JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, August 2004

Simulation of Contaminant Dispersion on the Cray X1: Verification and Implementation

S. Aliabadi*
Clark Atlanta University, Atlanta, Georgia 30314

A. Johnson†
Network Computing Services, Inc., Minneapolis, Minnesota 55415

and

J. Abedi‡, S. Tu§, A. Tate**
Clark Atlanta University, Atlanta, Georgia 30314

Stabilized finite element formulation developed for simulation of dispersion of contaminants is implemented on the Cray X1. The stabilization is based on the SUPG and PSPG techniques. The governing equations are the incompressible Navier-Stokes equations coupled with the heat and mass transfer equations. The Boussinesq approximation in the momentum equation accounts for the density change due to thermal expansion. Fully implicit nonlinear systems of equations are solved iteratively using the matrix-free GMRES technique. The stabilized finite element formulation is parallelized and vectorized on the Cray X1. The three-dimensional validation problem involves transient simulation of flow past a building with source point releasing trances. Two-dimensional problems are simulated to compare the numerical results with analytical solutions.

I. Introduction

THE dispersion of chemical and biological agents (contaminants) in urban areas is a security concern. High performance computing can be used as an effective tool to predict the level of contaminant concentration at various times and locations. There are two major challenges in flow computations. First, the incompressible Navier-Stokes equations coupled with the heat and mass transfer equations for contaminants need to be solved. Second, the computational domain includes a large-scale city with thousands of commercial and residential buildings, roads, bridges, tress and many other landscape features. This makes the geometry construction, mesh generation and computation a very difficult task. The authors have developed many high performance computing (HPC) tools [1-5], which can simulate the dispersion of contaminants in a large-scale city much faster than real-time (See Fig. 1 and attached Movie 1). The heart of these HPC tools is a stabilized finite element flow solver [4]. This flow solver is highly parallelized and optimized for very large-scale simulations and runs at a sustained computational speed of between 85 and 100 Gigaflops on a Cray T3E-1200 system with 1024 processor. The matrix-free[6-7] feature of this flow solver enables the users to use extremely refined finite element meshes for simulations. In the recent benchmark, we successfully used a totally unstructured finite element mesh with more than one billion tetrahedral

Received 5 November 2003; revision received 27 January 2004; accepted for publication 28 January 2004.

*Director and Associate Professor, Computation and Modeling Laboratory, Department of Engineering.
†Senior Scientist, Army HPC Research Center.
‡Assistant Professor, Computation and Modeling Laboratory, Department of Engineering.
§Postdoctoral Research Associate, Computation and Modeling Laboratory, Department of Engineering.
**Undergraduate Student, Computation and Modeling Laboratory, Department of Engineering.

elements for implicit simulation of flow around complex geometry.[1] Recently, we have parallelized and vectorized this flow solver for optimized implementation on the Cray X1 supercomputer. Here, the authors will limit the scope of this article to the validation and verification of the finite element flow solver and its implementation on the Cray X1.

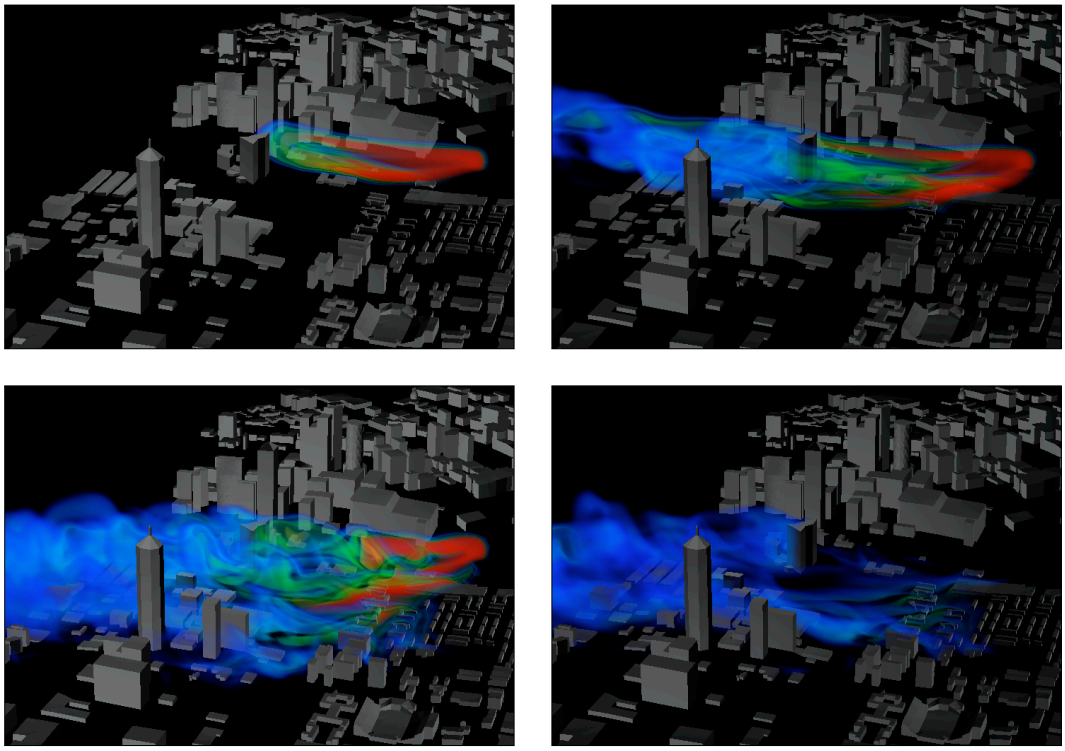


Fig. 1 Contaminant Dispersion in downtown Atlanta: The four pictures show the volume rendering of the contaminant at various instances. The red and blue colors represent the highest and lowest level of contaminant concentration. The simulations are carried out on the Cray T3E using 512 processors. See also Movie 1.

The Cray X1 is a completely new architecture built using a hybrid parallel, vector and multi-streaming design. It is a "Capability Machine" built to achieve high-sustained computational performance for large numerical simulations. The basic component of the Cray X1 is the Multi-Streaming Processor (MSP), which is a multi-module chip composed of four Single-Streaming Processor (SSP) modules and four cache modules. The MSP is the user-addressable processing element, and the division of work (i.e. streaming) of the four SSP modules is directed by the compiler (however, the user does have the ability to dictate the streaming process by using Cray Streaming Directives). Each SSP module is a fast vector processor which contains 2 vector pipes and 32 vector registers. Each SSP also contains a slower scalar processing element, and because of this scalar performance slowness, codes running on the Cray X1 should be fully vectorized. The peak computational performance of an MSP is 12.8 Gigaflops. Four MSPs reside together on a singe Cray X1 node board, along with shared memory (16 Gigabytes per node-board, but 32 Gigabytes is also available in some models), I/O modules and inter-processor network routing modules.

