Journal of Computational Mathematics, Vol.9, No.1, 1991, 74–85.

TWO SYNCHRONOUS PARALLEL ALGORITHMS FOR PARTIAL DIFFERENTIAL EQUATIONS*

Lu Tao
(*Chengdu Branch, Academia Sinica, Chengdu, China*)

Shih Tsi-min Liem Chin-bo
(*Hong Kong Polytechnic, Hong Kong*)

In many problems involving practical applications, e.g., in exploitation of oil fields and gas fields, nuclear engineering, large-scale structural engineering and forecasting of climate and tides, it is frequently necessary to solve initial boundary value problems of high dimensional linear or nonlinear partial differential equations defined in large regions. Solving these problems requires a large amount of computation and sometimes, even rapid responses. The idea of parallel computer system encourages numerical analysts to develop more efficient methods. It seems that the study of parallel algorithms will soon become a new target in the development of numerical analysis[2].

Based on the Schwarz algorithm, L.S. Kang[1] introduced an asynchronous parallel algorithm for solving partial differential equations of the elliptic type. In this paper, we introduce two synchronous parallel algorithms for the solution of partial differential equations. First, the region where the equations are defined is divided into some subregions which may be overlapping, and then the equations are solved in each subregion independently. Finally, a simple arithmetic mean is assigned to each point within the overlapping parts of the subregions.

The first algorithm introduced in this paper is based on the groupwise projection iterative method. We need only to solve the least-squares problem in each subregion. It can be proved that the parallel solution converges to the solution of the discrete equations for the entire region. Moreover, this result is independent of the type of differential equations. The proof of the second algorithm is based on the discrete maximum principle.

In §1 and §2 We give the descriptions of the two parallel algorithms respectively. §3 introduces the groupwise projection iterative method, and the proofs of the convergence of the two algorithms are given in §4 and §5. In §6 a numerical experiment is presented.

§1. The Parallel Algorithm 1

The boundary value problem of a partial differential equation, in general, can be written as

*Received February 6, 1988.

$$\begin{cases} Lu = f, & \text{in } \Omega, \\ lu = g, & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where Ω is a bounded region with boundary $\partial\Omega$, L is a differential operator, and l is a boundary operator. We may apply either the finite difference or the finite element method to (1) and obtain an algebraic system. For instance, consider the Dirichlet problem of a second order elliptic equation with homogeneous boundary value. We may start from its weak form

$$a(u, v) = (f, v) \quad \text{for } v \in E'(\Omega) \tag{2}$$

and then discretize by the finite element method.

Take a regular triangulation of the region Ω to get Ω_h. At each $P_j \in \Omega_h$, we define a linear shape function φ_j. Let

$$S^h = \text{ Span } \{\varphi_j, \text{ for } P_j \in \Omega_h\} \tag{3}$$

where

$$\varphi_j(P_i) = \delta_{ji}, \text{ for } \varphi_j \in S_h.$$

Hence the finite element approximation $\{u^h\}$ satisfies

$$a(u^h, \varphi_j) = (f, \varphi_j), \quad j \in I \tag{4}$$

where I is the set of indices of all grid points in Ω_h. $N(P_j) = \text{supp } (\varphi_j)$ is called the discrete neighbourhood of the point P_j. $N(P_j)$ contains only P_j and a few neighbouring points. We denote $N(P_j)$ by N_j. Obviously, (4) can be written as

$$a(u^h, \varphi_j) = \sum_{P_i \in N_j} C_{ji} u^h(P_i) = f_j, \quad \text{for } j \in I. \tag{5}$$

In other cases, such as shape functions of a higher degree, or when a finite difference method or a collocation method is used, the corresponding discrete system can be obtained in a similar way. At each grid point $P_j \in \Omega_h$, establish an equation:

$$L^h u^h(P_j) = \sum_{i \in I} C_{ji} u^h(P_i) = f_j, \quad j \in I. \tag{6}$$

We define the discrete neighbourhood as

$$N_j \equiv N(P_j) \equiv \{P_i; C_{ji} \neq 0\}$$

and (6) now becomes

$$L^h u^h(P_j) = \sum_{P_i \in N_j} C_{ji} u^h(P_i) = f_j, \quad j \in I. \tag{7}$$

In order to solve (7) by a parallel algorithm, we divide Ω_h into m subregions $\Omega_h^1, \cdots, \Omega_h^m$, $\Omega_h = \sum_{i=1}^{m} \Omega_h^i$. Some of the subregions can be overlapping. To reduce the waiting time among