PARALLEL/VECTOR INTEGRATION METHODS FOR DYNAMICAL ASTRONOMY

TOSHIO FUKUSHIMA
National Astronomical Observatory
2-21-1, Ohsawa, Mitaka, Tokyo 181-8588, Japan
toshio@nao.ac.jp

Abstract. This paper reviews three recent[1] works on the numerical methods to integrate ordinary differential equations (ODE), which are specially designed for parallel, vector, and/or multi-processor-unit (PU) computers. The first is the Picard-Chebyshev method (Fukushima, 1997a). It obtains a global solution of ODE in the form of Chebyshev polynomial of large (> 1000) degree by applying the Picard iteration repeatedly. The iteration converges for smooth problems and/or perturbed dynamics. The method runs around 100-1000 times faster in the vector mode than in the scalar mode of a certain computer with vector processors (Fukushima, 1997b). The second is a parallelization of a symplectic integrator (Saha *et al.*, 1997). It regards the implicit midpoint rules covering thousands of timesteps as large-scale nonlinear equations and solves them by the fixed-point iteration. The method is applicable to Hamiltonian systems and is expected to lead an acceleration factor of around 50 in parallel computers with more than 1000 PUs. The last is a parallelization of the extrapolation method (Ito and Fukushima, 1997). It performs trial integrations in parallel. Also the trial integrations are further accelerated by balancing computational load among PUs by the technique of *folding*. The method is all-purpose and achieves an acceleration factor of around 3.5 by using several PUs. Finally, we give a perspective on the parallelization of some implicit integrators which require multiple corrections in solving implicit formulas like the implicit Hermitian integrators (Makino and Aarseth, 1992), (Hut *et al.*, 1995) or the implicit symmetric multistep methods (Fukushima, 1998), (Fukushima, 1999).

Key words: numerical integration – orbit – celestial mechanics

1. Introduction

It has been claimed that vector/parallel computation is not effective for the numerical integration of dynamics with small number of freedom such as the orbital and rotational motions of planets and satellites. This is due to the step-by-step nature of existing numerical integrators such as the Runge-Kutta methods, the linear multistep methods, and the extrapolation methods((Hairer *et al.*, 1993). For example, the construction of Runge-Kutta methods takes the advantage of serial processing by assuming, in each phase, the availability of the result of all previous test integrations. The situation is unchanged in the symplectic methods (Kinoshita *et al.*, 1991), (Wisdom and Holman, 1991), which has been widely spread in the field of dynamical astronomy.

Recently, however, some papers appeared to destroy this barrier. They are

1. Picard-Chebyshev method (Fukushima, 1997a), (Fukushima, 1997b),
2. Parallel symplectic integrator (Saha *et al.*, 1997), and
3. Parallelized extrapolation method (Ito and Fukushima, 1997).

The first two seem[2] to be based on an idea to regard the ordinary differential equation as a one-dimensional partial differential equation. Parallel/vector com-

[1] All of the papers appeared in the Astronomical Journal in 1997.
[2] We are not sure whether Saha *et al.* had it in their minds when they wrote the paper.

Celestial Mechanics and Dynamical Astronomy **73**: 231–241, 1999.

puters are widely used in solving some partial differential equations. The key of parallelization is to rewrite the differential equations into a large set of nonlinear equations and to solve them by iterative procedures.

There are two ways to rewrite the differential equations into nonlinear equations. The one is to introduce the orthogonal function expansions in the expression of solutions. In this case, not the value of variables but the expansion coefficients are solved iteratively. The Picard-Chebyshev method is one of this family. It expands the solution in the form of Chebyshev polynomial. The other is the discretization. Saha *et al.* (1997) tactfully applies this idea to Hamiltonian systems and lead successfully a parallelization of a symplectic integration scheme, the implicit midpoint rule. On the other hand, the extrapolation method itself is easy to be adapted to parallel computation, since the method is based on the compilation of results of several (4-10) test integrations, which can be done in parallel. However, it was not experimented until the work of Ito and Fukushima (1997). The essence of their parallelization is the idea of folding, which makes the load balance among multiple processors almost equal.

