INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **39**:979–999 (DOI: 10.1002/fld.318)

Parallelization of a vorticity formulation for the analysis of incompressible viscous fluid flows

Mary J. Brown and Marc S. Ingber*

Department of Mechanical Engineering, University of New Mexico, Albuquerque, NM 87131, U.S.A.

SUMMARY

A parallel computer implementation of a vorticity formulation for the analysis of incompressible viscous fluid flow problems is presented. The vorticity formulation involves a three-step process, two kinematic steps followed by a kinetic step. The first kinematic step determines vortex sheet strengths along the boundary of the domain from a Galerkin implementation of the generalized Helmholtz decomposition. The vortex sheet strengths are related to the vorticity flux boundary conditions. The second kinematic step determines the interior velocity field from the regular form of the generalized Helmholtz decomposition. The third kinetic step solves the vorticity equation using a Galerkin finite element method with boundary conditions determined in the first step and velocities determined in the second step. The accuracy of the numerical algorithm is demonstrated through the driven-cavity problem and the 2-D cylinder in a free-stream problem, which represent both internal and external flows. Each of the three steps requires a unique parallelization effort, which are evaluated in terms of parallel efficiency.

KEY WORDS: vorticity method; parallel finite element method; parallel boundary element method; generalized Helmholtz decomposition

1. INTRODUCTION

The use of vorticity formulations for the analysis of incompressible viscous fluid flows has several advantages over primitive-variable formulations. Some of these advantages include a reduction of the number of equations to be solved through the elimination of the pressure variable, identical satisfaction of the compressibility constraint and the continuity equation, and an implicitly higher-order approximation of the velocity components. The main disadvantage of vorticity formulations is the fact that the boundary conditions are typically given in terms of prescribed velocity rather than prescribed vorticity or vorticity flux.

Several approaches have been developed over the years to derive vorticity boundary conditions from the prescribed velocity boundary conditions and the vorticity within the domain.

*Correspondence to: M. S. Ingber, Department of Mechanical Engineering, University of New Mexico, Albuquerque, NM 87131, U.S.A.

Received 9 November 2000
Revised 15 December 2001

Some of these approaches include streamfunction–vorticity methods [1–6], velocity–vorticity Cauchy methods [7], vorticity–velocity Poisson equation methods [8], Biot–Savart methods [9], and generalized Helmholtz decomposition methods [10–15].

The generalized Helmholtz decomposition (GHD) is a kinematic statement which relates the velocity at a point within the domain to the domain vorticity and the imposed velocity boundary conditions. Wu [13] uses the GHD to determine values of the boundary vorticity which then yields Dirichlet boundary conditions for his solution of the vorticity equation. Wu states that, for a two-dimensional problem, either component of the GHD can be used to determine the boundary vorticity, since the tangential and normal components are equivalent. Morino [14] discusses the use of the GHD to determine boundary conditions for viscous fluid flows. He states that the process of vorticity generation can be interpreted by satisfying the normal boundary condition at discrete time steps followed by infinitesimally small intervals when diffusion of the vortex sheets occurs. El-Refaee [16] uses the GHD for the solution of the interior velocity using the tangential component of the boundary velocity. He then uses the vorticity transport equation and Green's theorem to solve for the boundary vorticities followed by a successive under-relaxation iterative scheme to solve for the interior vorticities. In the current research, a Galerkin form of the GHD [17] is used to determine vortex sheet strengths which are then related to the vorticity flux at the boundary. The advantage of the current approach is that the velocity boundary conditions are far better satisfied using a Galerkin implementation of the GHD compared to using the more traditional point collocation methods.

A parallel computer implementation of a vorticity formulation to analyse incompressible viscous fluid flow is presented in this paper. The formulation is divided into three distinct steps, two kinematic steps and one kinetic step. The first kinematic step is the determination of Neumann boundary conditions for the vorticity equation using a Galerkin form of the GHD. The second kinematic step is the determination of interior velocities using the regular form of the GHD. The third kinetic step is the solution of the vorticity equation using a Galerkin finite element method (FEM). Each of the three steps of the algorithm requires a different approach for the parallelization effort. However, common to all three steps, a distributed memory architecture and a single program, multiple data (SPMD) programming paradigm using message passing interface (MPI) as the interprocessor communication protocol is assumed.

The primary limiting factor in the serial code version of the algorithm is the size of the arrays used to store the velocity integral arrays in step two. These large arrays benefit greatly from parallelization, both from the reduction in size of the array because of the distributed memory architecture as well as the distribution of the work load in creating these arrays and in performing the matrix–vector multiplication to determine the interior velocity field. This step is parallelized first to take advantage of the huge memory savings of dividing these large arrays among processors as well as the relative simplicity of the parallel algorithm. Since the evaluation of the velocity integrals at each interior node is independent, the integral evaluations can simply be distributed among processors and the results stored on the same processor. This evaluation is done only once, outside of the time stepping loop. Within the time loop, matrix–vector multiplications are performed on each processor to determine a portion of the interior velocity field. The only communication required is a broadcast of the interior vorticity field and a gather of the nodal velocities into an array.

The parallelization effort for the first and third steps requires a more involved strategy. The Galerkin GHD used to determine the boundary vortex sheets for the first step is essentially a boundary integral equation (BIE). Several approaches have been developed for parallel BIEs [18–21]. The assembly phase of most BIE formulations consists of a triple-nested do-loop. For Galerkin methods, the outer and middle loops are over boundary elements while the inner loop is over Gauss points. Distributing the outer loop elements over processors is equivalent to common row-wrapping techniques. Distributing the middle loop elements over processors is equivalent to column wrapping. Distributing the inner loop Gauss points over processors results in a fine grain parallelization which is not particularly well suited to distributed memory computer architectures [22]. However, none of these strategies is optimal when using a direct solver [23]. A direct solver is used in this research because, relative to the finite element discretized equation set, the boundary element discretized equation set is relatively small and results in a fully populated linear system allowing for efficient use of the direct solver. Further, the decomposition can be performed once, outside the time loop, and hence, only forward and backward substitutions are required within the time loop. The optimal strategy for the BIE using a direct solver uses a block-cyclic data distribution [24–26]. In this approach, portions of both the outer and intermediate loop are distributed over processors resulting in rectangular submatrices being assigned to processors.

Several parallel FEMs have also been developed over the past several years. Nodal assembly [27] divides the nodes among the processors, but requires a certain amount of bookkeeping to provide the connectivity information. Element-by-element solution algorithms [28] divide the elements among the processors, but the global matrix is never assembled. Matrix–vector operations are performed on elemental matrices and the global vector result is obtained by summing over all elements. These require some redundancy in that nodal data may be needed on more than one processor. Total-summed-row or 'fully summed equations' approaches [29] sum the elemental coefficient matrices at the beginning of the solve rather than summing the vector product. For this particular problem, since the nodal data for this problem are already required on each processor for the calculation of the interior velocities, no benefit would be gained by using the nodal assembly method. The total-summed-row method is chosen in this research to reduce the memory required since a sparse summed matrix is more memory efficient than individual element matrices. This is accomplished by assigning individual rows of the discretized finite element equations among processors. Each row of these equations is calculated independently of all other rows. In the current implementation, the equations are solved using a generalized minimal residual (GMRES) iterative method. During the solution phase of the algorithm using the GMRES, the only communication that is required is the gathering of the residual vectors.

Two benchmark problems are considered to show the accuracy of the present formulation. The first benchmark is the driven-cavity problem and the second benchmark is the cylinder in a free-stream problem. Results generated by the current parallel algorithm are compared to accepted standards available in the literature. A complete analysis is also performed to determine the effectiveness of the parallel strategies for each step of the numerical algorithm in terms of scalability and memory savings as well as the algorithm as a whole.

2. MATHEMATICAL FORMULATION

The vorticity form of the Navier–Stokes equations for an incompressible flow in two dimensions is given by

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = \nu \nabla^2 \boldsymbol{\omega} \tag{1}$$

where $\mathbf{u}$ is the velocity field, $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ is the vorticity field, t is time and ν the kinematic fluid viscosity. In the course of solving Equation (1), the velocity field, $\mathbf{u}$, must be determined from the vorticity field, $\boldsymbol{\omega}$, and the creation of vorticity on the boundary must be determined from the velocity boundary conditions. In the present formulation, determining both the interior velocity field and the creation of vorticity on the boundary are accomplished in a unified manner using the GHD.

The GHD for an incompressible fluid in two dimensions is given by

$$\alpha(\mathbf{x})\mathbf{u}(\mathbf{x}) = \int_\Omega \frac{\boldsymbol{\omega}(\mathbf{y}) \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Omega(\mathbf{y}) + \int_\Gamma \frac{[\mathbf{u}(\mathbf{y}) \times \mathbf{n}(\mathbf{y})] \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) - \int_\Gamma \frac{[\mathbf{u}(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})]\mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) \tag{2}$$

where $\mathbf{n}$ is the unit normal vector on the boundary (pointing away from the fluid), Ω represents the two-dimensional domain, Γ is the boundary of Ω and $\mathbf{r}$ is the distance between $\mathbf{x}$ and $\mathbf{y}$. The coefficient α is a function of the location of the field point $\mathbf{x}$. For field points outside the domain, $\alpha = 0$; for field points in the interior the domain, $\alpha = 2\pi$; for field points on smooth portions of the boundary, $\alpha = \pi$; and for field points at corners, α can be related to a local internal angle.