The Cray X1 is also a parallel computer. Multiple MSP processors are linked together within a fast interconnect network to achieve high-bandwidth and low-latency for inter-processor messages. Various parallel programming modules are supported such as MPI, SHMEM, Co-array Fortran (CAF), and Unified Parallel C (UPC). OpenMP on a single node board will also be supported in the near future. For the CFD codes discussed here, MPI is generally used; however, the performance of CAF and UPC has been tested within these codes. In September of 2002, the Army High Performance Computing Research Center (AHPCRC) was the first non-classified site to receive Cray X1 systems. Currently, the center supports a 128 processor (MSP) Cray X1 liquid-cooled system. This system has a total peak performance of 1.64 Teraflops and contains 512 Gigabytes of memory.

In our finite element flow solver, the governing equations are the incompressible Navier-Stokes equations coupled with the heat and mass transfer equations. This coupling makes these problems extremely nonlinear (especially at high Rayleigh numbers).[8] Many upwind finite difference schemes have been developed to relax the nonlinearities, but in most cases, the algorithms lack stability and contain inconsistencies. Robust, stable and consistent numerical algorithms are needed to solve these problems with a reasonable degree of accuracy.

The Galerkin finite element formulations are stabilized using the SUPG (Streamline-Upwind/Petrov-Galerkin) and PSPG (Pressure-Stabilization/Petrov-Galerkin) methods.[1-5, 9-10] The SUPG stabilization term allows us to solve flow problems at high speeds (advection dominant flows) and the PSPG term eliminates instabilities associated with the use of equal order interpolation functions for both pressure and velocity. The accuracy of this finite element flow solver for natural convection problems [11] (buoyancy driven flows) has been reported by Aliabadi et al.[4] In this article, we emphasis on the accuracy of the contaminant transport equation coupled with the incompressible Navier-Stokes equations.

There are many numerical studies of flow around buildings. The earlier studies were attempted by Murakami et al.[12] and Dawson et al.[13] who used κ–ε turbulence models to simulate flow around an isolated building. The TEMPEST code[19] has been used by many researchers to perform three-dimensional flow study around a benchmark problem where the experimental data is available. The experiment was carried out using a towing tank by USEPA (US Environmental Protection Agency, Snyder[15]). Zhang et al.[16] and Smith[17] used TEMPEST to carry out numerical studies of this particular problem. We will use the results obtained in Refs. 21-22 as reference for our verification study.

In Sec. II we will introduce the governing equations. The finite element formulations are provided in Sec. III. The matrix-free iterative scheme is reviewed in Sec. IV. The Cray X1 implementation is presented in Sec. V followed by the numerical examples in Sec. VI.

II. Governing Equations

The governing equations are the incompressible Navier-Stokes equations coupled with the heat and mass transfer equations written over the spatial domain Ω with boundary Γ. In non-dimensional form, these equations can be written as

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\frac{G_r}{R_e^2} \rho \theta \, \mathbf{n}_g + \nabla \cdot \boldsymbol{\sigma} + \mathbf{Y}, \tag{2}$$

$$\boldsymbol{\sigma} = -\mathrm{p}\mathbf{I} + \frac{2}{R_e}\boldsymbol{\varepsilon}(\mathbf{u}), \tag{3}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}), \tag{4}$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \frac{1}{R_e P_r} \nabla \cdot \nabla \theta + \dot{q}, \tag{5}$$

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \frac{1}{R_e P_r L_e} \nabla \cdot \nabla c + \dot{n}, \tag{6}$$

where ρ, $\mathbf{u}(u,v,w)$, $\mathbf{p}$, θ, c, $\mathbf{n}_g$ and $\mathbf{Y}$ are the total density, velocity, pressure, temperature, concentration of contaminant, the unit vector in the direction of the gravity, and the body force field, respectively. The strain tensor is denoted by $\boldsymbol{\varepsilon}$, and $\mathbf{I}$ represents the identity tensor. The Reynolds, Prandtl, Lewis, and Grashoff numbers are denoted by R_e, P_r, L_e, and G_r, respectively. Here $\dot{q}$ and $\dot{n}$ stand for the sources of heat and mass generations, respectively. In these equations, the Boussinesq approximation is considered to account for density variations due to thermal gradients.

Equations (1-6) are completed by an appropriate set of boundary and initial conditions. Let's assume that the boundary Γ is decomposed into four boundaries: Γ_{in}(inflow), Γ_{out}(outflow), Γ_{solid} (solid object), and Γ_{sym}(symmetry plane). On each boundary, we consider the following boundary conditions:

$$\begin{cases} \mathrm{u} = \mathrm{u}_{\mathrm{in}} \\ \theta = \theta_{\mathrm{in}} \\ c = c_{\mathrm{in}} \end{cases} \quad \text{on } \Gamma_{\mathrm{in}}, \tag{7}$$

$$\begin{cases} \mathrm{n}.\boldsymbol{\sigma} = \mathrm{t} \\ \mathrm{n}\cdot\nabla\theta = h \\ \mathrm{n}\cdot\nabla c = s \end{cases} \quad \text{on } \Gamma_{\mathrm{out}}, \tag{8}$$

$$\mathrm{u} = 0 \quad \text{on } \Gamma_{\mathrm{solid}}, \tag{9}$$

$$\mathrm{n}.\mathrm{u} = 0 \quad \text{on } \Gamma_{\mathrm{sym}}, \tag{10}$$

where **n** is the normal vector to the boundary. Here, the subscript "in" denotes imposed boundary condition. At outflow, the traction force per unit area, **t**, heat flux, *h*, and mass flux, *s*, are imposed (set to zero in our simulations). If the outflow boundary is far enough, these natural boundary conditions in Eq. (8) can be assumed to be zero. Note that pressure by itself has no boundary condition.