In this short article, we will review these works. Before doing so, however, we make some remarks on the characteristics of these three new methods. First, apart from its parallel nature, the Picard-Chebyshev method is based on an approach being quite different from the existing integrators like Runge-Kutta methods, and other step-by-step integrators. Thus, we will discuss it in details. While, the last two methods are the parallelization of the existing methods; symplectic and extrapolation methods. The solution produced by these are just the same as those by the corresponding serial methods. Thus the readers can refer the literature of serial versions, say the textbook of Hairer *et al.* (1993), for the integration error and other properties except one thing; the speed-up by parallelization. Thus, we discuss only this factor.

Apart from the three parallel methods discussed, there remains a possibility to parallelize the multistep methods (Miranker and Liniger, 1967). The key idea is the concept of *pipeline*, which means the shift of timing between the predictor and the corrector(s). We will add a perspective on this approach.

2. Picard-Chebyshev Method

Consider solving a general first-order ordinary differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t), \qquad y(t_0) = y_0 \tag{1}$$

One way is to start from an approximate solution $y^{(0)}(t)$ and to improve it iteratively. The series of the refined solutions, $y^{(n)}(t)$ for $n = 1, 2, \cdots$, are obtained successively by computing

$$y^{(n)}(t) = y_0 + \int_{t_0}^{t} f\left(y^{(n-1)}(s), s\right) ds \tag{2}$$

This is the Picard iteration method ((Hairer *et al.*, 1993), Section I.8).

We expand the solution $y(t)$ in a linear form of Chebyshev polynomials whose support is a certain long interval of period, say thousands of nominal orbital periods, for example. Then, each Picard iteration ((Fukushima, 1997a), Eq.(18)) is rewritten in a vector form mapping ((Fukushima, 1997b), Eq.(2)) as

$$\mathbf{Y}^{(n-1)} \to \mathbf{Y}^{(n)} \equiv \mathcal{G}\mathbf{f}\left(\mathcal{C}^T\mathbf{Y}^{(n-1)}, \mathbf{t}\right) \tag{3}$$

where $\mathbf{Y}^{(n)}$ is a column vector of the Chebyshev coefficients of the n-th approximate solution, $\mathcal{G}$ and $\mathcal{C}$ are certain matrices, $\mathbf{t}$ is a column vector of the zeros of the Chebyshev polynomial of the largest degree, and $\mathbf{f}$ means a vector notation of function evaluations. Of course, the Picard iteration method is not all-purpose. It works only if the iteration converges. When the perturbation is sufficiently small, (1) the zero polynomial is enough as a predictor, (2) the iteration converges rapidly, and (3) the integration interval for a single polynomial can be extended as long as hundreds of characteristic periods. As an example, Figure 1, which is taken from Figure 2 of (Fukushima, 1997a), shows the error distribution of intermediate solutions for a test problem integrated over 64 orbital periods;

$$\frac{\mathrm{d}y}{\mathrm{dt}} = \cos(t + \epsilon y) \tag{4}$$

where the perturbation parameter ϵ was set as 10^{-3}.

In this figure, note that the final error is quite small, of the order of 10^{-13} or so. Next, remark that the final error distribution is quite different from those obtained by other type of integrators, namely being roughly uniform through the integration period. This comes from the almost mini-max nature of Chebyshev approximation. Further, if we inspect them in details, we will learn that the errors in the middle of the integration period are somewhat larger than those at the final epoch of the integration period. This owes to the fact that the distribution of the evaluation points, i.e. the zeros of a certain high-degree Chebyshev polynomial, is more sparse around the center than near the ends. Also this figure supports the expectation that the iteration converges linearly, although the speed of convergence is somewhat slower than the expected rate, ϵ.

As for the computing speed, we remark that the function evaluation in the above mapping expression can be done in parallel, or more precisely speaking, can be vectorized easily. Figure 2, which is taken from Figure 1 of (Fukushima, 1997b), shows the comparison of the wall-clock time[3] for the Adams method in the scalar mode, the Picard-Chebyshev method in the scalar mode, and in the vector mode of the same computer, Fujitsu VX-1R. The curves in the figure are drawn as functions of the computational amount of function evaluations. Since the overhead of Picard-Chebyshev methods is larger than that of Adams method, the former outperforms the latter especially when the load of function evaluation is heavy.

[3] The usual CPU time is not appropriate in evaluating the performance of parallel/vector computers. Instead, used is the clock time in the real world, which is named *wall-clock* time.

