A Transputer based Parallel Algorithm for Surface Panel Analysis

Dr. S.R. Turnock
Department. of Ship Science
University of Southampton
Southampton. SO9 5NH U.K.

SUMMARY

A surface panel method has been developed to run in parallel across variable sized square arrays of transputers. A geometric parallelism is used for both the data distribution and the algorithm. A flexible geometry definition allows complex three-dimensional surfaces and multiple body problems to be solved. Each body surface is sub-divided into quadrilateral panels. A fast parallel block-iterative solver was developed which allows rapid solution of the dense but diagonally dominant linear system of equations. The parallel performance of the surface panel code is described and the necessary scaling of number of transputers and distributed memory per transputer to obtain solutions of surface panel problems of order of 10,000 panels is given. A final section gives, as an example, the use of the code in predicting ship rudder-propeller interaction.

1. INTRODUCTION

The use of surface panel methods for modelling potential flow around marine vessels is widespread. For the hydrodynamicist they provide a valuable tool capable of reasonable prediction of body forces without extravagant use of computational time. However, for larger problems involving detailed three-dimensional surfaces and multiple body problems the required computational time still restrict their use.

In general surface panel techniques are solved using implicit techniques which require the calculation of coefficients for a dense matrix and then the solution of a large linear system of equations. Parallel algorithms are easily produced for explicit schemes however there is a need for research into the ways in which the benefits of parallel processing can be applied to implicit algorithms such as the surface panel method. The work reported is part of a research programme to investigate ship rudder-propeller interaction and further details can be found in Molland (1992 a,b).

The implementation of a lifting surface panel method to run on an array of transputers using the developed communications harness is described. A suite of procedures for carrying out the various stages of the analysis has been written and is referred to as the PALISUPAN (**PA**rallel **LI**fting **SU**rface **PA**nel) code. A geometric parallelism was used for the data distribution and the numerical formulation of PALISUPAN. The parallelism is based on equally dividing the total number of lifting surface and wake panels amongst the numbers of transputers available on a given parallel computer.

An important parameter in parallel processing is the measurement of the performance of a particular parallel algorithm on a given parallel computer. How this is quantified and

how performance is compared to that of an equivalent serial algorithm are necessary questions in determining whether transputer based parallel computers provide a cost-effective method for carrying out a particular application.

All the software was written in Occam2. The overall software design philosophy was to minimise the development time and subsequent debugging by the use of simple geometric algorithms. A structured approach making full use of the procedures and channel communications of Occam2 allowed this to be successfully carried out.

A variety of methods can be used to produce parallel algorithms to solve a lifting surface panel problem with a total of N panels using T transputers. A parallel geometric algorithm where each transputer is assigned (N/T) panels is the simplest method and is one which naturally lends itself to the solution of computational fluid dynamic problems. Also, problems with different total number of panels can be easily scaled without the need to alter the software simply assigning a different number of panels to each transputer.

2. TRANSPUTERS

The transputer is a micro-processor based integrated circuit designed as a basic building block for the construction of both large and small scale parallel computers. Associated with the transputer is Occam2: a computer language specifically developed to make full use of the parallel processing capabilities of the transputer.

Transputers are a range of high-performance VLSI(Very Large Scale Integrated) technology devices, developed by Inmos Ltd, which consist of local memory, four high speed two-way links and a micro-processor unit all mounted on a single silicon chip. The provision of high speed communication links allows transputers to be connected together to produce a parallel processing computer. There are no limits to the number of transputers which can be connected together in a network. The only restriction is in the topology of the parallel machine. Each transputer can be connected to a maximum of four. Massively parallel machines can be built up from large numbers of transputers.

Parallel computers are classed according to the number of tasks (or instructions) and number of data streams they can process simultaneously. Transputer based machines belong to the most general class of Multiple-Instruction-Multiple-Data stream (M.I.M.D.) machines. The advantage of transputer based parallel processing systems is that the same basic processing unit can be used for both small-scale and large-scale computational applications. Code can be developed on inexpensive machines with a small number of transputers and then executed on a large array of transputers.

The performance of transputer based parallel computers can be scaled if all the component transputers have identical computational and communication loads. This facility allows parallel computers which use small numbers of transputers to be used to assess the performance of large scale computations. An important proviso is that an appropriate scale of problem size is used. As an example of such a study Robinson (1990) investigated the parallelism of a commercial fluid dynamics software package ASTEC, Lonsdale (1989), which uses an implicit finite volume solution method on a finite element mesh. They concluded that transputer based parallel systems can deliver greatly increased performance and also that parallel systems allow problems to be solved which could not be tackled on

sequential machines.

