The Scattered Decomposition for Finite Elements

R. Morison¹ and S. Otto¹

Received November 4, 1986

The solution of finite element problems with irregular geometries on a parallel computer of the hypercube type (MIMD, distributed memory) is considered. The technique of scattering the decomposition is found to be easy to implement and to effectively load balance the computation.

KEY WORDS: Finite element methods; irregular geometries; hypercube; parallel algorithms.

1. INTRODUCTION

The numerical solution of partial differential equations (PDEs) is a computer application whose use has grown as dramatically as the increase in computational power over the last few decades. Both the ability of PDEs to model engineering designs and physical phenomena and the growth of good numerical methods for PDEs implies the need for machines that rapidly solve these problems. The usefulness of numerical PDE methods for an application is limited by the amount of time a realistic problem requires. An engineer would like quick results in order to test several variants of a design. Researchers would like to test models by running multiple simulations while varying parameters and boundary conditions. Interactive solvers of complicated two- and three-dimensional PDEs with irregular boundaries could be one of the most important scientific applications for the next generation of supercomputers.

The current stage of supercomputing has made the above possible for large two-dimensional problems and small, simplified three-dimensional problems that can afford expensive supercomputer time. It is the advent of the much less expensive parallel supercomputers that will make large scale,

¹ Caltech Concurrent Computation Group, Caltech, Pasadena, California 91125.
three-dimensional modeling possible on a routine basis. Hypercube based parallel computers, for example, first developed at Caltech (Fox, 1984, 1985; Seitz, 1985; Fox et al., 1985; Fox and Otto, 1984), are now commercially available from Intel, Ncube, Ametek, and FPS. These machines are competitive with conventional machines on a cost–performance basis and promise to eventually surpass the performance of more conventional machines. This paper discusses a method for solving a large class of numerical PDE problems on a MIMD machine with local memory and a mesh connectivity with at least periodic boundary conditions, e.g., a torus of dimension equal to that of the PDE problem. The hypercube architecture is one such machine.

Two numerical methods currently used to solve PDEs are the finite element method (FEM) (Carey and Oden, 1983; Lapidus and Pinder, 1984) and the finite difference method (FMD) (Lapidus and Pinder, 1984; Dahlquist et al., 1974). The latter is the older and more established of the two, although the finite element method is becoming the more popular owing to its ability to handle irregular meshes, complicated boundary conditions, and variable material properties. The two have an important facet in common for our purposes—they both require the solution of a large system of equations as the overriding expense. For reasons we will elaborate on below we only consider iterative methods of solution for such a system. We shall also assume that the system is linear, i.e., of the form $Ax = b$, which arises in the common case of a linear PDE. Nonlinear PDEs are usually solved via a sequence of linear subproblems (Newton iteration) and so will also use the techniques of this paper. Before explicating the method we would like to consider the memory and computational requirements of problems in two and three dimensions.

The size of the FEM and the FDM can be quantified in terms of the degree of freedom (DOF). A DOF is the value of an independent variable at some point within the domain of the problem. Each point in the domain that has one or more DOFs is called a node. The more DOFs one uses to approximate the solution, the more accurate that solution will generally be. Each DOF generates one equation in the DOFs of the problem domain. Only a few nearby DOFs are actually involved in such an equation, though. The ones that are involved in the equation generated by a DOF are defined by the stencil, e.g., see Fig. 1, of the neighboring FEM nodes or the nodes defined by the FDM differencing scheme. For an $n$ DOF approximation we thus have $n$ equations in $n$ unknowns, or a matrix problem $Ax = b$ to solve. The $n \times n$ matrix $A$ is traditionally called the system matrix and $x$ and $b$ are called the displacement and force vectors, respectively. For a finite difference problem the entries of $A$ and $b$ are defined by the differencing stencil for the PDE. In the finite element