Materials of Conferences

PARALLEL EXPLICIT RUNGE-KUTTA METHOD 2ND ORDER: ACCURACY AND STABILITY CONTROL

[1]Novikov E.A., [2]Vashchenko G.V.
[1]Institute of Computational Modelling SB RAS,
[2]Institute of Computational Modelling SB RAS,
Siberian State Technological University,
Krasnoyarsk, Russia

Differential equations arise in many fields of application, such as in the simulation of phenomena in physics, mechanics, chemistry, biology and so forth. These equations are often in the form of a stiff initial value problems [1]-[3].

We describe a parallel explicit Runge-Kutta integration scheme second order in which an accuracy and stability control algorithm are included. The primary objective of this work is to illustrate a definite potential for the parallel performance.

Consider a stiff initial value problem

$$y' = f(y), \quad y(t_0) = y_0, \quad t_0 \le t \le t_k, \tag{1}$$

where y: $[t_0, t_k] \to R^N$,
f: $[t_0, t_k] \times R^N \to R^N$,
$[t_0, t_k]$ – interval integration.

Without loss of generality, assume that (1) is an autonomous system. Note that a non-autonomous system $y' = f(y, t)$ is always possible to write in an autonomous form as (1). Assume that there exists a unique solution to the problem (1).

Assume that p is an amount of processors on computational system, N is a dimension of the system (1) and $N > p$, we write a parallel explicit Runge-Kutta scheme as in [4]

$$y_{j_s}^{(n+1)} = y_{j_s}^{(n)} + 0.5\left(K_{1,j_s}^{(n)} + K_{2,j_s}^{(n)}\right),$$

$$y_{j_s}^{(0)} = y_{j_s}(t_0), \tag{2}$$

$$1 \le j \le p; (j-1)\cdot s \le j_s \le j \cdot s,$$

where $s = N/p$, if N multiple p, or $[N/p] + g$, otherwise,

$$y_{j_s}^{(n)} \in proc(j),$$

$$K_{1,j_s}^{(n)} = h_n f_{js}\left(y^{(n)}\right) \in proc(j),$$

$$K_{2,j_s}^{(n)} = h_n f_{js}\left(y^{(n)} + K_{1,j_s}^{(n)}\right) \in proc(j).$$

We design a parallel algorithm by using a graph reduction technique and decomposition technique on a subtasks [5], [6]. Our method based on the use of automatic control of accuracy and of stability dynamically as the solution develops. A value of a step size hn which will give the required solution with an estimated local error exactly equal to the requested tolerance and stability condition. The theoretical justification for using variable stepsize algorithms to integrate stiff problems has been given in [1]. It is shown that when a variable stepsize of integration is used, the efficiency of the explicit Runge -Kutta method can be increased by means of a step choosing algorithms in which an accuracy and stability are controlled. In our method, a stability control is based on using of a estimation of a largest eigenvalue of Jacobian matrix by a power method across of right side system (1) differential and the following control as $h \left| \lambda_{\max} \div \right| \le D$, where D is a stability region size. This approach leads not to increase in a calculation cost.

Parallel numerical algorithm is as follows. The $y_{j_s}^{(n)}$, $f_{js}\left(y^{(n)}\right)$ are distributed onto a p of processes according to the block scheme to ensure «good» load balance as well as the scalability of

the algorithm. An each task U_j is performed on one processor proc(j), $U_j \in proc(j)$. *Proc*(1) defines a value hn and broadcasts to the other *proc*(*j*), as operation one-to- all. An each *proc*(*j*) computes $y_{j_s}^{(n)}$ its part and broadcasts to the other *proc*(*j*), as *all-to-all*. In additional, *proc*(*j*) calculates a local norm, $\left\| K_{2,j_s}^{(n)} - K_{1,j_s}^{(n)} \right\|$ and sends to the *proc*(1).

The program code is written in C/C++ with MPI –functions. It is available from the authors. The computations were done on a cluster MVS ICM having 99 processors [7]. Some numerical results are presented to show the efficiency of the parallel method. In additional, we give the number of integration steps, number of a function *f* evaluations, number of callbacks, i.e. isa, ifu and iwo, respectively.

References

1. Novikov E. Explicit methods for stiff systems. – Novosibirsk: Nauka. RAN, 1997.

2. Hairer E., Wanner G. Solving Ordinary Differential Equations II, Springer-Verlag, 1996.

3. Butcher J.C. Numerical Methods for Ordinary Differential Equations // John Wiley&Sons. – 2008.

4. Vashchenko G.V., Novikov E.A. Parallel implementation of explicit Runge-Kutta methods // Vestnik KrasGau. – 2010. – №2. – P. 14–18.

5. Quinn M.J. Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill, 2004.

6. Hendrickson B., Tamara G. Kolda Graph partitioning models for parallel computing // Parallel Computing. – 2002. – Vol. 26, № 12. – P. 181–197.

7. Isaev S.V., Malyshev A.V., Shaidurov V.V. Development of the Krasnoyarsk's parallel computing centre // Computing Technologies. – 2006. – Vol. 11. – P. 28–33.

This work was supported by the RFFR under Grant No. 08-01-00621 and President under Grant SSCH-3431.2008.9.