3. SURFACE PANEL THEORY

In a lifting surface panel formulation the approximation of the full Navier-Stokes equation assumes that the flow is inviscid, incompressible and irrotational and satisfies Laplace's potential equation:

$$\nabla^2\phi = 0 \qquad [1]$$

A detailed description of the method and a review of its historical development is given by Hess (1990). Lamb(1932) showed that a quantity satisfying Laplace's equation can be written as an integral over the bounding surface S of a source distribution per unit area σ and a normal dipole distribution per unit area μ distributed over the S. If **v** represents the disturbance velocity field due to the bounding surface (or body) and is defined as the difference between the local velocity at a point and that due to the free-stream velocity then:

$$\underline{v} = \nabla\phi \qquad [2]$$

where ϕ is defined as the disturbance potential. This can be expressed in terms of a surface integral as:

$$\phi = \int\int_{S_B} [\frac{1}{r}\sigma + \frac{\partial}{\partial n}(\frac{1}{r})\mu]dS + \int\int_{S_W} \frac{\partial}{\partial n}(\frac{1}{r})\mu dS \qquad [3]$$

where S_B is the surface of the body and S_W a trailing wake sheet. In the expression r is the distance from the point for which the potential is being determined to the integration point on the surface and ∂/∂n is a partial derivative in the direction normal to the local surface. A dipole distribution is used to represent the wake sheet.

The conditions imposed on the disturbance potential are that:
1) the velocity potential satisfies Laplace's equation everywhere outside the body and wake;
2) the disturbance potential due to the body vanishes at infinity;
3) the normal component of velocity is zero on the body surface;
4) the Kutta-Joukowsky condition of a finite velocity at the body trailing edge is satisfied.
5) the trailing wake sheet is a stream surface with equal pressure either side.

For a steady-state solution, the wake dipole strength distribution is uniquely determined by the application of the Kutta condition at the body trailing edge. As conditions (1) and (2) are satisfied as functions of μ and σ, conditions (3) and (4) are used to determine μ and σ on the body. The Kutta condition only applies at the trailing edge and some other relationship has to be used to uniquely determine the distribution of μ and σ over the body. The numerical resolution of this non-uniqueness is referred to as the singularity mix of the lifting-surface method.

The numerical procedure used is based on that of Morino (1974) where the body surface is represented by a series of N quadrilateral panels each with an unknown but constant dipole strength per unit area. The vertices of these panels are located on the actual

surface of the body. The wake sheet is represented by M panels placed on the stream-surface from the trailing edge of the body surface. Its dipole strength per unit area is related to the difference in dipole potential at the trailing edge.
On the body surface the source strength per unit area is prescribed by satisfying the condition for zero normal velocity at the panel centroid:

$$\sigma_S = \overline{U} \cdot \overline{n} \qquad [4]$$

where **n** is the unit normal outward from the panel surface and U the specified inflow velocity at the panel centroid.

The numerical discretisation of [3] gives the potential at the centroid of panel i as:

$$\phi_i = \frac{1}{2\pi}\sum_{j-1}^{N} ((\boldsymbol{U} \cdot \boldsymbol{n_j}) S_{ij} - \phi_j D_{ij}) + \sum_{k-1}^{M} \Delta\phi_k W_{ik} \qquad [5]$$

where for panel j:S_{ij} is the source influence coefficient of a unit strength panel; D_{ij} the dipole influence coefficient; and W_{ik} the influence of the constant strength wake strip extending to infinity. As there are N independent equations corresponding to the N body surface panel centroids, [5] is closed and can be evaluated. Expressed in matrix form it becomes:

$$[\ D_{ij}\ ]\ \underline{\boldsymbol{\phi}} + [\ W_{ik}\ ]\ \underline{\Delta\boldsymbol{\phi}} = [\ S_{ij}\ ]\ (\boldsymbol{U} \cdot \boldsymbol{n}) \qquad [6]$$

The original Morino trailing edge Kutta condition, specified the wake strength $\Delta\phi$ as the difference in trailing edge panel potential, the matrix expression [6] can then be directly solved to give the vector of dipole potentials $\boldsymbol{\phi}$. Numerical differentiation of dipole potential along the body surface allows the surface velocity and hence pressures on the surface to be evaluated.

The methods used for the evaluation of the individual influence coefficient elements of the matrices S_{ij}, D_{ij} and W_{ik} are based on those described by Newman (1986).

