boldsymbol{v}_i^n$ in Eq.17 supports the phenomena.

From the region of parameters (c_1, λ_1) in Fig. 1 satisfying all the imposed conditions, we fix the pairs

$$c_1 = 8, \qquad \lambda_1 = \frac{3}{2}.$$

Then the other formula parameters are given through Eqs.11 and 12 as

$$c_2 = \frac{1229 - \sqrt{770563}}{778}, \qquad c_3 = \frac{1229 + \sqrt{770563}}{778},$$

$$\lambda_2 = \frac{5181 + \sqrt{3166665}}{4688}, \qquad \lambda_3 = \frac{5181 - \sqrt{3166665}}{4688}.$$

These values determine the 3-stage fourth order A-stable IRK which will be implemented hereafter.

Table 1. The effect of the parameter choosing.

λ_1, c_1	$-\log_2(\text{error})$	minimum eigenvalue of A	$1 - \boldsymbol{b}^t A^{-1}\boldsymbol{e}$	κ_{IRK}
3/2 , 8	27.580	0.7286	-0.6707	0.3278
2 , 3	27.549	0.6627	-0.7065	0.3308
3 , 3	27.523	0.6584	-0.7170	0.3384
1 , 7	27.299	0.9461	-0.6369	0.3838

3.3 Comparison with Voss' Method

Voss' method[12] is another type IRK method to solve autonoumous ODEs. When applied to $f(y) = Ly$, Voss' method, which can be considered as one of mono-implicit RKs[4], is given by

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{\nu=1}^{7} w_\nu k_\nu, \\ k_i &= L\left(y_{n+1} + h\sum_{j=1}^{i-1} a_j k_j\right) \quad (1 \le i \le 4), \\ k_i &= L\left(y_n + h\sum_{j=5}^{i-1} a_{ij} k_j\right) \quad (5 \le i \le 7). \end{aligned}$$

The formula parameters w_ν, a_j, a_{ij} were specified by him.

The parallel algorithm of Voss' method is given in Fig. 3 as its flowchart form. Here we introduce $F(y_n)$ and C_i as

$$F(y_n) \equiv -h\sum_{\nu=1}^{4} w_\nu L\left(y_{n+1} + h\sum_{j=1}^{\nu-1} a_j k'_j\right) - h\sum_{\nu=5}^{7} w_\nu L\left(y_n + h\sum_{j=5}^{\nu-1} a_{\nu j} k'_j\right),$$

$$C_i \equiv \frac{w_i^3}{\prod_{\substack{j=1 \\ j\neq i}}^{4} (w_i - w_j)} \qquad 1 \le i \le 4.$$

Like as in our algoritm, it can be carried out on the shared-memory multiprocessor parallel computers.

We compared two methods (IRK34 and Voss' methods) on the parallel computers, CRAY Y-MP and CONVEX240 through the test problem given in this section. The results for the numerical integration of the model equation with $h = 1/4$ are listed in Tables 2 and 3. The accuracy turned out to $-\log_{10}(\text{error}) = 8.936$ in IRK34 case while to $-\log_{10}(\text{error}) = 8.371$ in Voss' case. We can observe from both Tables that parallel IRKs attain speedup almost equal to the number of stages of the formula. This may be a siginificant feature of parallel IRKs. Secondly our method is less in the elapsed time than Voss' method, which takes one stage more than ours. This also suggests that for a parallel IRK with less parallel performance in comparison with the increase of stage numbers, the sequencial computation may win over the parallel one when the computational costs of k_i is cheap.

4 Parallel Algorithm for the Variable Coefficient Case

In this section we will consider the case of variable coefficient ODEs $y' = L(t)y + g(t)$.