For field points $\mathbf{x}$ on the boundary, Equation (2) can be augmented to allow for a slip velocity by including the vortex sheet of strength $\boldsymbol{\gamma}$ as follows [30]:

$$\alpha(\mathbf{x})[\mathbf{u}(\mathbf{x}) - \boldsymbol{\gamma}(\mathbf{x}) \times \mathbf{n}(\mathbf{x})] = \int_\Omega \frac{\boldsymbol{\omega}(y) \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Omega(\mathbf{y}) + \int_\Gamma \frac{[(\mathbf{u}(\mathbf{y}) - \boldsymbol{\gamma}(\mathbf{y}) \times \mathbf{n}(\mathbf{y})) \times \mathbf{n}(\mathbf{y})] \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) - \int_\Gamma \frac{[(\mathbf{u}(\mathbf{y}) - \boldsymbol{\gamma}(\mathbf{y}) \times \mathbf{n}(\mathbf{y})) \cdot \mathbf{n}(\mathbf{y})]\mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) \tag{3}$$

The solution of Equation (3) yields the vortex sheet strengths, $\boldsymbol{\gamma}$, representing the creation of vorticity during a given time step.

The vorticity equation (Equation (1)) represents the kinetics of the vorticity formulation while the modified GHD (Equation (3)) represents the kinematics of the formulation. The numerical formulation as described in the next section is divided into three steps. In the first step, a Galerkin implementation of the modified GHD is solved to determine the vortex sheet

strengths. The vortex sheet strengths represent the vorticity creation during the previous time step and, to first order in time, are related to the vorticity flux by [31, 17]

$$\frac{\partial \boldsymbol{\omega}}{\partial n} = \frac{\boldsymbol{\gamma}}{\nu \Delta t} \tag{4}$$

where Δt is the discrete time step. In the second step, interior velocities are calculated using the regular form of the modified GHD. Thus, both the interior velocities and vortex sheet strengths are calculated in a unified manner using the GHD. In the third step, the vorticity equation (Equation (1)) is solved using a FEM. However, after an explicit time step in the vorticity equation, the modified GHD is no longer satisfied without taking into account newly formed vorticity on the boundary. Kinematic compatibility is re-established by going back to step 1 and solving for the newly formed vortex sheet strengths. This three step process is interesting from an implementation point of view since each of the three steps requires a significantly different strategy for parallelization. The parallelization strategies for each step is discussed in Section 5.

3. NUMERICAL FORMULATION

Details of the numerical formulation have been given previously by Ingber and Kempka [17]. For completeness and because the parallelization effort is dictated by the numerical formulation, a brief discussion of the numerical formulation is presented below.

3.1. Galerkin approximation of the GHD

For convenience, define a new vector $\mathbf{t} = \mathbf{u} - \boldsymbol{\gamma} \times \mathbf{n}$. Hence, the GHD (Equation (3)) becomes

$$\alpha(\mathbf{x})\mathbf{t}(\mathbf{x}) = \int_{\Omega} \frac{\boldsymbol{\omega}(y) \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Omega(\mathbf{y}) + \int_{\Gamma} \frac{[\mathbf{t}(\mathbf{y}) \times \mathbf{n}(\mathbf{y})] \times \mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) - \int_{\Gamma} \frac{[\mathbf{t}(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})]\mathbf{r}(\mathbf{x},\mathbf{y})}{r^2(\mathbf{x},\mathbf{y})}\, \mathrm{d}\Gamma(\mathbf{y}) \tag{5}$$

Discretization is achieved by subdividing the domain Ω into finite elements and the boundary of the domain Γ into boundary elements. Within the eth finite element, the jth component of $\boldsymbol{\omega}$ is approximated as

$$\omega_j^e(\mathbf{y}) = \sum_{l=1}^{4} \omega_{lj}^e S_l(\mathbf{y}) \tag{6}$$

where ω_{lj}^e represents the value of the jth component of $\boldsymbol{\omega}$ at the lth node within the eth finite element and S_l represents the bilinear Lagrangian shape function associated with the finite element. Similarly, within the eth boundary element, the jth component of $\mathbf{t}$ is approximated as

$$t_j^e(\mathbf{y}) = \sum_{l=1}^{2} t_{lj}^e N_l(\mathbf{y}) \tag{7}$$

where, in this case, t_{lj}^e represents the value of the jth component of $\mathbf{t}$ at the lth node within the eth boundary element and N_1 represents the linear Lagrangian shape function associated with the boundary element. Substituting Equations (6) and (7) into Equation (5), the discretized form of the GHD can be written using indicial notation as

$$\alpha(\mathbf{x})t_j(\mathbf{x}) = \sum_{e=1}^{\mathrm{NFE}} \int_{\Omega_e} \frac{e_{ijk}\omega_{lj}S_l(\mathbf{y})d_k}{d_r d_r}\,\mathrm{d}\Omega + \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_e} \frac{e_{imp}e_{mjk}t_{lj}^e N_1(\mathbf{y})n_k d_p}{d_r d_r}\,\mathrm{d}\Gamma$$
$$- \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_e} \frac{t_{lj}^e N_1(\mathbf{y})n_j d_i}{d_r d_r}\,\mathrm{d}\Gamma \tag{8}$$

where e_{ijk} is the unit alternating tensor, NFE represents the number of finite elements, NBE represents the number of boundary elements, and $d_i = x_i - y_i$, where $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$.

Using the properties of the unit alternating tensor, this equation can be re-written as

$$\alpha(\mathbf{x})t_j(\mathbf{x}) = \sum_{g=1}^{\mathrm{NFE}} \int_{\Omega_g} \frac{e_{ijk}\omega_{lj}^g S_l d_k}{d_r d_r}\,\mathrm{d}\Omega$$
$$+ \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_e} \frac{t_{lk}^e N_1 d_k n_i - t_{li}^e N_1 d_k n_k - t_{lk}^e N_1 d_i n_k}{d_r d_r}\,\mathrm{d}\Gamma \tag{9}$$

Using the identities [17]

$$\int_{\Gamma} \frac{d_2 n_1 - d_1 n_2}{d_r d_r}\,\mathrm{d}\Gamma = 0 \tag{10}$$

and

$$\alpha(\mathbf{x}) = -\int_{\Gamma} \frac{d_k n_k}{d_r d_r}\,\mathrm{d}\Gamma \tag{11}$$

the left-hand side of Equation (9) can be incorporated into the right-hand side as shown below

$$0 = \sum_{g=1}^{\mathrm{NFE}} \int_{\Omega_g} \frac{e_{ijk}\omega_{lj}^g S_l(\mathbf{y})d_k}{d_r d_r}\,\mathrm{d}\Omega$$
$$+ \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_e} \frac{t_{lk}^e N_1(\mathbf{y})(d_k n_i - d_i n_k) - (t_{li}^e N_1(\mathbf{y}) - t_i(\mathbf{x}))d_k n_k}{d_r d_r}\,\mathrm{d}\Gamma \tag{12}$$

This formulation has the advantage of not having to evaluate $\alpha(\mathbf{x})$ explicitly, as well as eliminating the Cauchy principal value integral on the left-hand side of Equation (9).

Now to obtain a Galerkin approximation, Equation (12) is multiplied by the shape functions $N_m(x)$ and integrated over the boundary Γ. Assuming that $N_m(x)$ has support within the fth boundary element and, within that element

$$t_k(\mathbf{x})|_{\Gamma_f} = t_{lk}^f N_l(\mathbf{x})$$

the discretized Galerkin approximation for the GHD is given by

$$\begin{aligned} 0 = &\sum_{g=1}^{\mathrm{NFE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Omega_g} \frac{e_{ijk}\omega_{lj}^g S_l(\mathbf{y}) d_k}{d_r d_r} \, \mathrm{d}\Omega \\ &+ \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Gamma_e} \frac{t_{lk}^e N_l(\mathbf{y})(d_k n_i - d_i n_k)}{d_r d_r} \, \mathrm{d}\Gamma \\ &- \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Gamma_e} \frac{[t_{li}^e N_l(\mathbf{y}) - t_{li}^f N_l(\mathbf{x})] d_k n_k}{d_r d_r} \, \mathrm{d}\Gamma \end{aligned} \tag{13}$$

The discretized Galerkin form of the GHD represents a vector equation for the unknown out-of-plane components of the vortex sheet strengths, $\boldsymbol{\gamma}$, which are embedded in the vector $\mathbf{t}$. It has been shown previously [17] that the normal component of Equation (13) yields a rank deficient system of linear equations. Hence, the current formulation requires applying the tangential component of Equation (13) as given by

$$\begin{aligned} 0 = &\sum_{g=1}^{\mathrm{NFE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Omega_g} \frac{e_{ijk}\tau_i\omega_{lj}^g S_l(\mathbf{y}) d_k}{d_r d_r} \, \mathrm{d}\Omega \\ &+ \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Gamma_e} \frac{t_{lk}^e \tau_i N_l(\mathbf{y})(d_k n_i - d_i n_k)}{d_r d_r} \, \mathrm{d}\Gamma \\ &- \sum_{e=1}^{\mathrm{NBE}} \int_{\Gamma_f} N_m(\mathbf{x}) \int_{\Gamma_e} \frac{[t_{li}^e \tau_i N_l(\mathbf{y}) - t_{li}^f \tau_i N_l(\mathbf{x})] d_k n_k}{d_r d_r} \, \mathrm{d}\Gamma \end{aligned} \tag{14}$$

where τ_i represents the components of the unit tangential vector to the boundary.

