Journal of Computational Mathematics, Vol.18, No.4, 2000, 375–386.

PARALLEL ROSENBROCK METHODS FOR SOLVING STIFF SYSTEMS IN REAL-TIME SIMULATION*

Li-rong Chen De-gui Liu
(*Beijing Institute of Computer Application and Simulation Technology, Beijing 100854, China*)

Abstract

In this paper parallel Rosenbrock methods in real-time simulation are presented on parallel computers. Their construction, their convergence and their numerical stability are studied, and the numerical simulation experiments are conducted on a personal computer and a parallel computer respectively.

Key words: Rosenbrock methods, stiff system, real-time simulation, parallel algorithms

1. Introduction

In the fields of astronautics engineering and continuous system simulation, many models are described by stiff ODE's. In order to simulate (especially in real-time) these systems we have to use speedy algorithms so as to complete the computation within the designated time. Papers [5,7,8] present parallel-iterated Runge-Kutta methods and implicit Runge-Kutta methods. Although these methods have higher stability, a heavier workload will be imposed by the iteration. And our inability to predict the number of iteration times in advance will cause us difficulties in the application of the above methods to real-time simulation.

Aimed at real-time simulation of stiffly large systems on parallel computers, parallel Rosenbrock methods(PRM) are constructed through a frontal approach. The different internal stages of Rosenbrock methods are calculated in parallel on different processors. The methods have higher stability and need no iteration. In each step only one Jacobian matrix and one LU-decomposition need to be computed.

2. A Class of Parallel Rosenbrock Methods

Consider the autonomous initial value problems

$$y'(x) = f(y(x)), \quad y(x_0) = y_0, \quad x \in [x_0, x_M], \quad y \in R^n \tag{1}$$

* Received April 21, 1997.

here $f(y)$ is assumed to possess derivatives of all orders which will be needed in the following discussion.

The class of (sequential) Rosenbrock methods has the following form[4]:

$$\begin{aligned} y_{n+1} &= y_n + \sum_{i=1}^{s} c_i l_i \\ (I - h\gamma J) l_i &= hf(y_n + \sum_{j=1}^{i-1} \alpha_{ij} l_j) + hJ \sum_{j=1}^{i-1} \gamma_{ij} l_j \\ & \qquad i = 1, 2, \ldots, s \end{aligned} \tag{2}$$

where $\gamma,\ \alpha_{ij},\ \gamma_{ij},\ c_i$ are real coefficients, I denotes identity matrix, J is Jacobian matrix $f_y(y_n)$. Through a frontal approach, using the information before time point t_n, a class of s-stage parallel Rosenbrock methods is constructed by the following form:

$$\begin{aligned} y_{n+1} &= y_n + \sum_{i=1}^{s} c_i l_{in} \\ (I - h\gamma J) l_{in} &= hf(y_n + \sum_{j=1}^{i-1} \alpha_{ij} l_{jn-1}) + hJ \sum_{j=1}^{i-1} \gamma_{ij} l_{jn-1} \\ & \qquad i = 1, 2, \ldots, s \end{aligned} \tag{3}$$

As the quantities $y_n, l_{in-1},\ i = 1, \ldots, s-1$ are known, $l_{1n}, l_{2n}, \ldots, l_{sn}$ can be calculated on s processors in parallel. The information flow that describes the parallel execution of two-stage formula on two processors $P_1,\ P_2$ is shown in Fig.1.

Let t_l denote the CPU time needed to compute each l_{in}, and t_y denote the CPU time needed to compute y_{n+1} from l_{in} (or l_n), and let t_{syn} denote the CPU time needed to exchange information on s processors. Then in each step the sequential costs are $t_T = st_l + t_y$, and the parallel costs on an s-processor system are $t_P = t_l + t_y + t_{syn}$. So the speed-up is

$$S_P = \frac{st_l + t_y}{t_l + t_y + t_{syn}}$$

Particularly, when $t_l \gg t_{syn},\ S_P \approx s$.

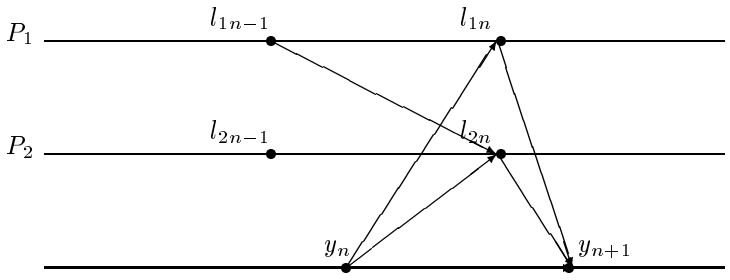

Fig 1. Information flow of parallel Rosenbrock formula