The boundary element method exhibits a geometric parallelism which may be implemented very efficiently on a coarse-grained parallel architecture. The geometric parallelism is manifested in terms of a domain decomposition procedure. A general purpose FORTRAN Communication Harness is used to provide the equation solution phase of the method, the harness being designed to exploit the features of a transputer network.

Introduction

In many boundary element problems, where a large number of similar operations is performed, a parallel machine can be used to exploit the inherent parallelism of the algorithm. Distributed memory machines, although requiring extra programming, can provide scalable performance, at considerably lower cost than that of current vector supercomputers.

To make use of an arbitrary network of transputers, it is necessary to have software which can effect the communication of data between processors and can also schedule the data for processing. In this paper we consider the
implementation of a multiblock solver for the Boundary Element Method (BEM) on a distributed memory machine with an arbitrary number of processors and with a hardware configuration using Mushtaq’s Communication Harness [1], [2]. This problem also demonstrates the generality of the Communication Harness for topology-independent and scalable programming, and its suitability for any problem where a large number of similar operations is performed. The implementation of the Communication Harness, written in Parallel 3L FORTRAN, was initially intended for porting a multiblock, two-dimensional, Navier-Stokes aerofoil code onto a network of transputers [3], [4].

The methodology developed for the multiblock Navier-Stokes solver is used to investigate parameters affecting the efficiency of our parallel implementation of the BEM solver using geometric data partitioning i.e. multiple blocks. It is essentially the multiblock algorithm that makes the problem suitable for parallelisation using Single Program Multiple Data stream (SPMD) principles i.e. the same algorithm (BEM) is replicated to all processors in the network, each operating on different data. This is achieved by discretising the domain of computation into sub-domains, called blocks, thus creating internal boundaries between blocks. Before a block can be updated, transfer of data, called halo data, across internal boundaries is necessary. This also necessitates the communication of halo data between transputers. The Communication Harness is used to schedule blocks for processing and to effect the communication of the halo data.

The multiblock method

The multiblock method provides an improvement in the quality of the physical domain discretisation at low computational cost. The degree of inherent parallelism in the computational domain increases with the number of blocks. The multiblock algorithm leads naturally to the creation of 'tasks', each of them
associated with a single block. Each task can be executed independently of the others, and only periodically does information between the tasks have to be exchanged. On a transputer network one or several blocks per processor would be a natural approach, where halo data is communicated via message passing routines. Another important consideration is that of the memory availability. The multiblock structure allows blocks to be distributed to the network of transputers thereby reducing the number of points kept in main memory of each transputer.

We can show [5] that there is an optimum size for the blocks. Many small blocks create congestion delays and fewer large blocks create structural delays. Small blocks are further desirable for parallelism because they allow more efficient load balancing. The computational aspect is handled using the Communication Harness.

The subdomain boundary element formulation for potential problems

To illustrate the process we consider the following two-dimensional potential problem defined over the plane region $D$ bounded by the closed curve $C = C_0 + C_1$:

\begin{align}
\nabla^2 u &= 0 \text{ in } D \\
u &= u_0 \text{ on } C_0 \\
\frac{\partial u}{\partial n} &= q = q_1 \text{ on } C_1
\end{align}

The usual BEM procedure [6] yields a system of algebraic equations in the form

\[ Hu + Gg = 0 \]

which, on application of the boundary conditions, may be written in the form

\[ Ax = b \]

where $A$ is the system matrix of coefficients which depend only on the boundary geometry, $b$ is a column vector of known values from the boundary and $x$ is a vector of unknown potentials and fluxes on the boundary.
Consider two subregions \( D_1 \) and \( D_2 \) of \( D \). From eqn (2), the system of equations corresponding to \( D_1 \) is written as

\[
\begin{bmatrix} H_{ij}^1 & H_{ij}^2 \\ H_{ij}^1 & H_{ij}^2 \end{bmatrix} \begin{bmatrix} u_{ij}^1 \\ u_{ij}^2 \end{bmatrix} = \begin{bmatrix} G_{ij}^1 & G_{ij}^2 \\ G_{ij}^1 & G_{ij}^2 \end{bmatrix} \begin{bmatrix} q_{ij}^1 \\ q_{ij}^2 \end{bmatrix}
\]  

(4)

and similarly for \( D_2 \)

\[
\begin{bmatrix} H_{ij}^1 & H_{ij}^2 \\ H_{ij}^1 & H_{ij}^2 \end{bmatrix} \begin{bmatrix} u_{ij}^1 \\ u_{ij}^2 \end{bmatrix} = \begin{bmatrix} G_{ij}^1 & G_{ij}^2 \\ G_{ij}^1 & G_{ij}^2 \end{bmatrix} \begin{bmatrix} q_{ij}^1 \\ q_{ij}^2 \end{bmatrix}
\]  

(5)

where the subscript \( I \) refers to nodes on the interface between \( D_1 \) and \( D_2 \).