The fluid velocity, temperature and concentration of contaminants are given initially as

$$\begin{cases} \mathrm{u}(t=0) = \mathrm{u}_0 \\ \theta = \theta_0 \\ c = c_0 \end{cases} \quad \text{on } \Omega. \tag{11}$$

Note that the initial velocity filed should also be divergence free,

$$\nabla \cdot \mathrm{u}_0 = 0 \ \text{ on } \Omega. \tag{12}$$

III. Finite Element Formulation

In the finite element formulations we first define appropriate sets of trail solution spaces S_{u}^h, S_p^h, S_θ^h, and $S_c^h$and weighing function spaces V_{u}^h, V_p^h, V_θ^h, and V_c^h for the velocity, pressure, temperature and the concentration of contaminant. The stabilized finite element formulation of Eqs. (1-6) can then be written as follows: for all $\mathrm{w}^h \in V_{\mathbf{u}}^h$, $q^h \in V_p^h$, $\psi^h \in V_\theta^h$, and $\Phi^h \in V_c^h$, find $\mathrm{u}^h \in S_{\mathbf{u}}^h$, $\mathrm{p}^h \in S_p^h$, $\theta^h \in S_\theta^h$ and $c^h \in S_c^h$ such that

$$
\begin{aligned}
&\int_{\Omega} q^h \nabla \cdot \mathbf{u}^h d\Omega \\
&+\int_{\Omega} \mathbf{w}^h \cdot [\rho(\frac{\partial \mathrm{u}^h}{\partial t} + \mathrm{u}^h \cdot \nabla \mathrm{u}^h + \frac{G_r}{R_e^2}\theta^h \, \mathrm{n}_g - \mathrm{Y})] d\Omega + \int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega \\
&+\int_{\Omega} \psi^h (\frac{\partial \theta^h}{\partial t} + \mathrm{u}^h \cdot \nabla \theta^h - \dot{q}) d\Omega + \int_{\Omega} \frac{1}{R_e P_r} \nabla \psi^h \cdot \nabla \theta^h \, d\Omega \\
&+\int_{\Omega} \Phi^{\mathrm{h}} (\frac{\partial c^h}{\partial t} + \mathrm{u}^h \cdot \nabla c^h - \dot{n}) d\Omega + \int_{\Omega} \frac{1}{R_e P_r L_e} \nabla \Phi^{\mathrm{h}} . \nabla c^h \, d\Omega \\
&+\sum_{e=1}^{ne} \int_{\Omega^e} \frac{\tau_m}{\rho} \left[\rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h - \nabla \cdot \boldsymbol{\sigma}(q_p^h, \mathbf{w}^h) \right] \\
&\cdot \left[\rho(\frac{\partial \mathrm{u}^h}{\partial t} + \mathrm{u}^h \cdot \nabla \mathrm{u}^h - \mathrm{g} + \frac{G_r}{R_e^2}\theta^h \, \mathrm{n}_g - \mathrm{Y}) - \nabla \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \right] d\Omega \\
&+\sum_{e=1}^{ne} \int_{\Omega^e} \tau_\theta \mathbf{u}^h \cdot \nabla \psi^h \left[\frac{\partial \theta^h}{\partial t} + \mathrm{u}^h \cdot \nabla \theta^h - \frac{1}{R_e P_r} \nabla \cdot \nabla \theta^h - \dot{q} \right] d\Omega \\
&+\sum_{e=1}^{ne} \int_{\Omega^e} \tau_{\mathrm{c}} \mathbf{u}^h \cdot \nabla \Phi^{\mathrm{h}} \left[\frac{\partial c^h}{\partial t} + \mathrm{u}^h \cdot \nabla c^h - \frac{1}{R_e P_r L_e} \nabla \cdot \nabla c^h - \dot{n} \right] d\Omega \\
&+\sum_{e=1}^{ne} \int_{\Omega^e} \tau_s \nabla \cdot \mathbf{w}^h \rho^h \nabla \cdot \mathbf{u}^h d\Omega = \int_{\Gamma_{\mathrm{out}}} \left(\mathbf{w}^h . \mathrm{t} + \psi^h h + \Phi^{\mathrm{h}} s \right) d\Gamma .
\end{aligned}
\tag{13}
$$

In Eq. (13), the first seven integrals together with the right hand side term are the Galerkin formulation of the governing equations. The 8th term (first element-level integral) includes the SUPG and the PSPG stabilizations of the momentum equation. The SUPG stabilization eliminates numerical oscillation due to advection dominance and the PSPG stabilization allow us to use equal order interpolation functions for both velocity and pressure. Here, the 9th and 10th terms (second and third element-level integrals) are the SUPG stabilization for the heat and mass transfer equations, respectively. The last term in the left-hand-side (4th element-level integral) is the least-square stabilization of the continuity equation. This term enhances stabilization at high Reynolds number flows. The finite element formulations are discretized using first order polynomials for both the unknowns and the weighing functions.

The stabilization parameters $\boldsymbol{\tau}_m$, $\boldsymbol{\tau}_\theta$, $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_s$ are defined in Refs. 1-5, 9-10 as

$$
\boldsymbol{\tau}_m = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2 \| \mathrm{u} \|}{h} \right)^2 + \left(\frac{4}{R_e h^2} \right)^2 \right]^{-\frac{1}{2}}, \tag{14}
$$

$$
\boldsymbol{\tau}_\theta = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2 \| \mathrm{u} \|}{h} \right)^2 + \left(\frac{4}{R_e P_r h^2} \right)^2 \right]^{-\frac{1}{2}}, \tag{15}
$$

$$
\boldsymbol{\tau}_c = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2 \| \mathrm{u} \|}{h} \right)^2 + \left(\frac{4}{R_e P_r L_e h^2} \right)^2 \right]^{-\frac{1}{2}}, \tag{16}
$$

$$
\boldsymbol{\tau}_s = \frac{h}{2} \| \mathrm{u} \| z, \qquad z = \begin{cases} R_u / 3 & R_u \le 3 \\ 1 & 3 < Ru \end{cases}. \tag{17}
$$

IV. Matrix-Free Iterative Scheme

The matrix-free iterative scheme[19] is a crucial part in the vectorization and parallelization of our flow solver. We review this scheme in this section.

The discretization and linearization of the finite element formulation in Eq. (13) results in a coupled linear system of equations, which need to be solved. Due to the extreme size of the problems, we use GMRES[20] iterative algorithm with diagonal preconditioning to obtain the approximate solution of the problem. The GMRES iterative algorithm requires many matrix-vector multiplications to obtain parameters, which minimize the residual. In a typical finite element technique, the matrix-vector multiplications are performed at the element-level, and therefore eliminating the need to form the left-hand side matrix globally. Still, the element-level matrices need to be formed and stored in the core memory. For tetrahedral elements and six degrees of freedom per node (three component of velocity, pressure, temperature and concentration of contaminant), the element-level matrices are 24x24, which require 4.5 kilobytes of memory per element to store the entries. We typically use unstructured finite element meshes with hundreds of millions of tetrahedral elements. Assuming a mesh with 500 million elements, the total memory to store the element-level matrices is 500,000,000 x 4.5 = 2.25 Terabytes. If each processor of the parallel computer has 512 Megabytes, we need more than 4500 processors only to store the element-level matrices. An alternative way is to store a sparse left-hand-side matrix. This will eliminate the storage requirements, but still significant amount of memory is required.