Recalling that the vector $\mathbf{t}$ contains both known values of the boundary velocity $\mathbf{u}$ and unknown values of the vortex sheet strength $\boldsymbol{\gamma}$, Equation (14) can be re-written in matrix form as

$$[A]\{\gamma\}^{\mathrm{T}} = \{f\} \tag{15}$$

Upon solution of Equation (15), the unknown values of the vortex sheet strengths are inserted into Equation (4) to determine Neumann boundary conditions for the vorticity equation.

3.2. Galerkin FEM solution of the vorticity equation

The Galerkin FEM used to solve the vorticity equation is outlined in this subsection. Multiplying the 2-D vorticity equation (Equation (1)) by the weighting function, w, and integrating over the domain yields

$$\int_\Omega w\frac{\partial\omega}{\partial t}\,\mathrm{d}A = -\int_\Omega u_x w\frac{\partial\omega}{\partial x}\,\mathrm{d}A - \int_\Omega u_y w\frac{\partial\omega}{\partial y}\,\mathrm{d}A - \int_\Omega\left[-\nu w\frac{\partial^2\omega}{\partial x^2} - \nu w\frac{\partial^2\omega}{\partial y^2}\right]\mathrm{d}A \tag{16}$$

where u_x and u_y are the components of the velocity vector **u**. Integrating the second-order terms by parts (applying Green's theorem), the weak form of the vorticity equation is written as

$$\int_\Omega w\frac{\partial\omega}{\partial t}\,\mathrm{d}A + \int_\Omega \nu\left(\frac{\partial\omega}{\partial x}\frac{\partial w}{\partial x} + \frac{\partial\omega}{\partial y}\frac{\partial w}{\partial y}\right)\mathrm{d}A + \int_\Omega u_x w\frac{\partial\omega}{\partial x} + u_y w\frac{\partial\omega}{\partial y}\,\mathrm{d}A = \int_\Gamma w\nu q_n\,\mathrm{d}s \tag{17}$$

where the flux q_n is related to the vortex sheet strengths through Equation (4), that is

$$q_n = \frac{\partial\omega}{\partial n} = \frac{\gamma}{\nu\Delta t} \tag{18}$$

The weak form of the vorticity equation is discretized by subdividing the domain Ω into finite elements and subdividing the boundary Γ into boundary elements. Using isoparametric bilinear Lagrangian interpolation for the finite elements and linear interpolation for the boundary elements, the weak form of the vorticity equation can be written in discrete form as

$$\sum_{i=1}^{\mathrm{nbe}} w_i^e\int_{\Gamma_e} N_i^e\frac{N_k^e}{\Delta t}\,\mathrm{d}\Gamma\gamma_k^e = \sum_{e=1}^{\mathrm{nfe}} w_i^e\int_{\Omega^e} S_iS_j\,\mathrm{d}\Omega\frac{\mathrm{d}\omega_j^e}{\mathrm{d}t} + \sum_{e=1}^{\mathrm{nfe}}\left(w_i^e\nu\int_{\Omega_e}\frac{\partial S_i}{\partial x}\frac{\partial S_j}{\partial x} + \frac{\partial S_i}{\partial y}\frac{\partial S_j}{\partial y}\,\mathrm{d}\Omega\right)\omega_j^e$$
$$+\sum_{e=1}^{\mathrm{nfe}}\left(w_i^e\int_{\Omega_e} S_i\frac{\partial S_j}{\partial x}u_{xk}^e S_k + S_i\frac{\partial S_j}{\partial y}u_{yk}^e S_k\right)\mathrm{d}\Omega\,\omega_j^e \tag{19}$$

where nfe is the number of finite elements, nbe is the number of boundary elements, w_i^e, ω_i^e, u_{xi}^e, u_{yi}^e represent the value of w, ω, u_x, and u_y, respectively, at the ith node within the eth finite element, S_i represents the bilinear finite element shape function, γ_i^e represents the value of γ at the ith node within the eth boundary element and N_i represents the linear boundary element shape function.

For convenience, the element capacitance matrices, element stiffness matrices and element load vectors are defined by

$$(C^e)_{ij} = \int_{\Omega_e} S_i^e S_j^e\,\mathrm{d}\Omega \tag{20}$$

$$(K_x^e)_{ij} = \nu\int_{\Omega_e}\frac{\partial S_i^e}{\partial x}\frac{\partial S_j^e}{\partial x}\,\mathrm{d}\Omega \tag{21}$$

$$(K_y^e)_{ij} = \nu\int_{\Omega_e}\frac{\partial S_i^e}{\partial y}\frac{\partial S_j^e}{\partial y}\,\mathrm{d}\Omega \tag{22}$$

$$(K_u^e)_{ij} = \sum_{k=1}^{4} u_{xk}^e \int_{\Omega_e} S_i^e \frac{\partial S_j^e}{\partial x} S_k^e \, \mathrm{d}\Omega \tag{23}$$

$$(K_v^e)_{ij} = \sum_{k=1}^{4} u_{yk}^e \int_{\Omega_e} S_i^e \frac{\partial S_j^e}{\partial y} S_k^e \, \mathrm{d}\Omega \tag{24}$$

$$(F^e)_i = \frac{1}{\Delta t} \gamma_k^e \int_{\Gamma_e} N_i^e N_j^e \, \mathrm{d}\Gamma \tag{25}$$

The discretized weak form can now be written in the following convenient form:

$$\sum_{e=1}^{\mathrm{nfe}} w_i^e (C^e)_{ij} \dot{\omega}_j^e + \sum_{e=1}^{\mathrm{nfe}} w_i^e \{(K_x^e)_{ij} + (K_y^e)_{ij} + (K_u^e)_{ij} + (K_v^e)_{ij}\} \omega_j^e = \sum_{e=1}^{\mathrm{nbe}} w_i^e (F^e)_i \tag{26}$$

After assembly and dividing through by the Galerkin vector $\{w_i\}$, the assembled finite element equations become

$$[K_x + K_y + K_u + K_v]\{\omega\} + [C]\{\dot{\omega}\} = \{F\} \tag{27}$$

The discretized equation set (Equation (27)) is inherently non-linear since the matrices K_u and K_v contain the unknown velocity field components. The velocity components in these matrices are evaluated using Equation (3) in step 2 of the algorithm. Time is discretized using an Euler explicit method resulting in a first-order accurate method in time.

In the current implementation of the numerical algorithm, both the discretized FEM and GHD equations are assembled outside of the time loop as well as the LU decomposition of the GHD equations. Also performed outside the time loop is the numerical quadrature used for evaluating the interior velocities. Hence, within the time loop, the majority of calculation is matrix–vector multiplication and back substitution.

4. BENCHMARK PROBLEMS

Two benchmark problems are considered in this section to show the capabilities of the numerical formulation. The first benchmark considers interior flow for an impulsively started driven cavity. The second benchmark considers exterior flow about a cylinder in a uniform stream.

4.1. Driven cavity flow—internal flow

The driven-cavity flow computations are performed in the unit square in which the top surface is impulsively given a velocity of 1.0 at time $t = 0$. The initial velocities and vorticities are zero. The kinematic viscosity is chosen to be $v = 0.01$ resulting in a Reynolds number of 100. The time step is chosen to be $\Delta t = 0.001$. Grid converged results are obtained with a uniform 41×41 mesh.

Streamline and vorticity contours are shown in Figure 1 at time $t = 10$ s. The system has essentially reached steady-state conditions at this time. The streamline and vorticity contours are qualitatively similar to the vorticity–streamfunction multigrid finite difference results of

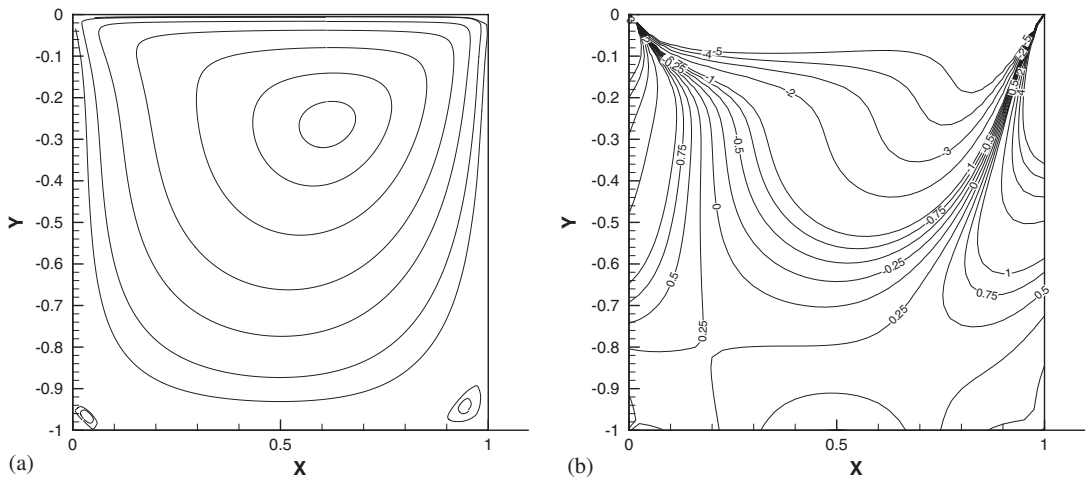


Figure 1. Flow in a driven cavity at $\Re = 100$: (a) steady-state streamline pattern; (b) steady-state vorticity contour pattern.

Table I. Comparison of primary and secondary vortex data between current vorticity solution and primitive variable FDM solution of Ghia *et al.* [12].