4.1 Parallel Algorithm

Let us introduce the following notations.

$$L_i^n \equiv L(t_n + c_i h), \quad g_i^n \equiv g(t_n + c_i h), \quad f_i^n \equiv L_i^n y_n + g_i^n,$$

$$\boldsymbol{L}_n \equiv \begin{pmatrix} L_1^n & \mathbf{o} & \mathbf{o} \\ \mathbf{o} & L_2^n & \mathbf{o} \\ \mathbf{o} & \mathbf{o} & L_3^n \end{pmatrix}, \quad \boldsymbol{K} \equiv \begin{pmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \boldsymbol{k}_3 \end{pmatrix}, \quad \boldsymbol{F}_n \equiv \begin{pmatrix} \boldsymbol{f}_1^n \\ \boldsymbol{f}_2^n \\ \boldsymbol{f}_3^n \end{pmatrix}.$$

To get $\boldsymbol{k}_i$, we have to solve the equation

$$\Phi(\boldsymbol{K}) \equiv (I - h\boldsymbol{L}_n(A \otimes I))\boldsymbol{K} - \boldsymbol{F}_n = 0. \tag{19}$$

In order to keep the parallelism, we apply Newton's iteration on the equation in spite of the fact that Eq.19 is a system of *linear* equations. Newton's iteration generates a sequence of approximations $\boldsymbol{K}^{(l)}$ by the recurrence

$$\boldsymbol{K}^{(l+1)} = \boldsymbol{K}^{(l)} - (I - h\boldsymbol{L}_n(A \otimes I))^{-1}\Phi\left(\boldsymbol{K}^{(l)}\right) \qquad l = 0, 1, \cdots. \tag{20}$$

Taking advantage of the diagonalization of the matrix A and of the approximate equality $\boldsymbol{L}_n(S^{-1} \otimes I) \approx (S^{-1} \otimes I)\boldsymbol{L}_n$, we convert Eq.20 to

$$(I - h\boldsymbol{L}_n(\Lambda \otimes I))\Delta\boldsymbol{U} = (S \otimes I)\Phi\left(\boldsymbol{K}^{(l)}\right) \tag{21}$$

and

$$\boldsymbol{U}^{(l+1)} = \boldsymbol{U}^{(l)} - \Delta\boldsymbol{U} \qquad l = 0, 1, \cdots, \tag{22}$$

where $\boldsymbol{U}^{(l)} \equiv (S \otimes I)\boldsymbol{K}^{(l)}$.

After all, for the variable coefficient case the parallel algorithm (IRK34V) becomes the following:

Repeat the following processes for $n = 0, 1, 2, \cdots$.

(i) Compute $\boldsymbol{f}_i^n \quad (i = 1, 2, 3)$

(ii) $\boldsymbol{x}_i = s_{i1}\boldsymbol{f}_1^n + s_{i2}\boldsymbol{f}_2^n + s_{i3}\boldsymbol{f}_3^n \quad (i = 1, 2, 3)$

(iii) Decompose $(I - h\lambda_i L_i)$ and solve $(I - h\lambda_i L_i)\boldsymbol{u}_i = \boldsymbol{x}_i \quad (i = 1, 2, 3)$.

(iv) Iterate the processes.

- (a) $\boldsymbol{k}_i = p_{i1}\boldsymbol{u}_1 + p_{i2}\boldsymbol{u}_2 + p_{i3}\boldsymbol{u}_3 \quad (i = 1, 2, 3)$
- (b) $\boldsymbol{r}_i = \boldsymbol{k}_i - hL_i(\alpha_{i1}\boldsymbol{k}_1 + \alpha_{i2}\boldsymbol{k}_2 + \alpha_{i3}\boldsymbol{k}_3) - \boldsymbol{f}_i^n \quad (i = 1, 2, 3)$
- (c) $z_i = \|\boldsymbol{r}_i\| \quad (i = 1, 2, 3)$
- (d) If first iteration, then
  $z_0 = \max(z_1, z_2, z_3)$ and if $z_0 < \varepsilon_{TOL}$ then go to step(v)
  else
  if $\max(z_1, z_2, z_3) < \varepsilon_{TOL} z_0$ then go to step(v)
- (e) $\boldsymbol{q}_i = s_{i1}\boldsymbol{r}_1 + s_{i2}\boldsymbol{r}_2 + s_{i3}\boldsymbol{r}_3 \quad (i = 1, 2, 3)$
- (f) Solve $(I - h\lambda_i L_i)\boldsymbol{u}_i' = \boldsymbol{q}_i \quad (i = 1, 2, 3)$ with the latest decomposition.
- (g) $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i - \boldsymbol{u}_i' \quad (i = 1, 2, 3)$

(v) $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{l=1}^{3}\beta_l\boldsymbol{k}_l$

Note that the above algorithm can be computed in parallel with 3 processors.