To eliminate this bottleneck, the authors have developed the matrix-free iterative scheme. The matrix-free-iterative scheme is very suitable iterative solver for vector operations and vectorization on the Cray X1. In the matrix-free method, the result of a matrix-vector operation can be directly computed as a vector. This operation takes place at the element-level and the element-level vectors are assembled to form the global vectors. The linearization of the finite element formulation in Eq. (13) results in a finite element formulation written as

$$
\begin{aligned}
&\int_\Omega q^h \nabla\cdot\Delta\mathbf{u}^h d\Omega \\
&+\int_\Omega \mathbf{w}^h\cdot[\rho(\frac{\Delta \mathrm{u}^h}{\Delta t}+\alpha \mathrm{u}^h\cdot\nabla\Delta \mathrm{u}^h+\alpha\Delta \mathrm{u}^h\cdot\nabla \mathrm{u}^h+\alpha\frac{G_r}{R_e^2}\Delta\theta^h \mathrm{n}_g)]d\Omega \\
&+\int_\Omega \alpha\boldsymbol{\varepsilon}(\mathbf{w}^h):\boldsymbol{\sigma}(\Delta p^h,\Delta\mathbf{u}^h)d\Omega \\
&+\int_\Omega \psi^h(\frac{\Delta\theta^h}{\Delta t}+\alpha \mathrm{u}^h\cdot\nabla\Delta\theta^h+\alpha\Delta \mathrm{u}^h\cdot\nabla\theta^h)d\Omega+\int_\Omega \alpha\frac{1}{R_eP_r}\nabla\psi^h\cdot\nabla\theta^h\, d\Omega \\
&+\int_\Omega \Phi^{\mathrm{h}}(\frac{\Delta c^h}{\Delta t}+\alpha \mathrm{u}^h\cdot\nabla\Delta c^h)d\Omega+\int_\Omega \alpha\frac{1}{R_eP_rL_e}\nabla\Phi^{\mathrm{h}}.\nabla\Delta c^h\, d\Omega \\
&+\sum_{e=1}^{ne}\int_{\Omega^e}\frac{\tau_m}{\rho}\Big[\rho\mathbf{u}^h\cdot\nabla\mathbf{w}^h-\nabla\cdot\boldsymbol{\sigma}(q_p^h,\mathbf{w}^h)\Big] \\
&\cdot\Big[\rho(\frac{\Delta \mathrm{u}^h}{\partial t}+\alpha \mathrm{u}^h\cdot\nabla\Delta \mathrm{u}^h+\alpha\frac{G_r}{R_e^2}\Delta\theta^h \mathrm{n}_g)-\alpha\nabla\cdot\boldsymbol{\sigma}(\Delta p^h,\Delta\mathbf{u}^h)\Big]d\Omega \\
&+\sum_{e=1}^{ne}\int_{\Omega^e}\tau_\theta\mathbf{u}^h\cdot\nabla\psi^h\Big[\frac{\Delta\theta^h}{\Delta t}+\alpha \mathrm{u}^h\cdot\nabla\Delta\theta^h-\alpha\frac{1}{R_eP_r}\nabla\cdot\nabla\Delta\theta^h\Big]d\Omega \\
&+\sum_{e=1}^{ne}\int_{\Omega^e}\tau_{\mathrm{c}}\mathbf{u}^h\cdot\nabla\Phi^{\mathrm{h}}\Big[\frac{\Delta c^h}{\Delta t}+\alpha \mathrm{u}^h\cdot\nabla\Delta c^h-\alpha\frac{1}{R_eP_rL_e}\nabla\cdot\nabla\Delta c^h\Big]d\Omega \\
&+\sum_{e=1}^{ne}\int_{\Omega^e}\tau_s\nabla\cdot\mathbf{w}^h\rho^h\nabla\cdot\Delta\mathbf{u}^h d\Omega=Residual,
\end{aligned}
\tag{18}
$$

where Δ denotes the increment. In this formulation, α is the time integration parameter.[19] Note that in the stabilization terms, partial linearization is performed. This finite element formulation is a linear formulation in terms of $\Delta \mathrm{u}$, Δp, $\Delta\theta$ and Δc. We can write this equation in a compact form using a bilinear operator as

$$A(N,M)=Residual, \tag{19}$$

where $N=(\mathrm{w}^h,q^h,\psi^h,\Phi^h)$ and $M=(\Delta \mathrm{u}^h,\Delta p^h,\Delta\theta^h,\Delta c^h)$.

In the discretization of this finite element formulation, we expand **N** and **M** using finite element basis functions:

$$\mathbf{N} = \sum_{i=1}^{nn-nd} \mathbf{c}_i N_i, \quad \mathbf{M} = \sum_{j=1}^{nn-nd} \mathbf{m}_j N_j, \tag{20}$$

where *nn* is the number of nodes and *nd* is the number of nodes with Direchlet boundary conditions. This results in a linear system of equations in this form:

$$\mathbf{Ax} = \mathbf{b}, \tag{21}$$

which needs to be solved at each iteration. Note that the right hand side is the residual, which in the discrete form can be calculated directly as a vector. In the matrix-free iterative scheme, the result of the matrix-vector multiplication can be computed similar to residual computation as a vector. This can be achieved by replacing the $\Delta\mathbf{u}^h$, Δp^h, $\Delta\theta^h$, and Δc^h in Eq. (18) with the corresponding minimizing vector values at the integration point. Let the minimizing vector to be

$$\mathbf{V} = \sum_{j=1}^{nn-nd} \mathbf{v}_j N_j. \tag{22}$$

Then

$$\mathbf{AV} = \mathbf{A(N,V)}, \tag{23}$$

which can be calculated as a vector directly. This operation is performed at the element level and assembled to form the global vector. For more details, readers can refer to Ref. 19.

V. Cray X1 Implementation

For high sustained performance of these CFD codes on the Cray X1, all of the loops in the code must be vectorized. Any significant scalar calculations in the code will be an overall drag on performance and limit the computer's overall effectiveness. When multi-streaming is mixed in with the vectorization, an optimal (minimum) vector length of 256 is required (without streaming, a single SSP has an optimal vector length of 64). Also, the usual rules that govern vectorization such as data inter-dependence and order of execution still apply. However, the Cray X1 is a new design and has a new compiler, so some of the traditional trouble spots for vectorization such as "IF" statements (branching), square-roots, and memory gather/scatter operations are no longer problems and the vectorizing compiler is very flexible. Our CFD codes, in general, do contain long vector lengths and does not have very many vectorization problems, so initial porting of the codes to the X1 went well and didn't take very much time. However, there was one significant problem with the vectorization of the matrix-free, vector-based formation of the GMRES resultant vector. Within the GMRES algorithm's iterations, node-level vectors will be generated that need to be multiplied by the left-hand-side matrix. In the matrix-free method, the left-hand-side matrix isn't formed, so we must "derive" the resultant vector directly each time a vector product is required by GMRES. We generally refer to this function as the "Block," and it is the main computational kernel of our CFD codes and is called many times. The Block usually takes between 70 and 80 percent of total computational time.