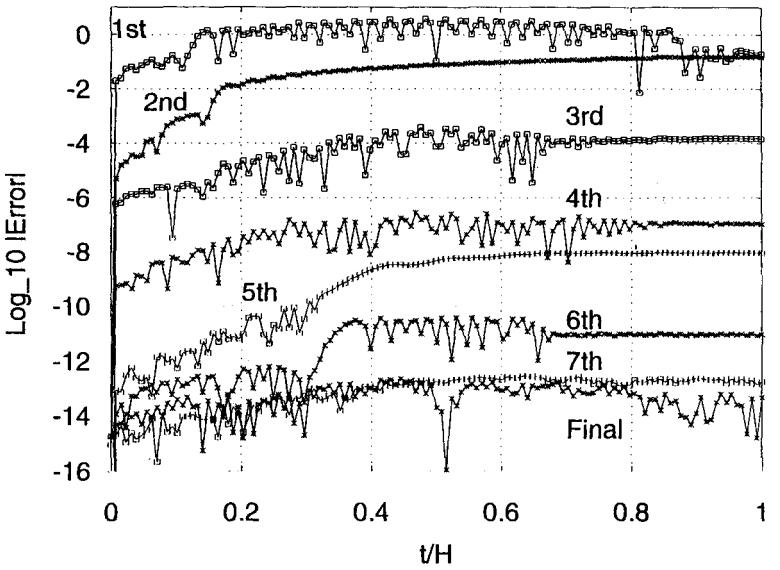

Fig. 1. Convergence of Picard-Chebyshev Method

Remark that the wall-clock time of Picard-Chebyshev method depends on the magnitude of perturbation parameter. In fact, if ϵ of Eq.(4) is zero, the problem reduces to the numerical quadrature of a known function of time, and therefore there is no need to repeat the Picard iteration. Then the total wall-clock time reduces by the factor of number of Picard iterations, 8 for $\epsilon = 10^{-3}$. Practically, there is a limitation of such reduction. At least two computation (i.e. one Picard iteration) is required to confirm the convergence. Thus the speed-up of factor 4 or 5 is expected for sufficiently small ϵ. Figure 4 of Fukushima (1997a) supports this expectation.

As for the applicability to long integrations, the readers may refer Section 4.6 and Figure 5 of Fukushima (1997a). The error of the Picard-Chebyshev method increases in proportion to the 3/2 power of the integration period. Also the computational time does the same. This means that integrations over a very long period are not suitable. Rather, the method works best when integrating problems over a middle-size period, say the period of hundreds to thousands nominal revolutions. A typical problem would be the orbital improvement of Moon's orbit and rotation over 25 years of LLR observation, which includes some 300 revolutions.

In conclusion, the Picard-Chebyshev method directly provides the nearly minimax-approximated polynomials interpolating the solution almost uniformly within the whole integration interval. Therefore, the method is especially suitable for the orbit improvement where the previous ephemeris serves as a good approximation.

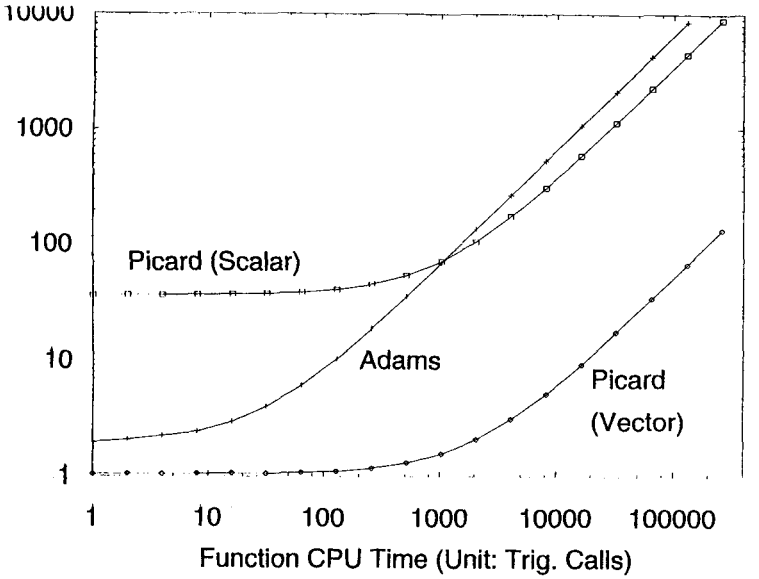

Fig. 2. Acceleration Factor of Picard-Chebyshev Method

Also the method is appropriate to solve perturbed dynamics where an approximate solution is known analytically. Typical examples would be the planetary motions, the satellite motions, the motions of comets and minor planets in Encke's method, and the rotational motions of the Earth and Moon.