For most three-dimensional geometries there is a cross-flow at the trailing edge and this requires an iterative approach to determine the correct wake strength by ensuring that the pressure difference Δp between the upper and lower panels at the trailing edge is zero. As Δp is primarily a function of the local trailing edge wake sheet strength $\Delta\phi$ an iterative Newton-Raphson approach is used (Lee 1987) to determine the wake strength for the point of zero pressure difference at the trailing edge. Once the solution vector $\boldsymbol{\phi}$ is obtained this is used to calculate Δp at the trailing edge. The correction vector of known strength is multiplied by the wake strip influence coefficient matrix W_{ik} and applied to the right hand side of the equation. This modifies the original matrix expression to:

$$[\ D_{ij} + W_{ik}\ ]\ \boldsymbol{\phi} = [\ S_{ij}\ ]\ \boldsymbol{U} \cdot \boldsymbol{n}_j - [\ W_{ik}\ ] \left(\frac{d\Delta\phi}{d\Delta p}\Delta p \right)^k \qquad [7]$$

where the wake strength for k=0 is taken to be the difference in potential between the trailing edge panels. The process is repeated until the pressure loading at the trailing edge has been removed.

The numerical solution gives a result vector which specifies a dipole strength at the centre of each panel. This corresponds to the potential ϕ on the surface of the body. To obtain practical engineering information from this surface potential distribution a numerical differentiation has to be carried out. The differentiation gives the disturbance velocity tangential to the panel surface. Once this velocity is determined the surface pressure and hence total body force can be evaluated.

4. SURFACE GEOMETRY DEFINITION

The ease with which the geometry of a lifting-surface problem can be distributed across the array of transputers will determine the usefulness of the fluid dynamic code.

In this work one of the principal features is the investigation of the performance of a lifting-surface code on a transputer network. Therefore, it is necessary to have a simple means of scaling the overall problem size by altering the number of panels used to define a lifting-surface. The decision was made to generate the actual panel vertex coordinates within the program but to use a pre-processing file to define the number of bodies and their individual geometry. This allows a problem to be scaled by using the internal panel generator to produce a different number of panels for the same overall body geometry.

A variety of means are available for defining a three-dimensional surface (or body). A ship hull form is conventionally defined using a series of lines which lie in parallel planes. These lines, whether waterlines, buttocklines or transverse sections, are themselves defined in terms of an ordered set of coordinates. A mathematical relationship is then used to generate the curved lines between the coordinates and hence specify a three-dimensional surface.

A useful means of relating the line coordinates to the curve passing through them is that of a parametric cubic spline procedure. A spline approximation is defined as a piece-wise polynomial approximation to a curve. Each segment of a line is represented as a polynomial. For a cubic spline at the end of each segment the gradient and curvature of the polynomial expression are matched to the adjoining polynomial expressions. This results in a curve made up of a series of cubic lines defined in terms of a single parameter. Defining the value of the parameter uniquely defines the value of a point on the line. The parameter is the arc distance along the original curve and is usually approximated as the straight-line distance between points used to define the line. For the purposes of this work a surface definition using parametric cubic spline procedures provides an accurate approximation to a three-dimensional surface. The end condition used throughout was that of zero curvature.

Each individual body (or part body) is defined in the same manner as that of a ship hull form; as an ordered series of lines with each line containing an ordered set of three-dimensional points. For a closed lifting body such as a rudder or wing, a wake sheet will be connected to the trailing edge and it is therefore sensible to start and finish each body definition line at the trailing edge. The lines are ordered so that the normal vector to a panel always faces out into the exterior flow field.

5. PARALLEL DATA DISTRIBUTION

The parametric cubic spline procedure is used to sub-divide the body surface into the required number of surface panels. Each guest process (transputer) is sent the complete problem geometry. From the overall geometry definition, each guest process generates its allocation of surface panels. As the parametric cubic splines used on each process are based on identical information the panel boundary points between transputers are the same.

Each surface panel is assigned an unique absolute identity number to allow the dipole, source and wake influence coefficient matrices to be correctly assembled. A unique body identity number is used to define the correct relationship between individual panels on a body which is necessary for the calculation of surface potential derivatives. By relating the panel identity number to the individual transputer number in the array, the process of setting up and solving the coefficient matrices is made independent of a particular geometry. This allows multiple body problems to be easily solved.

In addition to the body panel data the velocity field and wake sheet data need to be sub-divided amongst the guest transputers. An identical method to that for the body is used for the wake sheet where the number of wake strips on each transputer are those corresponding to the body trailing-edges. The means by which memory storage requirements are minimised was as follows.