The problem with vectorization of the Block function arises because of a unstructured memory scatter operation at the very end of this routine. The Block represents a typical finite-element vector formation (assembly), and a pseudo-code of the Block function is shown next.

```
do i=1,numer_of_elements

  n1 = ien(1,i)

  n2 = ien(2,i)
```

```
  n3 = ien(3,i)
  n4 = ien(4,i)
  u1 = d(1,n1)
  v1 = d(2,n1)
  w1 = d(3,n1)
  u2 = d(1,n2)
  v2 = d(2,n2)
  w2 = d(3,n2)
  u3 = d(1,n3)
  …several more memory gather statements like these…

  uX = u1*sh1DX + u2*sh2DX + u3*sh3DX + u4*sh4DX
  uY = u1*sh1DY + u2*sh2DY + u3*sh3DY + u4*sh4DY
  uZ = u1*sh1DZ + u2*sh2DZ + u3*sh3DZ + u4*sh4DZ
  vX = v1*sh1DX + v2*sh2DX + v3*sh3DX + v4*sh4DX
  vY = v1*sh1DY + v2*sh2DY + v3*sh3DY + v4*sh4DY
  vZ = v1*sh1DZ + v2*sh2DZ + v3*sh3DZ + v4*sh4DZ
  wX = w1*sh1DX + w2*sh2DX + w3*sh3DX + w4*sh4DX
  …many more statements like these…

  r(1,n1) = r(1,n1) + ures1
  r(2,n1) = r(2,n1) + vres1
  r(3,n1) = r(3,n1) + wres1
  r(1,n2) = r(1,n2) + ures2
  r(2,n2) = r(2,n2) + vres2
  …several more memory scatter statements like these…

enddo
```

In this pseudo-code, iterations take place over the mesh elements (on each individual processor) and in the beginning, memory is gathered from the node vectors. After the relevant variables are gathered, the main computations take place using these "localized" variables, and about 1,000 floating point operations take place for each iteration. After the calculations take place for each iteration, the results are added (i.e. assembled) to the global node-vector, which represents the "result" of the matrix-vector multiplication. Due to the fact that we use unstructured meshes, the node references 'n1, n2, n3, n4' could be referring to any nodes of the mesh for each iteration. If this loop was vectorized, it is possible that repeated node references are encountered and results will be added on top of each other during a vector operation. In such cases, data will be "lost" and errors will occur in the results. The Cray X1 vector compiler will not vectorize loops like this by default because of these memory scatter statements, and if we force vectorization by telling the compiler that it is safe to do so, we do observe errors in the results.

To avoid this problem with vectorization of these memory scatter operations, we applied an element coloring scheme as part of the set-up portions of our CFD codes. In this mesh pre-processing, the total number of elements are broken-up into smaller groups of elements (i.e. they are given a color). The rule for forming element groups is that no two elements in a group can be referring to the same mesh element (i.e. there will be no repeated node indices within the 'ien' array for any two elements in a group). The algorithm to generate these groups of elements is fairly simple. Once the elements are broken-up into groups, we can slightly modify our element loop in the Block function to contain an outer group loop, and an inter element loop. Since we have guaranteed that there will be no repeated 'n1, n2, n3, n4' indices, we can force vectorization of the inner element loop by applying a "CONCURRENT" directive. This directive is used in place of the more standard "IVDEP" directive found on previous Cray architectures. The modified pseudo-code is shown below.

```
do j=1,numer_of_groups
  e1 = groups_beg(j)
  e2 = groups_end(j)
!DIR$ CONCURRENT
  do i=e1,e2
   n1 = ien(1,i)
   n2 = ien(2,i)
   n3 = ien(3,i)
   n4 = ien(4,i)
   u1 = d(1,n1)
   v1 = d(2,n1)
   w1 = d(3,n1)
   u2 = d(1,n2)
   v2 = d(2,n2)
   w2 = d(3,n2)
   u3 = d(1,n3)
   …several more memory gather statements like these…
```

```
      uX = u1*sh1DX + u2*sh2DX + u3*sh3DX + u4*sh4DX

      uY = u1*sh1DY + u2*sh2DY + u3*sh3DY + u4*sh4DY

      uZ = u1*sh1DZ + u2*sh2DZ + u3*sh3DZ + u4*sh4DZ

      vX = v1*sh1DX + v2*sh2DX + v3*sh3DX + v4*sh4DX

      vY = v1*sh1DY + v2*sh2DY + v3*sh3DY + v4*sh4DY

      vZ = v1*sh1DZ + v2*sh2DZ + v3*sh3DZ + v4*sh4DZ

      wX = w1*sh1DX + w2*sh2DX + w3*sh3DX + w4*sh4DX

      …many more statements like these…


      r(1,n1) = r(1,n1) + ures1

      r(2,n1) = r(2,n1) + vres1

      r(3,n1) = r(3,n1) + wres1

      r(1,n2) = r(1,n2) + ures2

      r(2,n2) = r(2,n2) + vres2

      …several more memory scatter statements like these…


    enddo

  enddo
```

The number of elements in each group determines the "vector length" for the Block. In our unstructured meshes, usually generated by an automatic mesh generator, we typically see group sizes containing thousands of elements. On average, the majority of the groups may contain around 10,000 elements or larger, with a few smaller groups for the remainder of elements. These sizes are, of course, much larger than the preferred minimum vector-length size.

With the full vectorization of the Block function, very high sustained performance is achieved. For the Block function, which takes up the vast majority of time for our CFD codes, we observe anywhere between 4.25 to 5.1 Gigaflops per MSP on the Cray X1. This relates to a sustained performance of between 33 and 40 percent of peak performance of the MSP. We usually observe between 5% and 8% of peak performance on commodity processors (Pentium, SGI, IBM). When we include the other parts of the code, including the GMRES solver, residual and pre-conditioner formation, and inter-processor communication, we observe an overall sustained rate of around 3.5 Gigaflops (on average) per MSP. Comparisons of our CFD codes on the X1 with other architectures have shown that the Cray X1 is between 45 and 50 times faster than a Cray T3E-1200, between 16 to 25 times faster than an SGI Origin (500 MHz MIPS R14000), and between 12 and 14 times faster than an IBM p690 (1.3 GHz Power4). These numbers are all on a per-processor bases comparing total run times (excluding set-up time). Parallel scalability on the Cray X1 is also excellent with the code spending only a small percentage of total execution time on inter-processor communication. More information on the porting, optimization, and performance of these CFD codes on the Cray X1 can be found in Ref. 21.