4.2 Numerical Experiments

After fixing the number of inner iterations so as to achieve a sufficient accuracy, we made numerical experiments for the problem derived from the space-discretized Burgers equation whose analytical solution is known[3]. The ODEs are derived via the space-discretization with Kawamura-Kuwahara's third order finite difference scheme. The stiffness ratio for $m = 1000$ is approximately equal to 10^5.

We compared our algorithm (IRK34) with Gear's method (LSODE in netlib). The result is given in Table 4. This shows that regrettably our method is worse than Gear's. The reason is likely that the approximation $\tilde{u}_j = u_j(t_n) \approx u_j(t_n + c_i h)$ is poor because of $c_1 = 8$. However no formula in our consideration exists in $0 < c_1 < 1$ as pointed in Section 2. More investigation might be called for a wider class of parallelizable implicit RKs.

5 Conclusion and Future Works

In this note we get the followings.

1. Our new 3-stage fourth order formula is A-stable and using the parallel algorithm, 2.9 speed-up is aatained. This fact strongly suggests that in the type of parallelizable IRKs considered in this note parallelism across stage would be much more efficient in the large number of stages.

2. The minimum eigenvalue of the formula parameter matrix A, together with the square of the stability factor $R(\infty)$, influences the accuracy of numerical solution. Thus we define the quantity κ_{IRK}.

3. No L-stable formula exists in this collocation type.

4. Our algorithm for $y' = L(t)y + g(t)$ is worse than Gear's method from the viewpoint of CPU time.

Future works are expected in several points.

Stepsize control. Taking advantage of parallel computers, we might be able to control the stepsize h by computing fourth and third formulae in parallel.

Extension to DAEs $M(t)y' = f(t, y)$. In this case, we expect that the matrices $(I - h\lambda_i L_i^n)$ in the step (iii) as well as (f) in (iv) of the algorithm in Section 4 are replaced by the matrices $(M - h\lambda_i J)$, where $J \equiv \dfrac{\partial f}{\partial y}$.

Acknowledgement

The parallel computations were performed through the kind help by Mr. Yutaka MATSUNAGA (Tokyo Electron, Inc.) and Dr. Qasim SHEIKH (Cray Research, Inc.). The algorithm in Section 3.3 is given by the courtesy of Prof. Dave VOSS (Western Illinois University).

References

1. Bales, L. A., Karakashian, O. A. and Serbin, S. M., On the A_0-acceptability of rational approximations to the exponential function with only real poles, *BIT,* **28** (1988), 70 – 79.

2. Burrage, K., Efficient block predictor-corrector methods with a small number of corrections, *Proceedings of 13th IMACS World Congress on Computation and Applied Mathematics 1991,* pp268 – 269.

3. Byrne, G.D., Hindmarsh, A.C. and Brown, H.G., A comparison of two ODE codes GEAR and EPISODE, *Comput. Chem. Eng.,* **1** (1977), 133 – 147.

4. Cash, J. R. and Singhal, A., Mono-implicit Runge-Kutta formulae for the numerical integration of stiff systems, *IMA J. Numer. Anal.,* **2** (1982), 211 – 217.

5. Gear, C.W., Waveform methods for space and time parallelism, *J. Comput. Appl. Math.,* **38**(1991), 137 – 147.

6. Goto, A. and Mitsui, T., A 3-stage fourth order RK formula for parallel computers, *Trans. Japan SIAM,* **1**(1991), 291 – 304 (in Japanese).