	Current results	Multigrid FDM results
(x, y) co-ordinates of primary vortex	(0.6165, 0.737)	(0.6172, 0.7344)
Length of bottom left vortex on lower wall	0.0822	0.0781
Height of bottom left vortex along side wall	0.0818	0.0781
Length of bottom right vortex on lower wall	0.1360	0.1328
Height of bottom right vortex along side wall	0.1546	0.1484

Ghia *et al.* [32]. Ghia's result were generated using a 129×129 stretched grid. A comparison of the centre of the primary vortex and the extent of the secondary vortices is shown in Table I. As seen in the table, the current results compare very favourably to the FDM results.

A comparison of the results generated using the current vorticity formulation and Ghia *et al*. multigrid method for the horizontal velocities along the vertical bisector and vertical velocities along the horizontal bisector of the cavity is shown in Figure 2. The agreement is again seen to be very good. The current results were generated by running the calculation through the transient from initial rest conditions whereas the FDM results were obtain using a steady-state analysis.

4.2. Flow around a cylinder—external flow

A free stream of unit velocity is impulsively started at time $t = 0$ for the benchmark problem of external flow about the circular cylinder. The diameter of the cylinder is chosen to be 1 so that the Reynolds number is given by the inverse of the fluid viscosity. Initial conditions for the vorticity are set to zero.

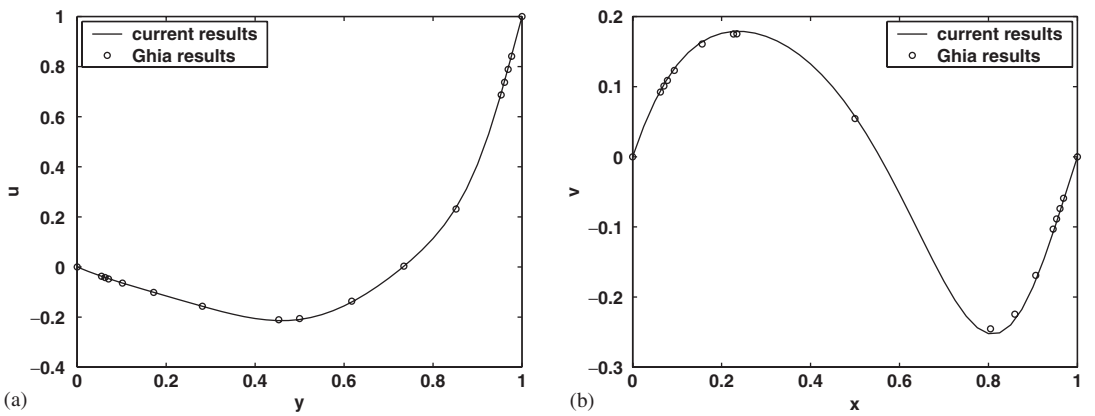


Figure 2. Steady-state results for (a) the u-component of velocity along the vertical cavity bisector and (b) the v-component of velocity along the horizontal cavity bisector.

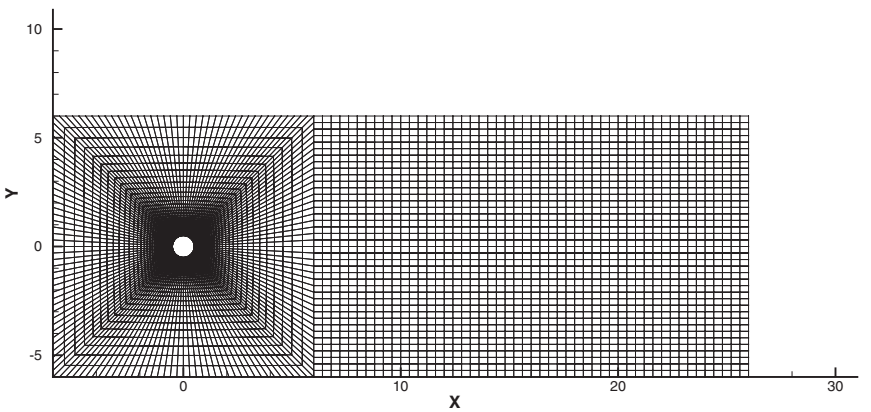


Figure 3. Finite element grid for flow about the cylinder benchmark problem.

A typical finite element grid is shown in Figure 3. The cylinder is approximated by 160 boundary elements. Lines emanate radially from the boundary element nodes until they intersect a square box with sides equal to six diameters. The finite elements are seen to be graded so that smaller elements are placed near the surface of the cylinder to be able to provide adequate resolution for the large vorticity gradients in this region. A uniform mesh is attached downstream of the square box. The total number of finite elements is 8080 and the number of finite element nodes is 8302. Although the serial version of the vorticity code was adequate to resolve the previous benchmark problem, it was not possible to run the external benchmark with current grid resolution on a single processor because the memory requirements were too large.

Vorticity fringe plots are shown in Plate 1 from time $t = 59.7$ to 66.7 at 1 s intervals for flow around the cylinder at a Reynolds number of $\Re = 100$. The vortex shedding pattern is qualitatively similar to those found in standard references such as Inoue and Yamazaki [33].

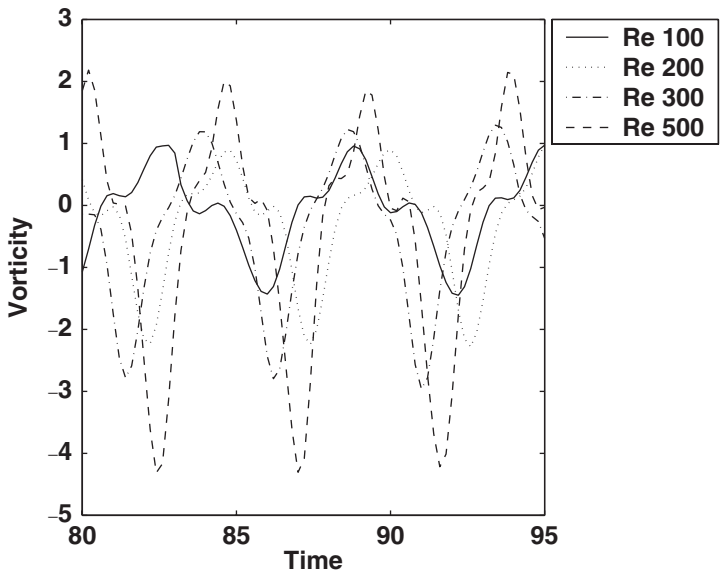


Figure 4. Calculated vorticity at the point $x = 6.4$, $y = 0.3$ in the wake of the cylinder. (The centre of the cylinder is placed at the origin of the co-ordinate system.)

The shedding frequency as characterized by the Strouhal number is approximately 0.159 which corresponds to a period of approximately 6.3s. The entire cycle of a positive vortex being shed from the lower rear portion of the cylinder and a negative vortex being shed from the upper rear portion of the cylinder is seen in the sequence of vorticity fringe plots. It is interesting to note the areas of strong vorticity attached to the rear of the cylinder which are actually opposite in sign to the vorticity being shed from that side of the cylinder.

A more quantitative measure of the accuracy of the current vorticity formulation is to compare the measured Strouhal number with those available through experiment and other numerical analyses. The Strouhal number is defined by $St = \omega d/V$ where ω is the vortex shedding frequency, d is the cylinder diameter and V is the free stream velocity. Although it is possible to obtain an approximate Strouhal number from the vorticity fringe plots shown in Plate 1, an easier approach is to choose a position in the wake of the cylinder and plot the vorticity at that point as a function of time. This plot is shown in Figure 4. Because the cylinder diameter and free stream velocity are both unity, the Strouhal number is equal to the frequency, ω, which is the inverse of the time period. The periodic motion of the flow is easily observable in Figure 4 and the periods for the four Reynolds numbers shown is easy to measure.

A plot of current numerical results for the Strouhal number as a function of the Reynolds number is shown in Figure 5. The current values are in good agreement with previous two-dimensional numerical results [34, 35]. Also shown in the figure are two curve fits of experimental data. Williamson [36] provided a curve fit for Reynolds numbers less than approximately 200 and Roshko [37] provided a curve fit for Reynolds numbers over approximately 250. It is seen in the figure that the current numerical results match the Williamson curve fit

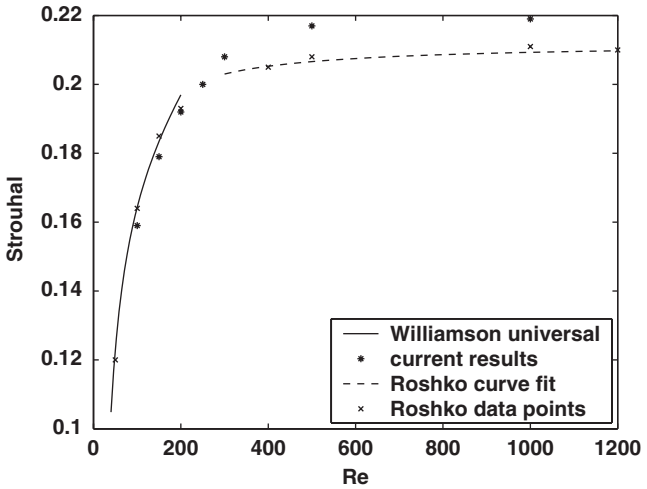


Figure 5. The Strouhal number as a function of the Reynolds number for flow about the cylinder.

quite well for $\Re < 200$ but are above the Roshko curve fit for $\Re > 250$. However, as discussed by Vorobieff and Ecke [38], there is a wake transition that occurs between Reynolds numbers of 200 and 300 associated with stretching of vortex lines for the three-dimensional experiments. This transition obviously cannot occur in a two-dimensional simulation. Vorobieff and Ecke were able to perform two-dimensional physical experiments using a soap film apparatus which also showed the two-dimensional wakes have a higher Strouhal number for $\Re > 200$ compared to three-dimensional wakes.