VI. Numerical Examples

The finite element formulation presented in this article can be applied to a wide range of incompressible flows from low speed Stokes flows to high speed high Reynolds number flows. The three problems presented in this section demonstrate the flow solver capability.

Stokes Problem. This validation problem, known as the Stokes problem, has been studied by many authors.[22-23] In this problem, we solve the two-dimensional Navier-Stokes equations, on a unit square $\Omega = [0,1]\times[0,1]$. Here, The Reynolds number, density and Grashoff number are set to be 100, 1 and 0, respectively. The body force field, $Y = (Y_1, Y_2)$, is given by

$$\begin{aligned} Y_1(t,x,y) &= 2(t+1)x^2(1-x)^2(2y-6y^2+4y^3) \\ &+u(t+1)^2(2y-6y^2+4y^3)(2x-6x^2+4x^3) \\ &+v(t+1)^2x^2(1-x)^2(2-12y+12y^2) \\ &-0.01(t+1)^2(2y-6y^2+4y^3)(2-12x+12x^2) \\ &-0.01(t+1)^2x^2(1-x)^2(-12+24y)+2x. \end{aligned} \tag{24}$$

$$\begin{aligned} Y_2(t,x,y) &= 2(t+1)y^2(1-y)^2(-2x+6x^2-4x^3) \\ &+u(t+1)^2y^2(1-y)^2(-2+12x-12x^2) \\ &+v(t+1)^2(2y-6y^2+4y^3)(-2x+6x^2-4x^3) \\ &-0.01(t+1)^2y^2(1-y)^2(12-24x) \\ &-0.01(t+1)^2(-2x+6x^2-4x^3)(2-12y+12y^2)-2y. \end{aligned} \tag{25}$$

The initial conditions are the following velocity and pressure fields:

$$\begin{aligned} u_0 &= x^2(1-x)^2(2y-6y^2+4y^3), \\ v_0 &= y^2(1-y)^2(-2x+6x^2-4x^3), \\ p_0 &= x^2-y^2, \end{aligned} \tag{26}$$

and no-slip condition $(\mathbf{u}=0)$ is applied on all boundaries. Actually, this problem is constructed by substituting the following analytic solutions into Eqs. (1-4) to obtain the body force field. The body force will be the driving force of this Stokes flow.

$$\begin{aligned} u(t,x,y) &= (t+1)^2x^2(1-x)^2(2y-6y^2+4y^3), \\ v(t,x,y) &= (t+1)^2y^2(1-y)^2(-2x+6x^2-4x^3), \\ p(x,y) &= x^2-y^2, \end{aligned} \tag{27}$$

We use our three-dimensional SUPG finite element incompressible flow solver to simulate this two-dimensional problem. The mesh is shown in Fig. 2, where the element length is approximately 0.02. In this mesh, the two-dimensional unit square is extended in the z-direction with a height of 0.06. A finer mesh with half the element length is also tested. Figure 3 shows the initial condition computed from Eq. (3). The pressure field does not change with time. However, the velocity field evolves due to the time-dependent body force. Figure 4 shows the velocity field at specific times. The time steps in the current test are $\Delta t = 0.005$ and 0.0025 for accuracy comparison. The implicit second order accurate Crank-Nicholson time integration scheme is used in our solver. Figure 5 shows error vs. time of the four cases. The error is computed as the root-mean-square of the difference between the numerical solution and exact solution on all nodes.

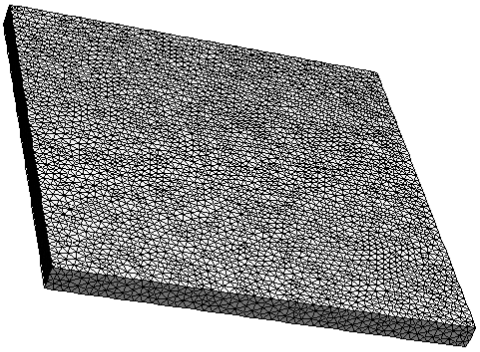


Fig. 2 Stokes problem: Finite element mesh

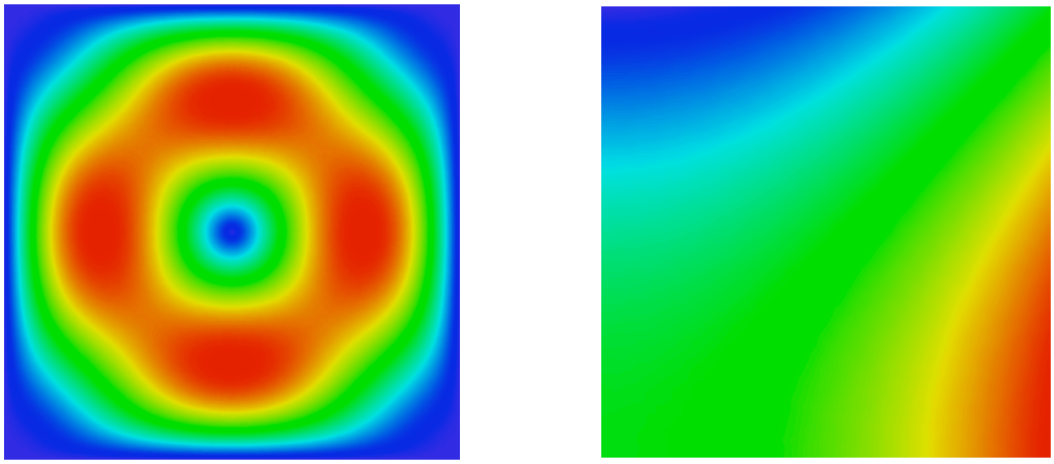
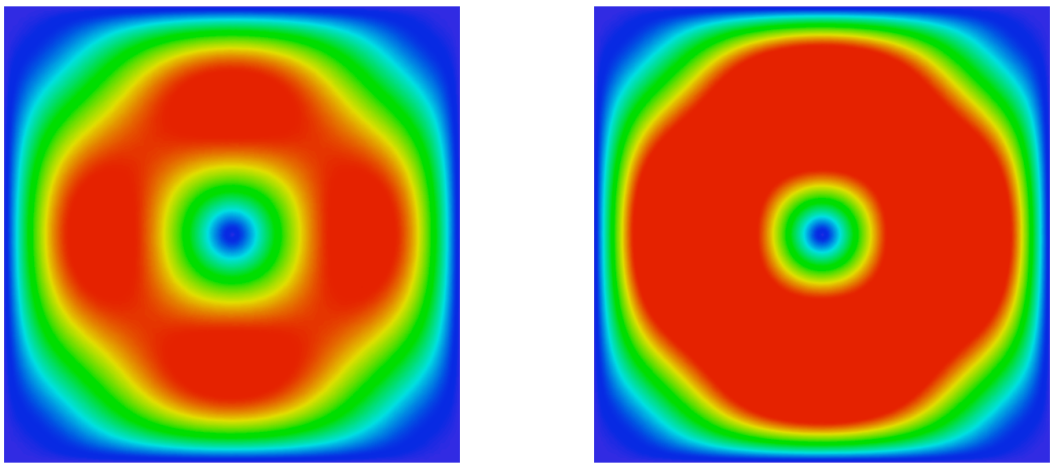
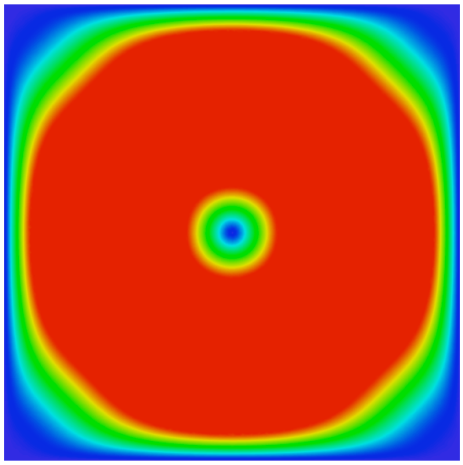