7. Jackson, K.R., A survey of parallel numerical methods for initial value problems for ordinary differential equations, *IEEE Trans. Magnetics,* **27**(1991), 3792 – 3797.

8. Karakashian, O. A. and Rust, W., On the parallel implementation of implicit Runge-Kutta methods, *SIAM J. Sci. Stat. Comput.,* **9** (1988), 1085 – 1090.

9. Karakashian, O. A., On Runge-Kutta methods for parabolic problems with time-dependent coefficients, *Math. Comp.,* **47** (1986), 77 – 101.

10. Miranker, W.L. & Liniger, W., Parallel methods for the numerical integration of ordinary differential equations, *Math. Comp.,* **21**(1967), 303 – 320.

11. Van der Houwen, P.J. & Sommeijer, B.P., Parallel iteration of high-order Runge-Kutta methods with stepsize control, *J. Comput. Appl. Math.,* **29**(1990), 111 – 127.

12. Voss, D. A., Parallel Runge-Kutta methods for stiff ODEs, submitted to publication.

Table 2. IRK34

	elapsed time (msec.)		
	1CPU	3CPU	speed up
Y-MP	606	205	2.97
CONVEX	2989	1032	2.89

Table 3. Voss' method

	elapsed time (msec.)		
	1CPU	4CPU	speed up
Y-MP	882	292	3.02
CONVEX	4876	1412	3.45

Table 4. The comparison with Gear's method.