In conventional Fortran or Pascal programs body geometry data are stored in multi-dimensional variable arrays. The maximum number of elements in each dimension is specified at compile-time. This results in the need to recompile the program every time it is wished to alter, for example the balance between the maximum number of spanwise and chordwise panels. Also, for multiple body problems, each body either requires its own assigned array or to be assigned as a part of an overall array. This sort of programming problem results in cumbersome detail with inherent costs in programming and debugging time. A more sensible strategy is to write an explicit indexing procedure and store all the information as a single-dimension array of real numbers. The index procedure allows the required coordinate information to be directly accessed. The maximum size of the array can then be related to the memory available on a transputer and a far more flexible approach to multiple body problems becomes possible.

The panel geometry definition is used throughout the lifting-surface panel algorithm. Therefore, all the component processes have to be able to use the information. In Fortran such information is transferred between sub-routines using common-block type definitions. Similarly in Occam2 and Pascal the information can be accessed if the individual procedures are defined within the scope of the array definition. Again, the larger the amount of array information the more cumbersome the programming detail. Fortunately in Occam2 the whole problem can be circumvented by defining "live" storage. A storage array such as that for the panel vertices is written as a process running in parallel with the numeric algorithm. Information is accessed through communication along channels in and out of the memory store process. Considerable numeric processing can be carried out within the memory store process running in parallel, thereby helping to ensure satisfactory 'hiding' of inter-transputer communications. The design of the "live" memory store uses a one-dimensional array with a simple index procedure for access, the details of which are given in Turnock (1993).

The use of such a structure for memory storage has resulted in many programming advantages. The development time of individual component processes of the lifting-surface program is greatly reduced as many repetitive programming tasks can be included within the Store procedure. Another advantage is that data storage requirements for the component processes can be kept to a minimum. The store structure allowed different variants to be developed all using the same protocol but with different functions occurring within each store.

6. PARALLEL ALGORITHM

The data distribution and the numerical algorithm used geometric parallelism. The overall structure of the program was the same as that used for a previously developed parallel Euler solver, Turnock 1990. At the outermost level, the program consists of the harness and guest (or host) process running in parallel. The use of the Harness software removes the need rewrite inter-transputer communication procedures for different applications and greatly speeds program development. The host process is mounted on the transputer connected to the outside world (screen, keyboard, hard disk etc.) and controls the guest process transputers. Each guest process operates independently and communicates with any other guest or the host process via the harness. For a geometric algorithm the same executable code is mounted on each transputer.

The scaling of a lifting-surface problem of a given total number of panels onto the available number of transputers is of interest. The program was developed to allow the performance of the program and its component processes to be measured for variable sized square arrays of transputers and for different total number of body panels across fixed sized square arrays of transputers.

The total number of panels which are geometrically distributed across an array is limited by the amount of external memory assigned to each transputer. To maximise the total number of panels available the memory requirements of the component processes were kept to a minimum. In some places this resulted in extra numeric processing but it was considered that the greater the number of panels per transputer the greater the use of the code. A common restraint on panel method performance is a limit on the number of panels which can be processed at once. This block size severely limits performance as a large amount of slow data transfer operations from storage to active memory are required for processing each block. By removing the need for this type of operation a considerable time saving is achieved at the small cost of extra numeric processing. Once the surface panel geometries have been generated by each guest process the three stages of analysis are carried out as described in the following sections.

6.1. Calculation of Influence Coefficient Matrix

The setting up of the lifting surface system of equations requires the calculation of the source and dipole influence coefficients S_{ij}, D_{ij} between every panel (i=1 to N) and panel centroid (j=1 to N), and the wake strip influence coefficient W_{kj} between every wake strip (k=1 to NWs) and panel centroid (j=1 to N). As each guest process stores a fraction (N/T) of the total number of panels considerable communication around the transputer network is required to allow the elements of the influence coefficient matrices S_{ij}, D_{ij}, W_{kj} to be

evaluated and stored.

The means by which information is transferred determines the efficiency of the setting-up process. The final results of the calculation are three matrices. The source and dipole matrices are of size N by N, where N is the total number of panels used to define the Lifting surface. The wake influence matrix has dimensions NWs by N where NWs is the number of wake strips.

The boundary condition vector **[U.n]** at each panel centroid is known and hence each panel's source strength. So, rather than storing the elements of matrix $[S_{ij}]$ it is better to carry out the matrix-vector multiplication as part of the setting up process. This halves the required matrix memory storage.