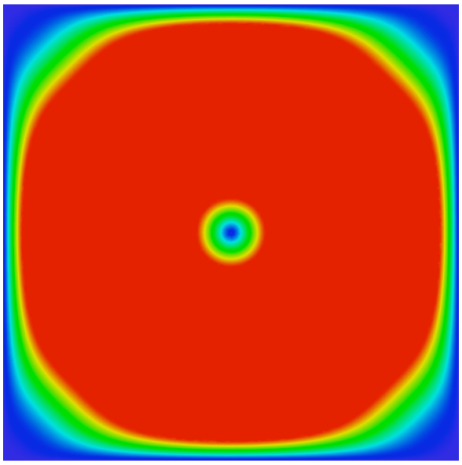
Fig. 3 Stokes problem: Initial condition for the velocity (left) and pressure (right).



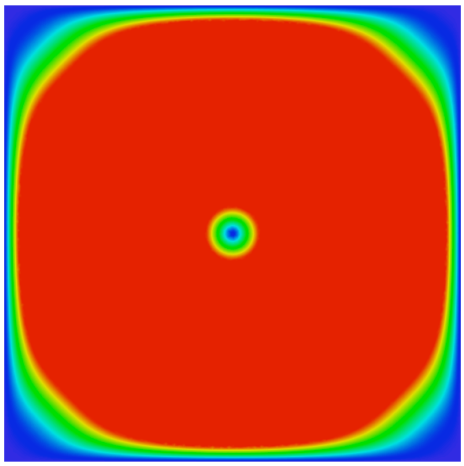
(a) $t = 0.06$ (b) $t = 0.2$



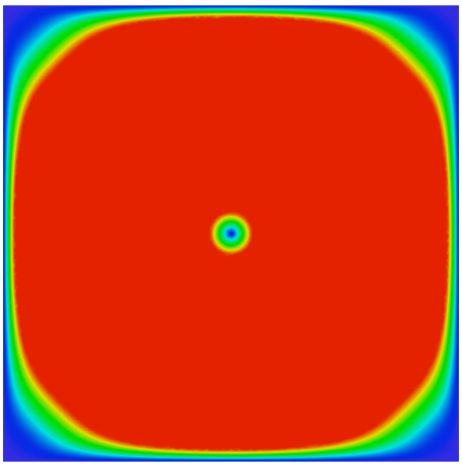
(c) $t = 0.4$



(d) $t = 0.6$



(e) $t = 0.8$



(f) $t = 1.0$

Fig. 4 Stokes problem: Velocity field at different times.

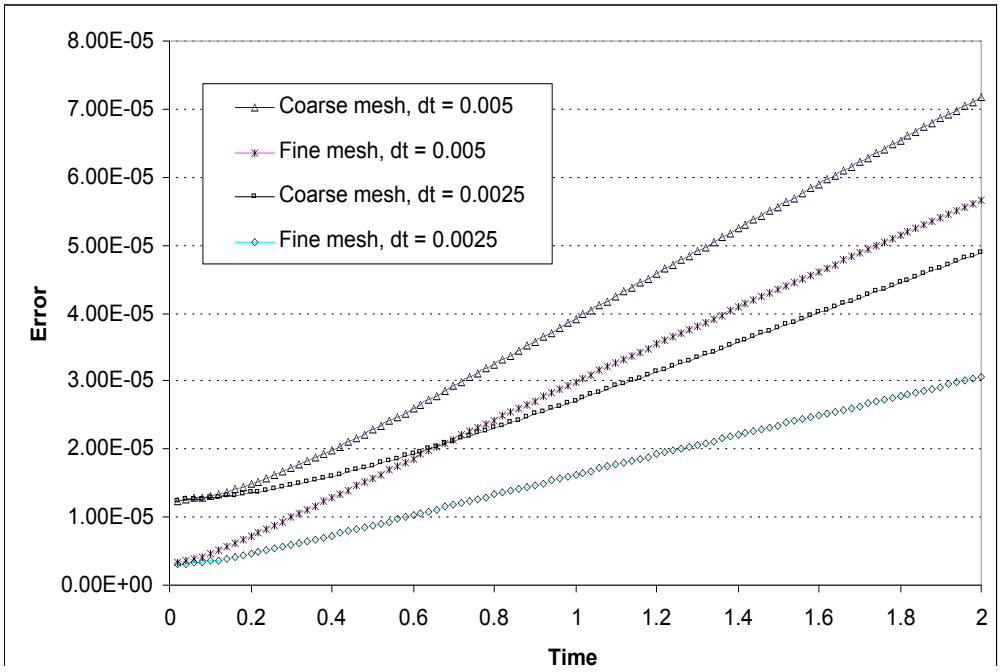


Fig. 5 Stokes problem: Error vs. time. The element length is 0.02 for the coarse mesh and 0.01 for the fine mesh.