	$-\log_{10}(error)$	CPU time (sec.)
LSODE	1.573	58
IRK34V	1.256	7493

Figure 1. The region of the desired A-stable formula

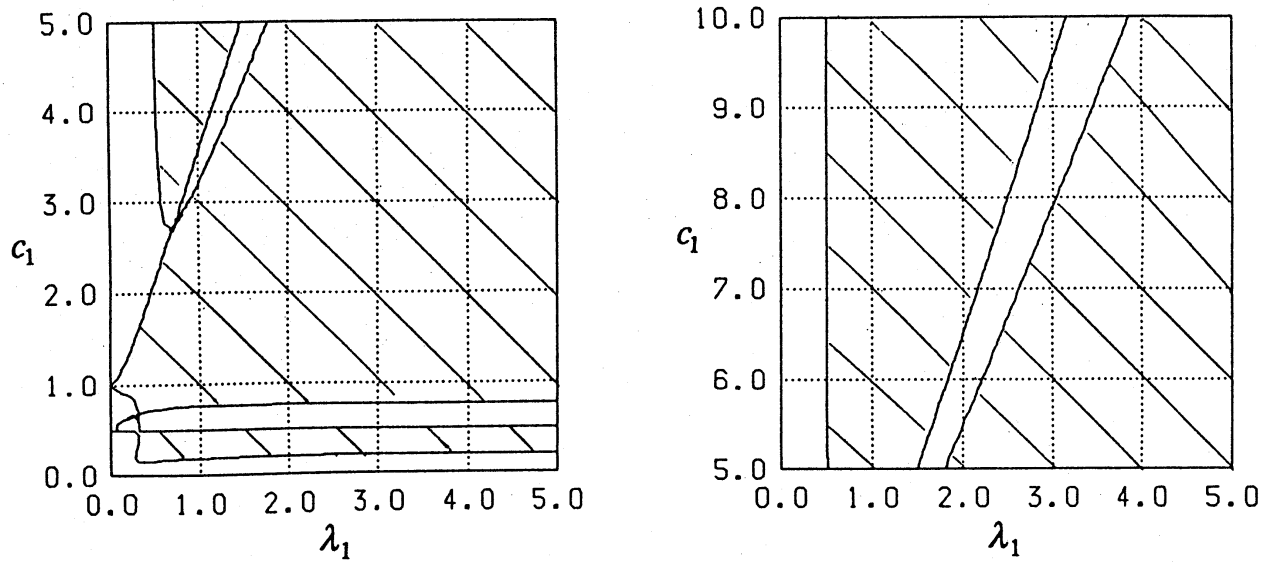

Figure 2. Flowchart of IRK34

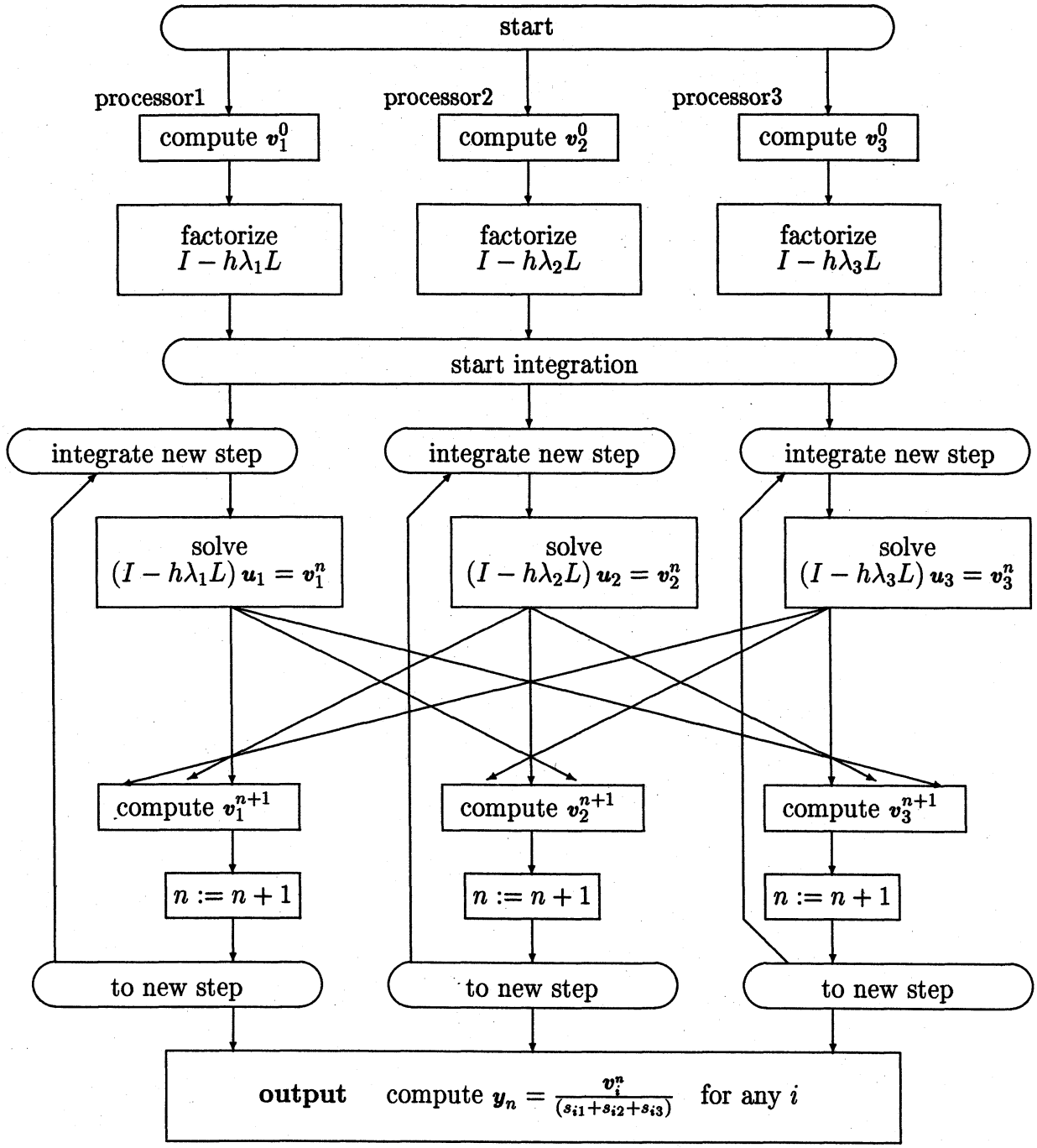

Figure 3. Flowchart of Voss' method.