The compatibility conditions on the interface can be expressed in the form

\[
\begin{align*}
u_I &= u_I^1 = u_I^2 \\
q_I &= q_I^1 = -q_I^2
\end{align*}
\]  

(6)

The boundary element procedure is applied to each block in turn as if they were independent of each other, only periodically or iteratively exchanging the compatible potentials and fluxes between the common interfaces. One possible iterative approach would be to write the equations in the well-known form for each block as

\[
\begin{align*}
A^1 x^1 + H_{ij}^1 u_I &= b^1 + G_{ij}^1 q_I \\
A^2 x^2 - G_{ij}^2 q_I &= b^2 - H_{ij}^2 u_I
\end{align*}
\]  

(7)

where, for each block, \( A^1 \) and \( A^2 \) are the 'usual' system matrices, \( b^1 \) and \( b^2 \) are the vectors comprising the known boundary conditions and \( x^1 \) and \( x^2 \) are the vectors of unknown potentials and fluxes. The first equation enables \( x^1 \) and \( u_I \) to be obtained in parallel with \( x^2 \) and \( q_I \) from the second equation, the interface terms \( u_I \) and \( q_I \) being communicated between blocks as halo data at the end of each iteration. The procedure does, however, require an initial value for \( u_I \) and \( q_I \) to begin the computation.

We illustrate the algorithm using the test-case shown in Figure 1. The region is sub-divided into sixteen blocks with unit initial halo values for both
the potential and flux. Common nodes at the intersection points of either physical or halo boundary are replaced by nodes, equal to the number of intersecting sides, by a small separation distance, typically one percent of element length between successive nodes. A halo node common to more than two blocks is not permitted by the Communications Harness.

![Diagram](a) Single domain, 160 Nodes (40 nodes per side),
(b) Sub-division of (a) into 16 blocks with 160 nodes (10 nodes per block side)

Five different types of halo exchange were investigated to compare the convergence rates to the steady-state. Figure 2 shows the various types of halo exchange. The figure illustrates the nodal designations, which are designated a potential value, $u$, or a flux value, $q$. The term *alternating* halo data denotes that every alternate node, on each block side, represents either a potential or derivative value, and similarly *non-alternating* refers to the whole side of a block representing either potential or derivative values. The term *continuous* halo data refers to adjacent halo nodes on adjacent blocks both having a designation of potential or derivative, and similarly *discontinuous* refers to the adjacent halo nodes having an opposite designation *i.e.* one halo node is designated a potential value, the other a derivative value.
The multiblock boundary element method was investigated for the 16 block topology shown in Figure 1(b) employing the various forms of halo exchange possibilities illustrated in Figure 2. In Figure 3, we show the effect on convergence of each halo exchange type and we see that the non-alternating and discontinuous halo data with averaging, yields the fastest convergence rate. The figure also shows that alternating halo data causes an unstable solution profile and divergence. Clearly, alternating halo data is not suitable for the multiblock boundary element method. It is surprising to observe that continuous halo data yields a very poor convergence rate.
The total execution time depends on the total number of nodes on the boundary of a domain, in the case of the multiblock boundary element method, the total number of nodes on the boundary of the block. Some blocks will include sides that coincide with the physical boundary while the other sides of the block are conceptual or halo boundaries created solely for the purposes of the multiblock sub-division. Assuming that a fixed number of nodes on the physical boundary is desired we can minimise the increase in executable time caused by the multiblock method by minimising the total number of halo nodes on the halo boundary.

In Figure 4 we show that minimising the execution time, by reducing the number of halo nodes, does not have an adverse effect on the convergence rate. In fact it is observed that, with the exception of the case with two nodes on the halo boundary per block side, fewer halo nodes are preferred for a maximum convergence rate and a minimal execution time.
The multi-partition method

The multiblock method leads naturally to the region under consideration. However it leads to an increase in the total number of nodes with a consequent increase in computation for each phase of the boundary element method. An obvious extension of the idea is to develop a multi-partition method, illustrated in Figure 5, where the original boundary is sub-divided into multiple boundary partitions.
The system of equations corresponding to eqn (3) may be written as

$$[A_1 \ A_2 \ A_3 \ A_4]^T \mathbf{x} = [b_1 \ b_2 \ b_3 \ b_4]^T \tag{8}$$

In eqn (8) the submatrices corresponding to each boundary partition can be generated independently of the others. Similarly the recovery of the internal potentials can also be performed independently i.e. the set-up and recovery phases can be performed in parallel for each boundary partition with no inter-processor communication requirement whatsoever. Overhead due to communication will therefore occur only in the equation solution phase and current indications are that the indirect bi-conjugate gradient algorithm Bi-CGstab and the direct Gauss-Jordan methods provide solutions equivalent to their sequential counterparts.

**Conclusion**

The multiblock boundary element method has been implemented on a transputer network using the Communication Harness, which exploits the SPMD programming style, for arbitrary message-routing and task scheduling. Of the three distinct phases of the boundary element method, the equation set-up and the recovery of the internal potentials are extremely well load-balanced and provide a highly efficient parallelism. The equation solution process is the phase which does not have such a property. The solution process used is an iterative one and comprises a direct (LU decomposition) or an indirect (bi-conjugate gradient) solver in each block with updating of the interface potentials and fluxes. The choice of style for the interface node is crucial to ensure convergence.

An obvious extension to the work is to develop a totally iterative process using the multi-partition method which eliminates the need to discretise the domain and current indications are that the bi-conjugate gradient algorithm Bi-
CGstab and the Gauss-Jordan method provide stable and accurate solutions and are independent of any form of coupling i.e. the interface nodes.

References