To store the right hand side vector **RHS**, the dipole matrix $[D_{ij}]$ and wake strip matrix $[W_{kj}]$ three memory store variants were used. The dipole matrix is stored in a general square matrix routine which evenly distributes the matrix elements across the square array of transputers. Similarly, an equivalent general vector procedure is used for the vector **RHS** and a rectangular matrix for $[W_{kj}]$.

At this stage it is worth noting that, if there are N panels distributed across a network of T transputers, each transputer has information for (N/T) panels. The dipole matrix is of size N^2 and each transputer stores a sub-matrix of dimension (N^2/T). This indicates that as the dimension of the square array increases, for a fixed amount of memory per transputer, the maximum number of panels located on each transputer will decrease. This is due to the matrix store progressively occupying more of the transputers memory. The memory requirement (in bytes) for a panel, wake and matrix store can be approximated by:

$$M = 4 \times \left[3 \times 1.5 \times \left[\frac{N}{T} + 1 \right]^2 + 2 \times \left[\frac{N^2}{T} \right] \right] \qquad [8]$$

where each real number requires 4 bytes and all the memory requirements but that for the panel and dipole store arrays are ignored. This provides a good estimate of the number of panels which can be solved on a given transputer array. For instance the Ship Science Transputer System, of dimension N=2 and 1 Mbyte of memory per transputer, can solve at least a 400 panel problem. A machine with 8 Mbyte per transputer could solve a 10,000 panel problem if it were of dimension 11. A surface panel problem containing this number of panels is the order of the largest scale of lifting-surface problems currently solved on supercomputers. The Meiko Computing Surface available for use in this work has 8MByte per transputer and with a sixteen transputer array can solve a 3800 panel problem.

The efficiency of the setting up process has to be as high as possible to ensure satisfactory operation of the lifting surface program on large dimension arrays solving large-scale panel problems. This means the strategy chosen has to keep the volume and amount of data communication to a minimum and ensure that every guest process always has information to process.

The even distribution of panels amongst the transputers helps to minimise communication. Each transputer has to calculate the self-influence of its own panels and

their corresponding centroids. To calculate the influence of panels located on other transputers to its own panel centroids it is more sensible to communicate the panel centroid to the other transputers (three real numbers) rather than to receive the other transputers panel vertices (twelve real numbers).

For a particular centroid it is necessary to calculate the source and dipole influence coefficients of every panel and of every wake strip. This is ideal for a parallel algorithm. Simultaneously, the setting up process is: (1) calculating panel influence coefficients; (2) calculating wake strip influence coefficients; (3) receiving panel centroids; (4) communicating the resultant matrix and vector elements to their final store destination. The multiplicity of parallel tasks ensures that each guest process has minimal delays while waiting to carry out a numerical operation.

At the outer level the setting-up process (CalculateIC) runs in parallel with the various live-memory stores and the harness. To minimise data storage the panel geometric coefficients are recalculated every time a panel is used and by processing a package of centroids with each panel the number of times the geometric coefficients are recalculated is reduced. For a given number of surface and wake panels on a fixed array of transputers the

Figure 1 Performance of the CalculateIC process on a four transputer array

total amount of communication remains constant. Therefore, the least efficient operation of the CalculateIC process occurs when there is a minimum of numeric processing per panel to be carried out. In Figure 1, which plots the number of panels distributed across four transputers against the time to calculate all the influence coefficients, lines are shown for three test cases of a rudder without reflection plane, a rudder with reflection plane and a four-bladed propeller. The amount of numeric processing per panel increases as the number of image panels increases and therefore the overall time increases. However, if the time to run a 200 panel problem on four transputers and one transputer is compared speed-ups of 2.96 (η_c=73.9%), 2.99 (η_c=74.8%) and 3.28 (η_c=82%) are obtained for the three cases, where the code efficiency η_c is defined in this case as the ratio of the speed-up obtained to that of the number of transputers used. This demonstrates the increased efficiency with additional numeric processing per panel. The relatively small increase in efficiency is due to the additional operations to reflect/rotate panels and generate their geometry coefficients.

Overall the calculate influence coefficient process exhibits a quadratic increase in processing time with the number of panels. The knee in the curve occurs when the panel centroid is sent as two packets rather than one which results in extra processing as each panel geometry data is generated twice as many times.

6.2. Iterative Solution Of Linear Systems Of Equations

Efficient solution of the linear system of equations generated by the Calculate influence process is essential. The basic system to be solved is:

$$[\, M \,]\, \underline{\phi} = \underline{R} \qquad [9]$$

where vector **R** is known , matrix [M] is dense but diagonally dominant, and vector ϕ is unknown.