5. PARALLELIZATION

The main reason for the parallelization of the vorticity formulation is to increase the capabilities of the code, both by increasing the size of the problem that can be analysed, thus increasing the flexibility and usefulness of the code, and by decreasing the overall run time. The current implementation of the vorticity formulation became memory bound before it became CPU bound. For large problems, these memory limitations can be conveniently remedied by a distributed memory parallel machine.

The parallelization of the current formulation is completed in three steps with each step corresponding directly to one of the three steps of the algorithmic formulation. That is, the three steps include (1) parallelization of the Galerkin GHD to determine the vortex sheet strengths; (2) parallelization of the velocity integrals to determine the interior velocity fields and (3) parallelization of the Galerkin FEM to update the vorticity field in time.

In the following three subsections, the parallel strategy for each step is presented. In particular, the data distribution, load balancing, work performed inside and outside the time loop and message passing required by each step is discussed. The parallel program paradigm used is the single program, multiple data (SPMD) model with MPI for parallel communication.

5.1. *Step* 1, *parallelization of the Galerkin GHD*

The choice of parallel solver dictated the data distribution and matrix assembly associated with the Galerkin GHD. ScaLAPACK is used as the dense parallel matrix solver replacing the LU decomposition used in the serial code. ScaLAPACK was chosen primarily because of the fact that it uses a block cyclic data distribution which minimizes communication during the solution phase of the algorithm [23]. Further, ScaLAPACK was available on all machines used in the testing and timing.

The processors can be considered to be arranged on a 2-D array, and each processor calculates only a section of the dense matrix. ScaLAPACK allows the user to specify the size of the blocks in the block-cyclic data distribution which is most efficient for the hardware. Using too small of a block size increases the communication overhead, while too large a block size can exceed hardware cache limits and be inefficient. The block size was chosen to be the number of boundary elements divided by the number of processors in a row of the 2-D array. This means that no wrapping occurs. For the particular solver chosen within ScaLAPACK, there is a restriction that requires the 2-D processor array to be square, (e.g. 2×2 or 4×4). This requires that the submatrices assigned to each processor also be square. The matrix is assembled and decomposed in parallel outside of the time loop, so the only work performed inside the time loop is the updating of the right-hand side vector, and the back-substitution to determine the updated values of the vortex sheet strengths.

The only communication required during the assembly phase are two row-gathers to incorporate the left-hand side of the GHD into the right-hand side and to regularize the Cauchy principle value integral. However, significant communication is required of the parallel decomposition routine used in ScaLAPACK. Within the time loop, additional broadcasts are required to obtain updated values of the vorticity to evaluate the GHD, but overall storage requirements on a single processor are reduced because of the distribution of data among the processors. Some of ScaLAPACK's intrinsic functions are used to facilitate the book-keeping involved with the 2-D processor array as well as some PBLAS routines (parallel, basic linear algebra subroutines).

5.2. *Step* 2, *parallelization of the velocity integrals*

The limiting factor in the serial code is the size of the arrays generated to evaluate the velocity integrals. Although these arrays are calculated outside of the time loop, the largest array contains $8 * \mathrm{NP}^2$ elements where NP is the number of nodes contained in the finite element grid. This caused the serial code to become memory bound for fine FEM discretizations. An alternate approach would be to evaluate the velocity integrals inside the time loop which would alleviate the storage problem. However, most likely, the resulting code would then quickly become CPU bound. Obviously, the large arrays resulting from the velocity integral evaluations could benefit greatly from parallelization, both from the reduction in size of the array stored in memory on each processor of a distributed memory architecture and from the distribution of the work load in creating these arrays.

The parallelization of this step is the most straightforward since each evaluation of the velocity integral at a given field point (finite element node) is independent of all the other field points. Hence, the finite element nodes are simply evenly distributed among the processors. Outside of the time loop, the integral arrays are determined using Gaussian quadrature. Inside the time loop, the velocities are determined on the local processors by performing a simple

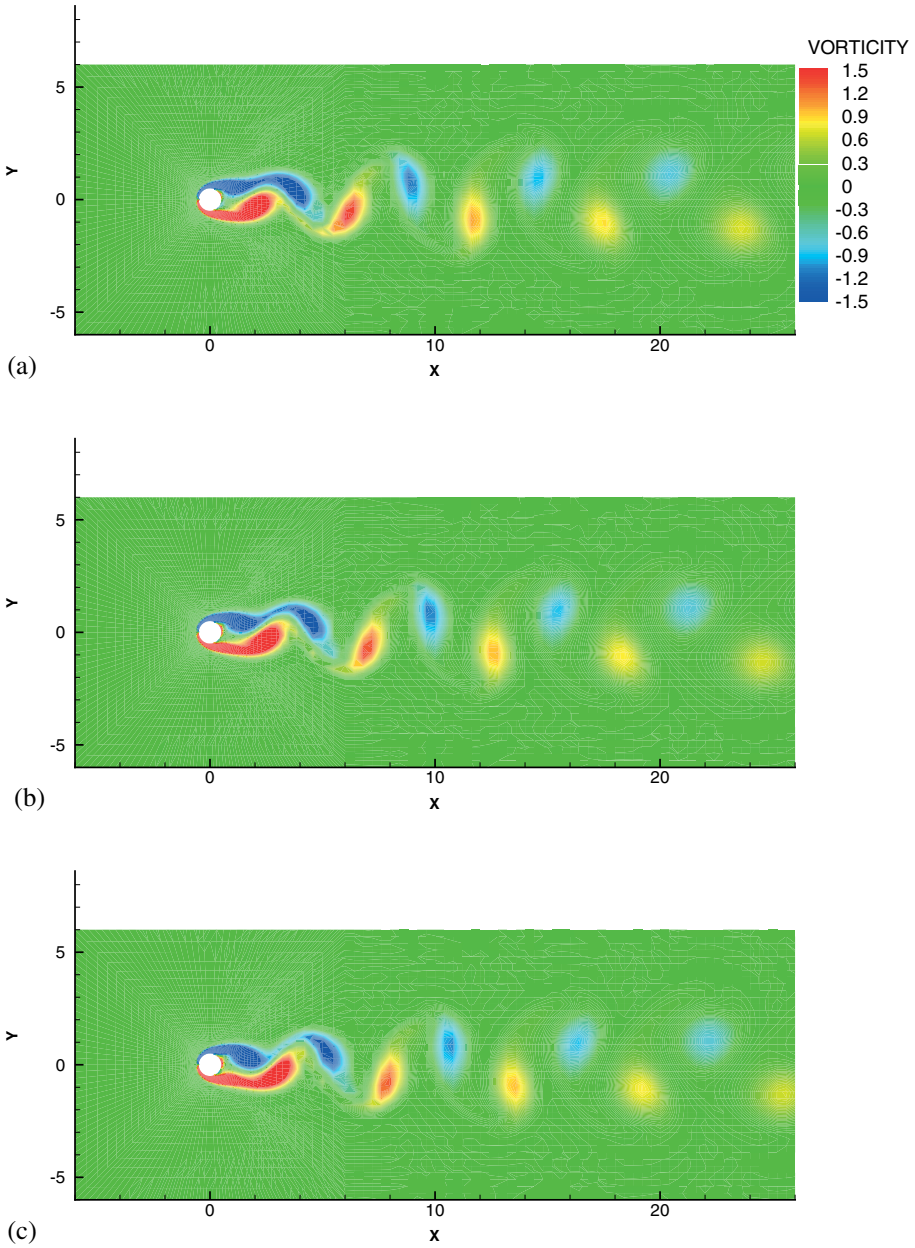
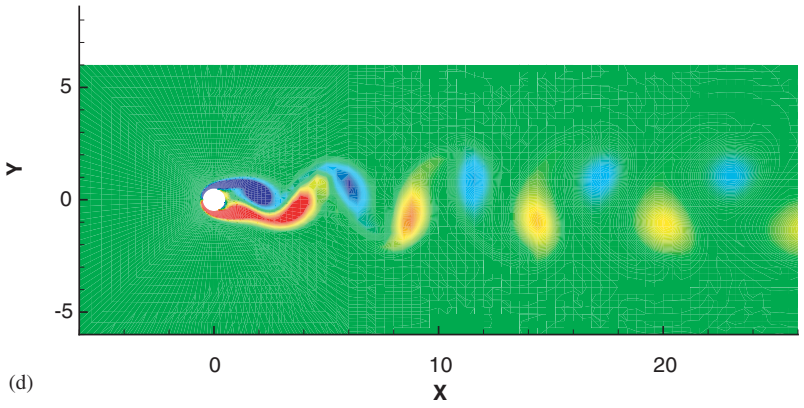
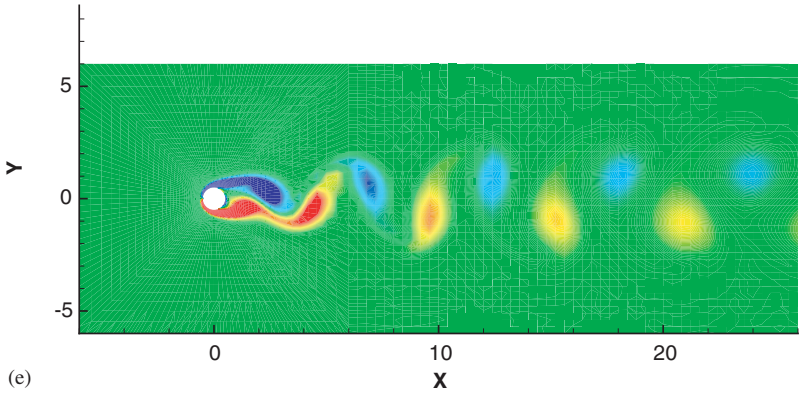