3. Parallel Symplectic Integrator

The typical way to solve the initial value problem of ordinary differential equations for a long time span is to discretize the integration period into thousands of small time intervals and to go step by step from the initial epoch. To do this in parallel, let us rewrite the integral expression

$$y(t) = y_0 + \int_{t_0}^{t} f\left(y(s), s\right) ds \tag{5}$$

into a following discretized form by using the implicit midpoint rule;

$$y_m = y_0 + h \sum_{k=0}^{m-1} f\left(\frac{y_k + y_{k+1}}{2}\right) \tag{6}$$

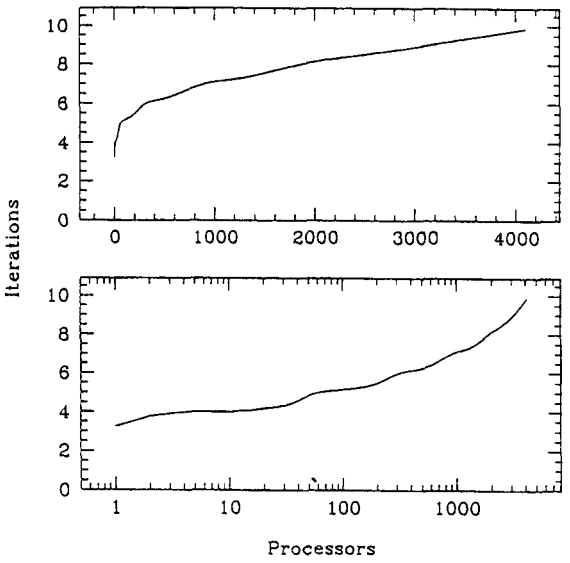

Fig. 3. Convergence of Parallel Symplectic Integrator

These can be regarded as a large set of nonlinear equations. Consider to solve them by a certain iterative procedure like

$$y_m^{(n-1)} \to y_m^{(n)} \equiv y_0 + h \sum_{k=0}^{m-1} f\left(\frac{y_k^{(n-1)} + y_{k+1}^{(n-1)}}{2}\right) \tag{7}$$

Here it is easy to see that the function evaluation can be done in parallel.

Saha *et al.* (1997) applied this parallelization to integrate a perturbed Hamiltonian system;

$$H = H_0(p) + \epsilon H_1(p, q) \tag{8}$$

The result is a following iterative scheme;

$$p_m^{(n)} \leftarrow p_0 + u_m\left(P^{(n-1)}, Q^{(n-1)}\right),$$

$$q_m^{(n)} \leftarrow q_0 + V_m\left(P^{(n)}\right) + v_m\left(P^{(n-1)}, Q^{(n-1)}\right) \tag{9}$$

where

$$P = (p_0, p_1, \cdots), \qquad Q = (q_0, q_1, \cdots),$$

$$V_m(P) = h \sum_{k=0}^{m-1} \left(\frac{\partial H_0}{\partial p}\right)\left(\frac{p_k + p_{k+1}}{2}\right),$$

$$u_m(P,Q) = -\epsilon h \sum_{k=0}^{m-1} \left(\frac{\partial H_1}{\partial q}\right)\left(\frac{p_k + p_{k+1}}{2}, \frac{q_k + q_{k+1}}{2}\right),$$

$$v_m(P,Q) = \epsilon h \sum_{k=0}^{m-1} \left(\frac{\partial H_1}{\partial p}\right)\left(\frac{p_k + p_{k+1}}{2}, \frac{q_k + q_{k+1}}{2}\right) \tag{10}$$

Remark that this formulation is symplectic since the argument of V_m is not $P^{(n-1)}$ but $P^{(n)}$, namely the momenta not before but after the kick. The local truncation error is of the order of ϵh^2, since the implicit midpoint rule is of the second order. The global truncation error is expected to grow linearly since the method is symplectic. The iteration converges linearly. Figure 3, which is taken from Figure 3 of Saha *et al.* (1997), shows the average number of iterations required for convergence as the number of simulated processors. Although they did not measure the performance by actual numerical experiments in parallel computers, based on this figure, Saha *et al.* (1997) gave an estimation of acceleration factor, i.e. the ratio of wall-clock time in parallel and serial computations, as much as around 50 for 1000 processors.