The choice of technique for solving a system of linear equations revolves around whether an iterative approach can converge more rapidly to solution than a direct method takes to explicitly solve the whole system. In general a dense full matrix, such as that generated by a lifting-surface, is more likely to favour an iterative approach. This is especially true because of the leading diagonal dominance of the system. This dominance arises from the self-influence of each dipole panel.

How the various algorithms are made to run in parallel and whether this influences the choice between iterative and direct methods needed to be investigated. A range of iterative methods were developed and their performance compared with a parallel direct matrix inversion using Gaussian elimination described in Turnock (1993).

i) Jacobi-Iterative Scheme

An iterative scheme uses an initial guess for the unknown vector ϕ to generate a better approximation to the solution. The process is repeated until the solution has converged to a given level of accuracy. The simplest possible method for diagonally dominant matrices is to use the main diagonal elements to calculate the correction. This process is known as the Jacobi method.

The Jacobi correction $\Delta\phi$ to the original estimate of vector ϕ can be written as:

$$R^* = [\, M \,]\, \phi^{k-1} \qquad [10]$$

$$\Delta\phi_{jj} = \frac{1}{M_{jj}} \left(R_j - R^*_j \right) \qquad [11]$$

where M_{jj} is the j^{th} leading diagonal element. The k^{th} approximation then becomes:

$$\underline{\phi}^k = \underline{\Delta\phi}^k + \underline{\phi}^{k-1} \qquad [12]$$

A convergence check is carried out on correction vector $\Delta\phi$. The maximum absolute value is sent to the host process which either allows the guest process to proceed or, if all the maxima are below a cut-off criteria, finishes the calculation. The cut-off criteria used was that the absolute maximum change in an element is less than $1x10^{-5}$.

The matrix-vector multiply process is the only one which requires communication during each iteration. As a result each iteration is fast. However, as only one element is used in the updating process a large number of iterations are required. An advantage of the scheme is that all the processes are working all the time in generating the next iteration. The Gauss-Seidel scheme uses the same technique for updating, only each element update is carried out sequentially, based on using the latest values for each element. The interdependence for element updating means that a parallel single-element Gauss-Seidel method is inefficient.

For the Jacobi scheme the initial approximation is usually taken as the zero vector. However, for the lifting-surface method with the iterative Kutta condition it is better to use the final solution for the next series of iterations as most values of potential only change by a small amount. Generally this halved the number of iterations required for convergence.

ii) Single-Block Iterative Scheme

The drawback of the Jacobi scheme of using only one element in the updating process suggests the use of a block of information from the original matrix. A block size equal to the number of panels on each guest transputer can be easily accommodated.

The first stage is to transfer the leading diagonal matrices to the guest process where their self-influence panels are stored. These sub-matrices are then directly inverted using Gaussian elimination. The iterative process can then commence. It is identical to the Jacobi except that instead of the Vector Divide stage the sub-matrix inverse is used to multiply the vector difference. where T is the transputer number and $[M]_{TT}^{-1}$ the sub-matrix inverse.

$$\underline{\Delta\Phi}^k_T = [\, M \,]^{-1}_{TT} \left(\underline{R}_T - \left([\, M \,] \Phi^{k-1} \right)_T \right) \qquad [13]$$

The implementation of the scheme is identical in all other respects to the Jacobi. A drawback to this scheme is that if the number of panels per transputer gets too large the inversion of the sub-matrix will take a long time. As in the case of the Jacobi scheme for each iteration the only communication is during the matrix-vector multiply stage. Also, once the sub-matrices have been passed to their destination, the direct inversion is an independent process.

iii) Multi-Block Jacobi Scheme

To restrict the maximum block size the dimension of the sub-matrices was made variable between 1 (Jacobi - single element) and that for 1 block per transputer (single-block). The implementation is the same as for the single-block scheme except that each guest process has a number of blocks sent to it. Each block is then inverted and used in the iteration process to update its portion of the $\Delta\phi$ vector. The ability to change the block size allows the iterative procedure to be tuned to obtain minimum iteration time for a given lifting-surface problem.

Figure 2 Variation in convergence time with block size for 400 panels

Figure 2 plots convergence time against block size for a 400 panel problem. A pronounced minimum is found for a block size of 25 panels. This corresponded to the number of panels in each chordwise strip used to define the geometry. This is not unexpected as the panel potential strengths are dominated by other panels in the same chordwise strip and so only using the influence of these panels in updating should give the quickest convergence. Clark (1985) also found this was the best block size for his accelerated convergence scheme.