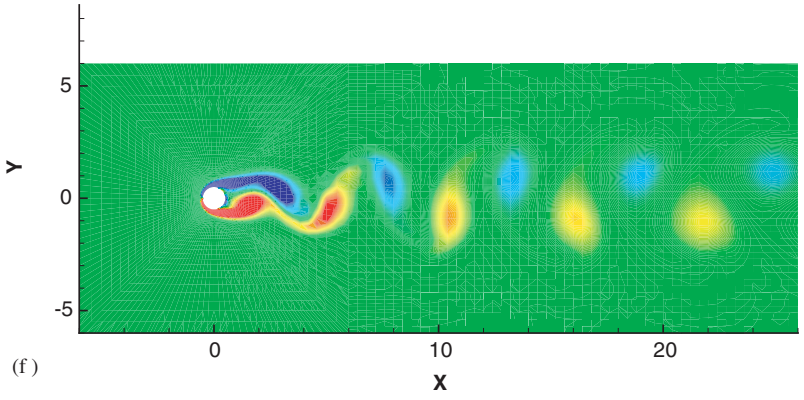
Plate 1. Vorticity contour fringe plot for flow around the cylinder at a Reynolds number of $\Re$=100, (a) $t = 59.7$, (b) $t = 60.7$, (c) $t = 61.7$, (d) $t = 62.7$, (e) $t = 63.7$, (f) $t = 64.7$, (g) $t = 65.7$ and (h) $t = 66.7$.



(d)

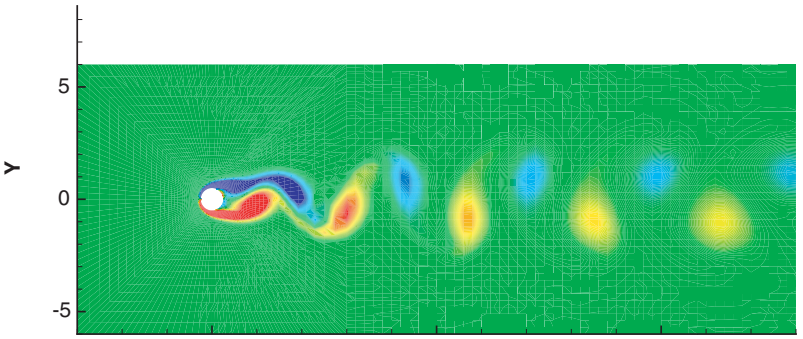


(e)

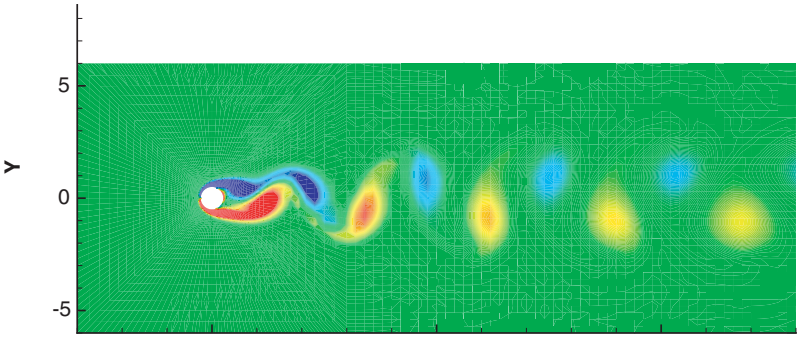


(f)

Plate 1. *Continued*



(g)



(h)

Plate 1. *Continued*

matrix–vector multiplication. The communication required in this step consists of a broadcast of the updated vortex sheet strengths from step 1 and updated vorticities from step 3 and a gather of the velocities for use in step 3 (the kinetic step).

5.3. *Step* 3, *parallelization of the finite element section*

A row-wrap data distribution is chosen for the parallelization of the kinetic step using the FEM. The parallel strategy chosen for this phase is the total-summed-row approach. Typically, this strategy is based on domain partitioning in which nodal data and connectivity for both the interior elements and border elements required to generate a complete row of the discretized matrix equation reside on a given processor [29]. However, in the current implementation, since the nodal data are required for the calculation of the interior velocities in step 1, this information is broadcast to all processors outside of the time loop after it is read in from the input file. Hence, no domain partitioning is necessary and rows in the matrix equation can simply be assigned evenly to processors.

Part of the equation assembly can be performed outside of the time loop. However, to assemble the $[K_u]$ and $[K_v]$ (see, Equation (27)) portions of the stiffness matrix, updated values of the velocity filed determined in step 2 are required. Inside the time loop, the completion of the stiffness matrix is performed by matrix–vector multiplication. The load vector is also determined inside the time loop again by performing a matrix–vector multiplication. Since the assembly of each row of the matrix equation is independent, there is no communication in the assembly phase of the algorithm.

The iterative equation solver PIM [39] was chosen to solve the sparse linear system using a restarted GMRES method with Jacobi preconditioning. The data are stored in CSR (compressed sparse row) format and only non-zero matrix values are stored. PIM requires a user-defined parallel matrix–vector multiplication scheme. The majority of the communication required during the solution phase of this step is a gather of the residual vector.

6. PARALLEL PERFORMANCE

The parallel performance of the vorticity code is evaluated using the driven-cavity problem in a unit square for various levels of discretization. Outside the time loop, timers were placed at the beginning and end of the velocity integral evaluation, the assembly of the GHD, the ScaLAPACK decomposition, and the finite element assembly portions of the code. A plot of the CPU time for each of these portions of the code versus the number of processors is shown in Figure 6 for a variety of different problem sizes. The term ‘nps’ in the legend of the figures indicates the number of nodes per unit length. Therefore, the number of finite element nodes is nps^2 and the number of boundary element nodes for the GHD is $4 * \text{nps}$. Recall that, because of the limitation imposed by ScaLAPACK, all processor arrays are square. All timings in this section were performed on an IBM SP2 at the Maui High Performance Computing Center with PS2C nodes and 512 MB of RAM.

All timings except for the ScaLAPACK decomposition show almost perfect scalability. (The scalability of the algorithm as a whole is measured later.) That is, for the integral evaluation, GHD assembly, and FEM assembly, the CPU time essentially decreases in half when the number of processors is doubled. The reason for this is that there is little or

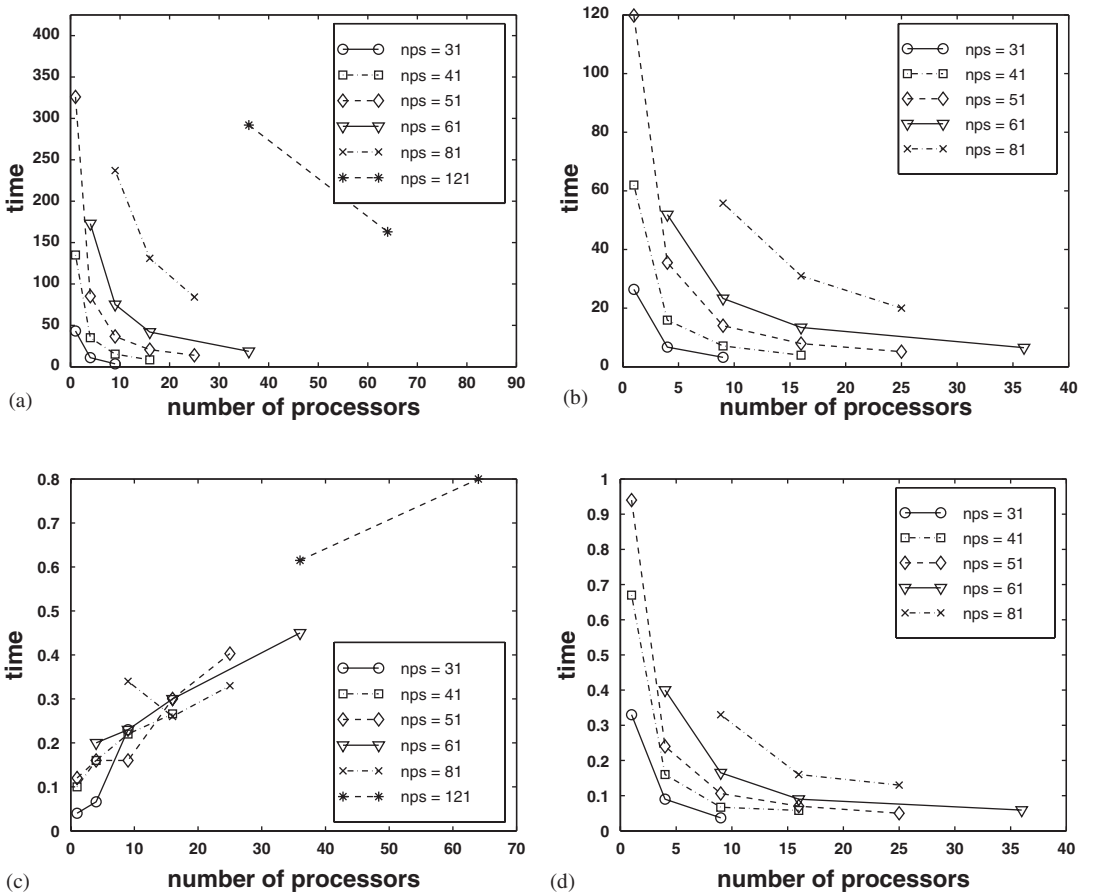


Figure 6. CPU times accrued outside the time loop as a function of the number of processors for (a) velocity integral evaluations, (b) Galerkin GHD assembly, (c) ScaLAPACK decomposition, and (d) finite element assembly.

no communication in these sections of the code as discussed in the previous section. Even though the ScaLAPACK decomposition time increases with increasing number of processors, these times are negligible compared to the velocity integral evaluation and Galerkin GHD assembly times. However, the timings indicate, for the problem sizes considered here, the communication costs associated with the decomposition dominate over the computational costs. Even for the largest problem shown ($\mathrm{nps} = 121$) which results in 480 boundary elements and a fully populated 484×484 matrix, the boundary element matrix is relatively small compared to the associated banded $14\,400 \times 367$ finite element matrix. Despite the additional cost of performing the decomposition in parallel outside the time loop, there are still advantages of this parallelization inside the time loop associated with the forward and backward substitution (as discussed below) to determine the vortex sheet strengths and there is also a savings in memory usage per processor.