4. Parallelized Extrapolation Method

The extrapolation method is the direct application of Richardson's deferred extrapolation to the limit $h \to 0$ to the modified midpoint rule which assures the symmetric property and thus the h^2-expansion of the solution (Hairer *et al.*, 1993). Remark that all of the trial integrations, i.e. the integration by the modified midpoint rule with different h, can be done in parallel. Actually we prepare multiple processors sharing the memory and assign each processor unit (PU) to the test integration of different stepsizes like PU-1 for $h = H$, PU-2 for $h = H/2$, PU-3 for $h = H/3$, and so on[4]. Since the computational load of each test integration is inversely proportional to h, the above naive assignment allows an idling for the processors with larger h. In fact, in the above example, the PU-1 becomes idle just after one step integration, the PU-2 does after the second step, and so on. In order to avoid such an inefficiency, we introduce the concept of *folding*.

Imagine the case of 8-stage extrapolation method with the test stepsizes $h = H, H/2, \cdots, H/8$. Assume to prepare 4 PUs. If we assign processors with little care, like assigning PU-n to the test integration of $h = H/(2n-1)$ and $h = H/(2n)$, then the task of the PU-4, i.e. the test integrations of $h = H/7$ and $h = H/8$, becomes the bottle neck of the total procedures. In this case, the acceleration factor will be not so large as $(1+2+\cdots+8)/(7+8) = 2.4$. On the other hand, if we couple the test integration of $h = H/n$ with that of $h = H/(9-n)$, like $h = H$ and $h = H/8$, then the loads of 4 PUs become the same, and as a

[4] Here H is the basic stepsize of the extrapolation method, namely the amount of time advanced after the extrapolation, and h is the test stepsize, i.e. the stepsize of test integrations whose result are to be extrapolated.

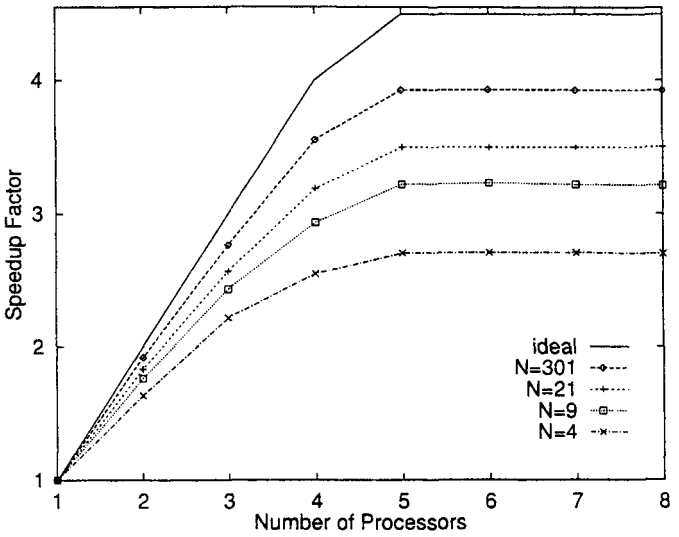

Fig. 4. Acceleration Factor of Parallelized Extrapolation Method

result, the acceleration factor becomes the same as the number of used processors as $(1+2+\cdots+8)/(1+8)=4$. This is the technique of *folding*. In their Table 2, Ito and Fukushima (1997) provides the optimal foldings for the cases of up to 10 PUs. By using these, they developed a parallelization of the extrapolation method. Figure 4, which is taken from Figure 3 of Ito and Fukushima (1997), shows the acceleration factor as a function of number of processors. Even for a system of small freedom as 9-bodies problem, the acceleration factor of around 3.5 is achieved by using 4 or 5 PUs. It is remarkable that this situation is unchanged even when allowing the stepsize and/or the order variable.

In summary, the parallelized extrapolation method is all-purpose and leads to a speedup factor of 3-4 by using 4 PUs or so.