6.3. Calculation of Body Forces, Velocity Fields, and Wake Adaption

Once the surface potential has been found the overall body force and moments can be calculated, velocity fields determined and if necessary the prescribed wake adapted to follow the local flow. All these processes can be carried out independently. A farm algorithm where the host process sends out identical packets of data to each guest process to process independently is ideal. For example to calculate the three-dimensional velocity field away from a body for a series of positions a packet of nodes is sent to all guest processes. Each guest calculates for every node in the packet the disturbance velocity field due to its body surface and wake sheet panels. The packet of velocity information is returned to the host process where it is combined with the packets returned from the other guest process to give the total velocity for each of the series of points. As it is a farm algorithm the performance of the velocity field process is proportional to the number of panels, and the number of points, and inversely proportional to the total number of guest transputers.

7. PERFORMANCE OF CODE

Implicit algorithms require communication and the amount of communication traffic depends on the number of transputers used to solve a given problem. The estimation of the performance of an implicit algorithm on different sized arrays of transputers requires expressions for the time to carry out the communication transfer as well as for the numeric calculation. For an explicit algorithm, as represented by the Euler solver described in Turnock(1990), the ratio of calculation to communication is constant for a given number of nodes per transputer and hence performance can be scaled from timing measurements made on a small array of transputers. However, for an implicit algorithm, how the communication time scales with array size has to be known for an estimation process based on the performance of a small array of transputers.

As the performance of the harness across an array of transputers is known (Turnock 1993) an estimate can be made of the total communication time for any given array of transputers. In general, there will be one or more transputers in the network whose messages (to and from) have to travel over the number of links equal to the dimension of the array. This transputer will have the maximum data transmission time and determine the code efficiency.

For an implicit algorithm, code efficiency scales with the size of the transputer array, where code efficiency is defined as the ratio of the time spent carrying out the calculation to that of the total time. The total time includes the time spent communicating information. Figure 3 plots code efficiency against the total number of panels. These are code efficiencies for a 4 transputer array with a diameter of 2 links as compared to the performance obtained

Figure 3 Code-efficiency of lifting-surface component processes on a four transputer array

for an identical number of panels on a single guest transputer. The memory size of 1MByte per transputer on the Ship Science Transputer System only allowed performance on up to 200 panels to be assessed. Lines are shown for influence coefficient setting up, solution of linear system of equations(block Jacobi and Gaussian elimination), and velocity field calculation. Also shown are lines for the two fundamental processes: matrix transposition and matrix-vector multiplication. For all the processes η_c increases with the number of panels. The velocity field calculation performance is constant as is expected for a farm algorithm. The efficiency of 91% reflects the overheads associated with generating the test nodes on the host transputer. Therefore, the efficiency of the wake adaption process would be expected to be higher.

The efficiency of the CalculateIC process is only 75% and reflects the inherent complexity of the process. It also suggests that potential improvements could be expected from further development of this process.

The Gaussian elimination process, for an implicit algorithm, shows a rapid increase

in efficiency as the number of panels increase as does the Matrix-Vector and Block-Jacobi processes to a 70%-80% efficiency. Even the Transpose operation has a 50% efficiency for a 200 panel problem.

Overall, even for a relatively small panel problem (200) the code efficiency of the component processes of the lifting-surface panel analysis are all in the range of 70% to 90%. This efficiency would be expected to increase for larger sized panel problems.

8. SHIP RUDDER-PROPELLER INTERACTION

The Interaction Velocity Field method for solving rudder-propeller interaction is described in Turnock 1993. It is an iterative process: for each iteration the flow is solved individually for a rudder and propeller geometry. The performance of PALISUPAN is problem specific. It is dependent on the total number of panels, panel distribution, wake shape, imposed inflow velocity field, presence of a reflection plane or rotational images, variables such as rudder incidence or propeller r.p.m. and so on. As an illustration the overall performance is given for a typical rudder geometry (400 panel rudder at -0.4^{o}

Figure 4 Comparison of time to carry out a single rudder-propeller iteration cycle for different advance ratio

incidence with representative propeller velocity field) and propeller geometry (400 panel 4 bladed propeller and hub).

Figure 4 shows the total time (in minutes) for one complete rudder-propeller interaction cycle for three advance ratio. The total time is broken down into its four components. The propeller calculation (both solution and velocity field) dominates the overall time. The increase in time with reducing advance ratio indicates the increased three-dimensionalilty of the rudder/propeller inflow at higher propeller thrust loading (low J).