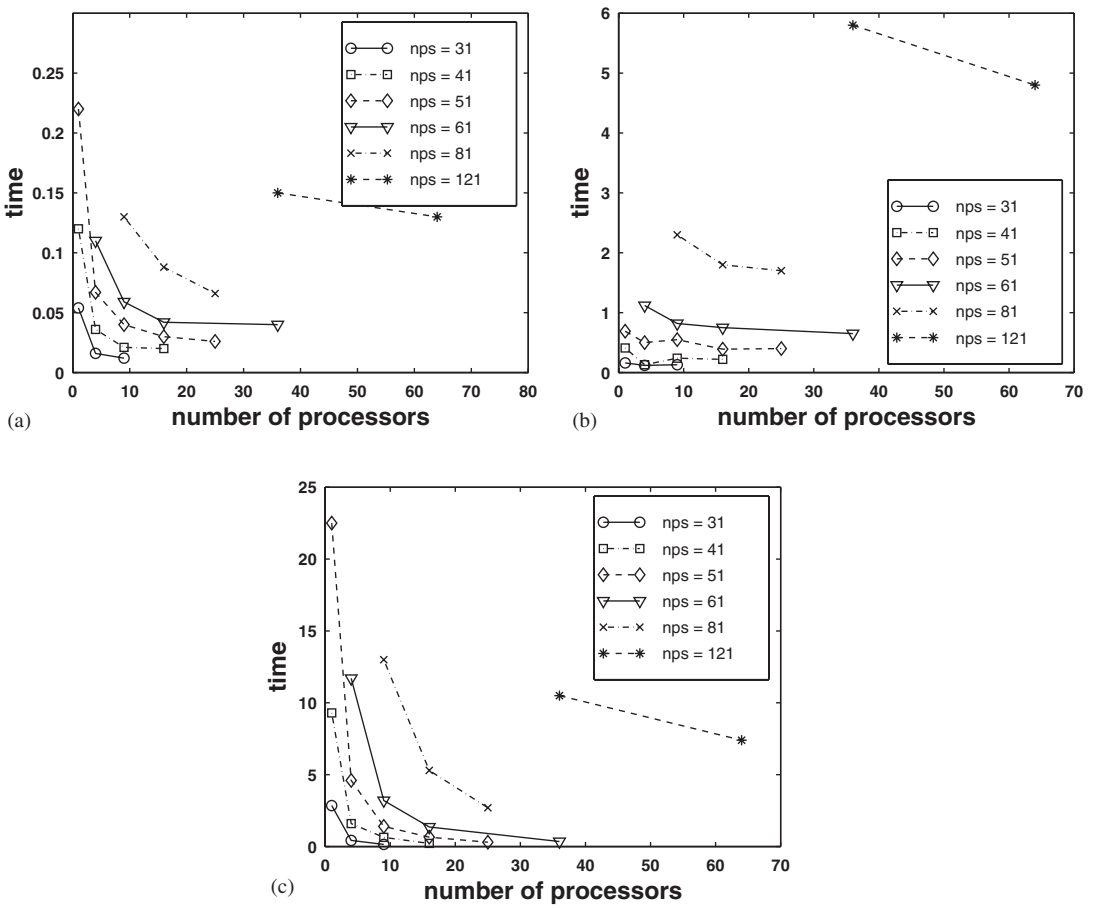


Figure 7. Clock times accrued per time step as a function of the number of processors for (a) Galerkin GHD forward and backward substitutions, (b) PIM solution of the discretized FEM equations and (c) velocity integral evaluations.

Timings performed inside the time loop include determining the vortex sheet strengths (step 1), determining the interior velocity field (step 2) and determining the updated vorticity field (step 3). These timings per time step are shown in Figure 7. The forward and backward substitutions to determine the vortex sheet strengths of step 1 and the matrix–vector multiplication to determine interior velocities of step 2 again show almost perfect scalability. The time within the loop used by PIM to solve the discretized FEM equations is not as scalable particularly for smaller problems using large numbers of processors.

Log–log plots of the total clock time outside the time loop and per time step inside the time loop are shown in Figure 8 for the case nps = 41. Also shown in the figure is the straight line curve representing ideal scalability. Ideal scalability is defined by the CPU time required on one processor divided by the number of processors. The clock time outside the loop shows essentially ideal speedup despite the poor scalability of the ScaLAPACK decomposition. As discussed above, the reason for this is that the decomposition represents only a fraction of

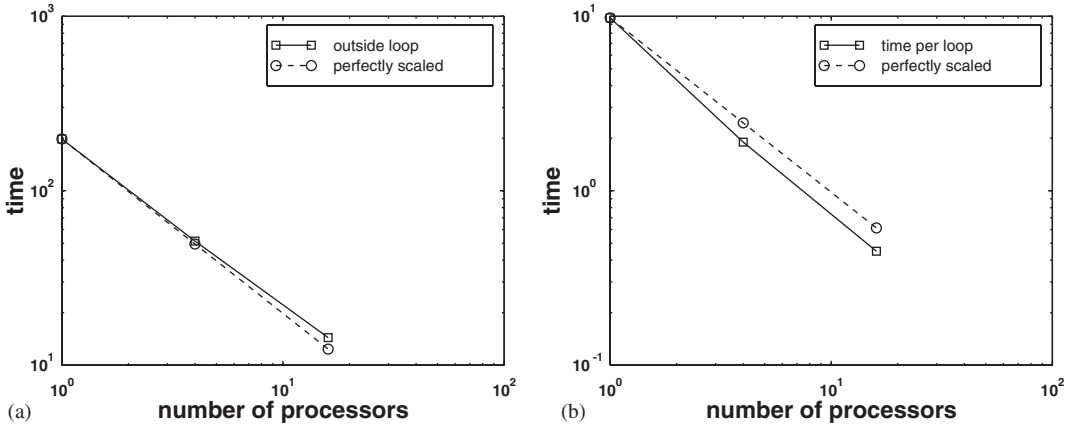


Figure 8. Clock times (a) outside the time loop and (b) inside the time loop per time step for the driven-cavity problem with NPS = 41.

the CPU effort outside the loop. Inside the loop, the speedup is also near ideal. In fact, the superlinear speedup shown inside the time loop increasing from 1 to 4 processors is most likely due to a reduction of cache misses between the 1 and 4 processor runs.

As discussed previously, the serial version of the vorticity code became memory bound far before it became CPU bound. To make sure that storage was minimized, all arrays whose size depended on the finite and boundary element discretizations such as nodal data, connectivity data, and arrays to store the results of numerical integrations were made allocatable at run time. These allocatable arrays comprised a minimum of 98% of the total memory usage as determined by a run-time system call. To show the advantages of the distributed memory, the number of allocated array elements per processor was measured for a number of different discretizations.

The first such measurement performed was for the driven-cavity problem with nps = 41. The total number of array elements when using one processor was 25.5×10^6. This number was reduced to 6.5×10^6 per processor using four processors and 1.8×10^6 per processor using 16 processors. These measurements show that the majority of array elements were distributed either in steps 1, 2 or 3 of the parallelization effort. As an example going from one processor to four processors, the total reduction in array elements was 19.0×10^6. Of that reduction, 0.1×10^6 could be attributed to step 1 (parallelization of the Galerkin GHD), 18.2×10^6 could be attributed to step 2 (parallelization of the velocity integrals) and 0.7×10^6 could be attributed to step 3 (parallelization of the Galerkin FEM). Hence, it is seen that the majority of the savings in memory per processor came from the parallelization of the evaluation of the velocity integrals.

To demonstrate the memory usage problem more dramatically, a plot of the total number of allocatable array elements per processor for various discretizations of the driven-cavity problem is shown in Figure 9. Also shown in the figure is a horizontal line indicating the maximum number of array elements that could be stored (double-precision) in memory on a 512 MB RAM machine, such as the IBM PS2C nodes used in this analysis. Hence, points above this horizontal line indicate problems with domain discretizations that would be memory

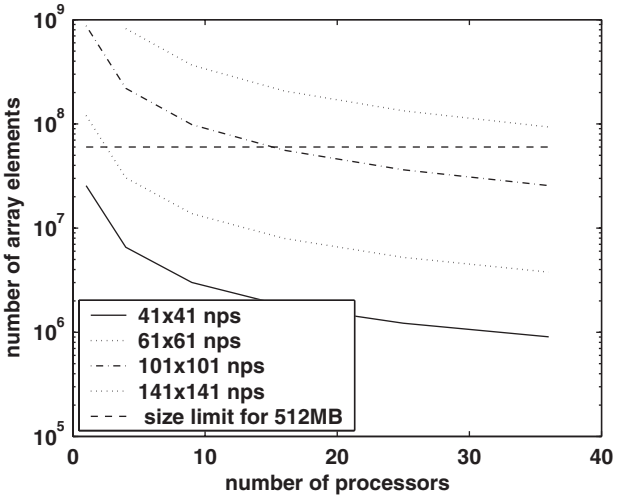


Figure 9. Number of allocatable array elements per processor for the driven-cavity problem.

bound on the current hardware. The figure clearly shows that larger problems are enabled by increasing the number of processors.