5. Pipelined Predictor-Corrector Method

More than three decades ago, Miranker and Liniger (1967) proposed a parallel method to accelerate the predictor-corrector methods. Their main idea is, in each process of prediction and correction, to advance not only a single step but also multiple steps together. Take an example of predictor phase. The ordinary predictor has a following general form (Hairer *et al.*, 1993);

$$y_{n+1} = -\sum_{j=0}^{J} \alpha_j y_{n-j} + h \sum_{j=0}^{J} \beta_j f_{n-j} \tag{11}$$

Miranker and Liniger (1967) predict not only this value at the $(n+1)$-th step but also the values in future steps as

$$y_{n+k} = -\sum_{j=0}^{J} \alpha_j^{(k)} y_{n-j} + h \sum_{j=0}^{J} \beta_j^{(k)} f_{n-j}, \qquad (k = 2, \cdots) \tag{12}$$

Since these predictions can be done in parallel, one can advance multiple integration steps at a single wall-clock step by using multiple processors. Unfortunately, this seems not practical. Usually the stepsize in multistep methods are taken as large as possible while the methods are numerically stable. This means that, in such extreme cases, the prediction/correction for doubly- and further advanced steps would cause instability.

In the author's viewpoint, rather the second idea of Miranker and Liniger (1967), which we call *pipeline*, seems effective especially when handling predictor-corrector methods requiring multiple correction stages[5]. Of course, it might be effective. However, are there such complicate methods worth to be applied? Yes, certainly. Good examples are the implicit time-symmetric methods such as the implicit Hermitian integrator (Makino and Aarseth, 1992), (Hut *et al.*, 1995) and the implicit symmetric multistep methods (Fukushima, 1998), (Fukushima, 1999). Now, the idea of pipeline is as follows. Imagine to perform a predictor-corrector method requiring m-correction stages such as PE(CE)^m method[6]. Prepare $m+1$ PUs and assign the PU-0 to the PE process of the $(n+1)$-th step, the PU-1 to the first CE process of the n-th step, the PU-2 to the second CE process of the $(n-1)$-th step, and so on;

$$\text{PU}-0: \quad y_{n+1}^{(0)} = -\sum_{j=0}^{J} \alpha_j^{(P)} y_{n-j}^{(j)} + h \sum_{j=0}^{J} \beta_j^{(P)} f_{n-j}^{(j)}, \tag{13}$$

$$f_{n+1}^{(0)} = f\left(y_{n+1}^{(0)}, t_{n+1}\right) \tag{14}$$

$$\text{PU}-k: \quad y_{n+1-k}^{(k)} = -\sum_{j=1}^{J} \alpha_j^{(C)} y_{n+1-(j+k)}^{(j+k)} + h \sum_{j=0}^{J} \beta_j^{(C)} f_{n+1-(j+k)}^{(j+k+1)}, \tag{15}$$

$$f_{n+1-k}^{(k)} = f\left(y_{n+1-k}^{(k)}, t_{n+1-k}\right) \tag{16}$$

where $y_n^{(k)}$ denotes the value at the n-th step after k corrections and the superscripts of coefficients, (P) or (C), specifies the preditor or corrector, Remark that all these processes are independent with each other, and as a result, can be done in parallel.

[5] Miranker and Liniger (1967) applied the pipeline to the PECE method only, and therefore, did not stress so much about its applicability and effectiveness.

[6] Here P and C stand for the predictor and the corrector, respectively, while E does for the evaluator, i.e. the evaluation process of f.

TABLE I
Characteristics of Parallel/Vector Integration Methods

Method	Suitable Problems	Acceleration Factor
Picard-Chebyshev	Perturbed Dynamics	100-1000
Parallel Symplectic	Hamiltonian Systems	~ 50[7]
Parallel Extrapolation	General	3-4
Pipelined Multistep	Smooth	3-4[8]

[7] Expected [8] Not yet experimented

In other words, the pipelined predictor-corrector method can be done in the same wall-clock time as that of predictor-only formulas.

In general, the implicit methods have better properties than the explicit methods such as at the points of small error constants or of better numerical stability. Therefore, the technique of pipeline will enlighten the implicit symmetric methods.