It is worth noting the fact that although surface panel methods are considered to only have a moderate computational requirement, this work has demonstrated that there are large possible savings in time through the use of parallel processing. In this case, if the velocity field calculations are assumed to work at 90% efficiency, the propeller at 80% and the rudder calculation at 75% the overall process efficiency is 83%. This is very good performance for what should be a worst case test for parallel processing.

The cumulative tally of calls to the Newman panel process is 7.36 million which gives an average time per call of 9 clock ticks on a 4 transputer array. The time to carry out the J=0.51 iteration cycle estimated for a single transputer (with enough memory) would be 226 minutes, measured on a 4 transputer array as 68 minutes, estimated for a 9 transputer array at 30 minutes and estimated for a 16 transputer array as 17 minutes.

9. CONCLUSION

A complex lifting-surface panel algorithm has been successfully implemented to run across variably sized square arrays of transputers. The ease with which the parallel algorithm has been developed has highlighted the use of the geometric approach for both the algorithm and data distribution. Development time was also reduced through the separation of the CFD algorithm from the Harness.

Significant memory saving and reduced programming time has resulted from the development of a live-memory datastore process for storing large arrays of information.

The high code efficiencies (70% to 90%) obtained for the PALISUPAN component processes indicate that the harness process performs well for the large communications requirements of implicit algorithms and that the use of geometric data distribution and geometric algorithms allows implicit methods to perform as well as explicit algorithms and to benefit from the use of parallel processors.

The overall performance of PALISUPAN in solving rudder-propeller interaction illustrates the benefits of parallel processing. On a 16 transputer array the time per complete iteration cycle would be of the order of 20 minutes, a reasonable timescale for the investigation of detailed design information.

A large scale panel problem such as the analysis of a complete aircraft with flaps with the order of 10,000 panels could be solved using 121 transputers with 8MBytes per transputer. The estimated time to solve such a system, with the order of 200 million calls to Newman panel, would be of the order of 60 minutes. This performance is comparable

with that of existing super-computers for a fraction of the investment and running costs.

The transputer used in this work is now a 5 year old design. The rapid developments in similar devices will result in even better performance and greater cost saving in the future.

ACKNOWLEDGEMENTS

The work described in this report covers part of a research project funded by the S.E.R.C./M.O.D. through the Marine Technology Directorate Ltd. under research grant Ref No GR/E/65289.

REFERENCES

Clark, R.W., 1985, "A new iterative matrix solution procedure for three-dimensional panel methods", Proceedings of AIAA 23rd Aerospace Science Meeting, Reno, Nevada, Jan. 1985.

Hess. J.L., 1990, "Panel Methods in Computational Fluid Dynamics", Annual review of fluid mechanics. Vol 22. pp.255-274.

Lamb, H., 1932, "Hydrodynamics", Cambridge University Press, sixth edition.

Lee, J-T, 1987, "A potential based method for the analysis of marine propellers in steady flow", Ph.D. thesis, M.I.T. Dept. of Ocean Engineering.

Lonsdale, R.D., & Webster, R., 1989, "The application of finite volume methods for the modelling three-dimensional flow on an unstructured mesh", Proceedings of the 6th international conference on numerical methods in laminar and turbulent flow, Swansea, July 1989.

Molland A.F., & Turnock, S.R., 1992 a, "Wind tunnel investigation of the influence of propeller loading on ship rudder performance. Royal Institution of Naval Architects, (Written discussion, Paper W3).

Molland, A.F., & Turnock, S.R., 1992 b, "The prediction of ship rudder performance characteristics in the presence of a propeller", presented at the 2nd International conference on Manoeuvring and Control of Marine Craft, Southampton, U.K. 15th-17th July.

Morino, L., & Kuo, C-C, 1974, " Subsonic Potential aerodynamics for Complex Configurations: A general theory", A.I.A.A. Journal, Vol 12., No. 2.

Newman, J.N., 1986, "Distribution of sources and normal dipoles over a quadrilateral panel". Journal of Engineering Mathematics,Vol 20., pp113-126.

Robinson, G. & Lonsdale, R., 1990, " Fluid Dynamics in parallel using an unstructured mesh", in "Parallel processing in engineering applications" ed. Ackley, R.A., Computational Mechanics Institute.

Turnock, S.R., 1990, Parallel Implementation of an Explicit Finite Volume Euler Solver on

an Array of Transputers. pp. 1557-1568, Proc. of International Congress of Aeronautical Sciences, Stockholm , Sweden, September.

Turnock, S.R., 1993, "Prediction of ship rudder-propeller interaction using parallel computations and wind tunnel measurements", Ph.D. Thesis, University of Southampton.