7. CONCLUSIONS

A vorticity formulation for the analysis of incompressible viscous fluid flows is presented based on a three-step algorithm. A Galerkin implementation of the generalized Helmholtz decomposition (GHD) is used in the first kinematic step to provide accurate vorticity flux boundary conditions. The regular form of the GHD is used in the second kinematic step to determine the interior velocity field. A finite element formulation for the vorticity equation is used in the third kinetic step to determine updated values of the interior vorticity field. The accuracy of this formulation was demonstrated using the interior driven cavity and exterior flow about a cylinder problems as benchmarks.

The current formulation is of interest from a parallel implementation point of view since each of the three steps requires a separate parallel strategy. The parallel implementation for the Galerkin GHD to determine vorticity flux boundary conditions is based on a block–block data distribution in which square portions of the discretized GHD matrix equations are assigned to processors. The LU decomposition using ScaLAPACK actually displayed a type of inverse scaling, that is, the LU decomposition actually took more time with increasing number of processors because of the communication overhead. Nevertheless, the parallelization of this step is still worthwhile since the backward and forward substitution within the time loop scales almost perfectly. Since the decomposition is only performed once and the backward and forward substitutions are performed literally thousands of time, there is an overall substantial savings in run time by parallelizing the first step.

The parallelization of the velocity integrals is an essentially embarrassingly parallel step since the evaluation at each finite element node is independent of each other node. Hence,

interior nodes can simply be distributed among the processors. As expected, almost ideal scaling is measured for this step.

The parallel implementation for the finite element solution of the vorticity equation is based on the total-summed-row approach in which rows of the discretized matrix equations are row-wrapped among processors. The assembly scales very well but the PIM GMRES routine is only modestly scalable. This result is somewhat puzzling since the matrix–vector multiplications required by PIM are independent. The only communication that should be required by the routine is a gather of residual vectors. However, as with many such software packages, the internal workings of PIM are opaque to the user.

The performance of the parallel vorticity code clearly shows the advantages of using multiple processors. Overall, the algorithm scales very well. However, the CPU savings tell only half of the story. Without the parallelization, the code examining flow over a 2-D cylinder would not have been capable of including much of a wake region because the serial code would become memory bound for reasonable discretizations. The memory savings per processor in a parallel implementation enable the analysis of larger problem domains while maintaining an adequate level of discretization.

REFERENCES

1. Roache P. *Computational Fluid Dynamics*. Hermosa Press: Albuquerque, 1972.
2. Parmentier EM, Torrance KE. Kinematically consistent velocity fields for hydrodynamic calculations in curvilinear coordinates. *Journal of Computational Physics* 1975; **19**:404.
3. Quartapelle L. Vorticity conditioning in the computation of two-dimensional viscous flows. *Journal of Computational Physics* 1981; **40**:453–477.
4. Quartapelle L, Vlaz-Gris F. Projection conditions on the vorticity in viscous incompressible flows. *International Journal of Numerical Methods in Fluids* 1981; **1**:129.
5. Anderson CR. Vorticity boundary conditions and boundary vorticity generation for two-dimensional viscous incompressible flow. *Journal of Computational Physics* 1989; **80**:72–97.
6. Koumoutsakos P, Leonard A, Pepin F. Boundary conditions for viscous vortex methods. *Journal of Computational Physics* 1994; **113**:52–61.
7. Gatski TB, Grosch CE, Rose ME. The numerical solution of the Navier–Stokes equations for 3-dimensional, unsteady, incompressible flows by compact schemes. *Journal of Computational Physics* 1989; **82**:298–329.
8. Daube O. Resolution of the 2D Navier–Stokes equations in velocity–vorticity form by means of an influence matrix technique. *Journal of Computational Physics* 1992; **103**:402–414.
9. Chorin AJ, Marsden JE. *A Mathematical Introduction to Fluid Mechanics.* Springer-Verlag: Berlin, 1990.
10. Wu JC, Thompson JF. Numerical solutions of time-dependent incompressible Navier–Stokes equations using an integro-differential formulation. *Computers and Fluids* 1973; **1**:197–215.
11. Wu JC. Numerical boundary conditions for viscous flow problems. *AIAA Journal* 1976; **14**:1042–1049.
12. Wu JC, Gulcat U. Separate treatment of attached and detached flow regions in general viscous flows. *AIAA Journal* 1979; **19**:20–27.
13. Wu JC. Boundary elements and viscous flows. In *Boundary Element Technology VII*. Babbia CA, Ingber MS (eds.). Elsevier Applied Science: Amsterdam, 1992.
14. Morino L. Helmholtz decomposition revisited: vorticity generation and trailing edge condition. *Computational Mechanics* 1986; **1**:65–90.
15. Morino L. Boundary integral equations in aerodynamics. *Applied Mechanics Review* 1993; **46**:445.
16. El-Refaee MM. Vortex lock-on for a rotationally oscillating circular cylinder—a BEM numerical study. *Engineering Analysis with Boundary Elements* 1995; **15**:235–247.
17. Ingber MS, Kempka SN. A Galerkin implementation of the generalized Helmholtz decomposition for vorticity formulations. *Journal of Computational Physics* 2001; **169**: 215–237.
18. Kamiya N, Iwase H, Kita E. Parallel implementation of boundary element method with domain decomposition. *Engineering Analysis with Boundary Elements* 1996; **18**(3):209–216.
19. Davies AJ. Fine-grained parallel boundary elements. *Engineering Analysis with Boundary Elements* 1997; **19**(1):13–16.
20. Gomez JE, Power H. A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number. *Engineering Analysis with Boundary Elements* 1997; **19**(1):17–32.

21. Semeraro BD, Gray LJ. PVM implementation of the symmetric-Galerkin method. *Engineering Analysis with Boundary Elements* 1997; **19**(1):67–72.
22. Davies AJ. The boundary element method on a transputer network. In *Boundary Elements XIII*. Brebbia CA, Gipson GS (eds.). Computational Mechanics Publications: Southampton, 1991.
23. Hendrickson BA, Womble DE. The torus-wrap mapping for dense matrix calculations on massively parallel computers. *SIAM Journal of Scientific Computing* 1994; **15**(5):1201–1226.
24. Ingber MS, Womble DE, Mondy LA. A parallel boundary element formulation for determining effective properties of heterogeneous media. *International Journal for Numerical Methods in Engineering* 1994; **37**: 3905–3919.
25. Natarajan R, Krishnaswamy D. A case study in parallel scientific computing: the boundary element method on a distributed-memory multicomputer. *Engineering Analysis with Boundary Elements* 1996; **18**:183–193.
26. Baltz BA, Ingber MS. A parallel implementation of the boundary element method for heat conduction analysis in heterogeneous media. *Engineering Analysis with Boundary Elements* 1997; **19**:3–11.
27. Hutchinson SA, Ng KT. A finite element algorithm for elliptical equations over unstructured domains on a data parallel computer. *International Journal for Numerical Methods in Engineering* 1994; **37**:3153–3167.
28. Barragy E, Carey GF. A parallel element-by-element solution scheme. *International Journal for Numerical Methods in Engineering* 1988; **26**:2367–2382.
29. Shadid JN, Hutchinson SA, Moffat HK, Hennigan GL, Hendrickson B, Leland RW. A 65+ Gflop/s unstructured finite element simulation of chemically reacting flows on the intel Paragon. *Proceedings of Supercomputing 94*. IEEE Computer Society Press: Silver Spring, MD, 1994.
30. Kempka SN, Glass MW, Perry JS, Strickland JH, Ingber MS. Accuracy considerations for implementing velocity boundary conditions in vorticity formulations. *Technical Report SAND*96-0583, Sandia National Laboratories, 1996.
31. Wu JZ, Wu XH, Ma HY, Wu JM. Dynamic vorticity condition: theoretical analysis and numerical implementation. *International Journal for Numerical Methods in Fluids* 1994; **19**:905–938.
32. Ghia U, Ghia KN, Chin CT. High-resolutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics* 1982; **48**:387–411.
33. Inoue O, Yamazaki T. Secondary vortex Streets in two-dimensional cylinder wakes. *Fluid Dynamics Research* 1999; **25**:1–18.
34. Allievi A, Bermejo R. Finite element modified method of characteristics for the Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 2000; **32**:439–464.
35. Chan CT, Anastasiou K. Solution of incompressible flows with or without a free surface using the finite volume method on unstructured triangular meshes. *International Journal for Numerical Methods in Fluids* 1999; **29**: 35–57.
36. Williamson CHK. Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder. *Physics of Fluids* 1988; **31**:2742–2744.
37. Roshko A. On the development of turbulent wakes from vortex streets. *NACA Report* 1191, 1954.
38. Vorobieff P, Ecke RE. Cylinder wakes in flowing soap films. *Physical Review E* 1999; **60**(3):2953–2956.
39. PIM. http://www.mat.ufrgs.br/pim-e.html.