6. Conclusion

We reviewed three numerical integrators designed for parallel/vector computers; the Picard-Chebyshev method (Fukushima, 1997a), (Fukushima, 1997b), the parallel symplectic integrator (Saha *et al.*, 1997), and the parallelized extrapolation method (Ito and Fukushima, 1997). We also proposed the fourth scheme; the pipelined predictor-corrector method based on the idea of Miranker and Liniger (1967). Their characteristics are summarized in Table I although the acceleration factor for the last method was based on a rough estimation of the reduction of the number of function evaluations by means of parallelization. Also in listing the factor of Picard-Chebyshev method, we took the effect of magnitude of perturbation into account. In the case of parallel symplectic method, one may have an impression that the acceleration factor of 50 is a small gain at the cost of using 1000 processors. Further, the acceleration factor of 3 or 4 of the parallel extrapolation method and/or the pipelined multistep methods proposed here may be thought as a minor improvement. However, we stress that it is quite difficult to achieve a factor 2 in the parallelization of ordinary Runge-Kutta methods. Thus, we regard that even the factors 3 or 4 show good performance of the parallelizations.

Since the applicability of these methods are different one by one, it is difficult to recommend one of them. This comes from the fact that the comparison of serial integrators also depends on the problem to be solved. For example, in the case of orbital improvement, only one Picard iteration is enough. Thus the Picard-Chebyshev method would be the most appropriate. While, in doing long-time integration of Hamiltonian systems, symplectic methods are of the highest cost-

performance. Thus, it would be worth to apply its parallelization to such problems. On the other hand, the extrapolation method is known to be tough, namely to be able to handle violent situations as close encounters. Thus the parallel extrapolation method would be recommended to integrate non-smooth problems like pure three body problems.

Finally let us make a comment about the way of comparisons presented here. As for the speed of these parallel methods, we have only considered the factor of acceleration, which was defined as the inverse ratio of total wall-clock time of the same processor used in serial and parallel/vector modes, respectively. Of course, the fastest serial processor would be faster than the serial usage of the fastest parallel/vector processor. Thus, the acceleration factors we presented do not mean the ratio of the fastest serial computation and the fastest parallel/vector computation. To evaluate the ratio, one needs a number of state-of-the-art serial and parallel/vector computers. It is beyond our ability. Rather, we think it is more appropriate to separate such machine-dependent effects with the machine-independent part like the acceleration factor.

References

Fukushima, T.: 1997a, 'Picard Iteration Method, Chebyshev Polynomial Approximation, and Global Numerical Integration of Dynamical Motions', *Astron. J.*, **113**, 1909-1914.

Fukushima, T.: 1997b, 'Vector Integration of Dynamical Motions by the Picard-Chebyshev Method', *Astron. J.*, **113**, 2325-2328.

Fukushima, T.: 1998, 'Symmetric Multistep Methods Revisited', *in Proc. 30th Symp. on Cele. Mech. (Fukushima et al. eds)*, 229-247.

Fukushima, T.: 1999, 'Symmetric Multistep Methods Revisited II: Numerical Experiments', *in Proc. IAU Coll. No.173*, to be printed.

Hairer, E., Nϕrsett, S.P., and Wanner, G.: 1993, *Solving Ordinary Differential Equations I (2nd ed.)*, Springer-Verlag, Berlin.

Hut, P., Makino, J., and McMillan S.: 1995, *Astrophys. J. Lett.*, **443**, 93-.

Ito, T., and Fukushima, T.: 1997, 'Parallelized Extrapolation Method and Its Application to the Orbital Dynamics', *Astron. J.*, **114**, 1260-1267.

Kinoshita, H., Yoshida, H., and Nakai, H.: 1991, *Cele. Mech. and Dyn. Astr.*, **50**, 59.

Lambert, J.D., and Watson, I.A.: 1976, 'Symmetric Multistep Methods for Periodic Initial Value Problems', *J. Inst. Maths Applics*, **18**, 189-202.

Makino, J., and Aarseth S.J.: 1992, *Publ. Astron. Soc. Japan*, **44**, 141.

Miranker, W.L., and Liniger W.: 1967, 'Parallel Methods for the Numerical Integration of Ordinary Differential Equations', *Math. Comp.*, **21**, 303-320.

Quinlan, G.D., and Tremaine, S., 1990, 'Symmetric Multistep Methods for the Numerical Integration of Planetary Orbits', *Astron. J.*, **100**, 1694-1700.

Saha, P., Stadel, J., and Tremaine, S.: 1997, 'A Parallel Integration Method for Solar System Dynamics', *Astron. J.*, **114**, 409-415.

Wisdom, J., and Holman, M.: 1991, 'Symplectic Maps for the N-body Problem', *Astron. J.*, **102**, 1528-1538.