Contaminant Transport. In this problem we measure the accuracy of the contaminant transport in a circular velocity field on the unit square domain $\Omega = [-0.5,0.5]\times[-0.5,0.5]$. This problem is presented in Ref. 24 and also discussed in Ref. 23.

The governing equation is given in Eq. (6). In this problem, the diffusion and source term are set to zero. The velocity field is given by

$$u = -y \text{ and } v = x, \tag{28}$$

whose streamlines are circular around the center of the unit square. The concentration is set to be zero on all external boundaries. On the internal boundary *OA* (see Fig. 6), the concentration is prescribed to be a cosine curve. The initial condition is given by

$$\begin{cases} c = \cos\left[2\pi(y+0.25)\right] & \text{on OA in Fig. 6,} \\ c = 0 & \text{on the rest of the domain} \end{cases} \tag{29}$$

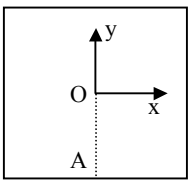
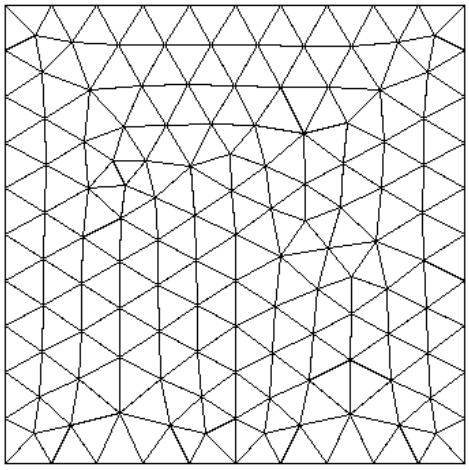


Fig. 6 Contaminant transport: Geometry of the problem.

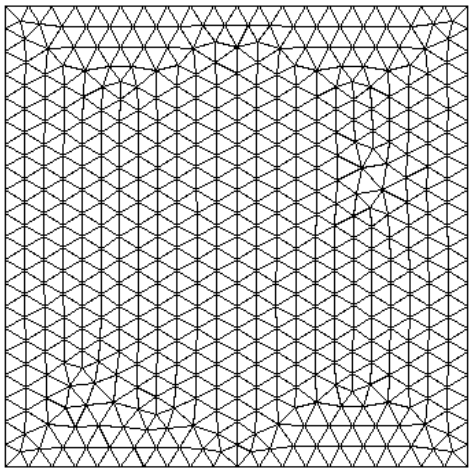
The exact steady-state solution is essentially a pure advection of the initial condition along the circular streamlines. In other words, this is a rigid rotation problem about the center of the domain. The steady state is assumed to be reached when the following criteria is satisfied:

$$\frac{\left\| c^n - c^{n-1} \right\|}{\left\| c^n \right\|} \leq TOL\,, \tag{30}$$

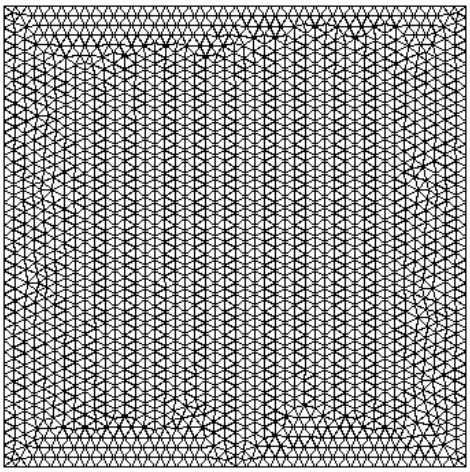
where $\|\cdot\|$ is the L_2 norm and TOL is the given tolerance value chosen as 1.e-4 in the current case. Figure 7 shows four meshes with different refinement values. The refinement value is the length of the triangle edges on the boundary and can be considered as the element characteristic length. Figure 8 shows the solution on different meshes. Also shown is the exact solution on the same mesh. Table 1 summarizes the results for all four meshes. The error analysis shows that the scheme is first order accurate in the advection limit using linear triangles. From these results, we can see that the error also drops linearly with the refinement value. Figure 9 demonstrates this linear relationship. The error is the root-mean-square of the difference between the numerical solution and the exact solution on all nodes. Figure 10 shows the convergence history for all four meshes.



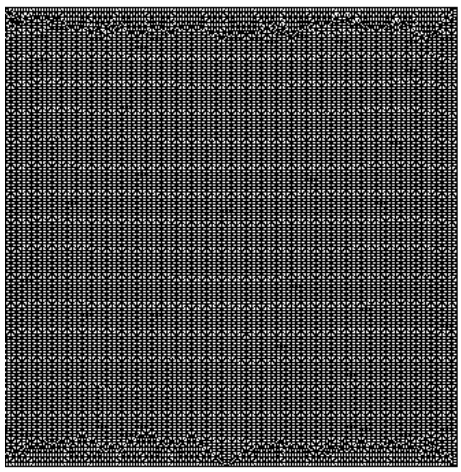
(a) mesh 0 with element length of 0.1.



(b) mesh 1 with element length of 0.05.

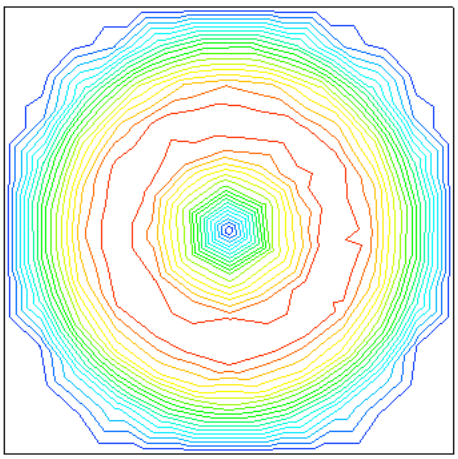


(c) mesh 2 with element length of 0.025.

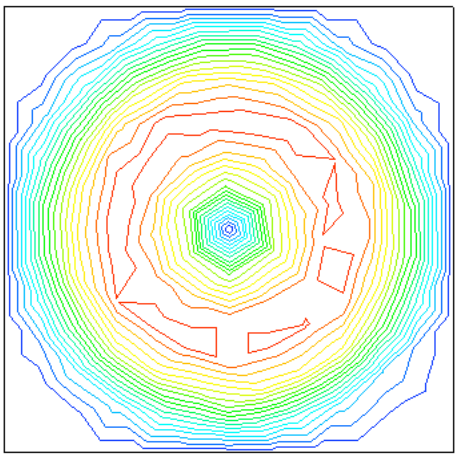


(d) mesh 3 with element length of 0.0125.

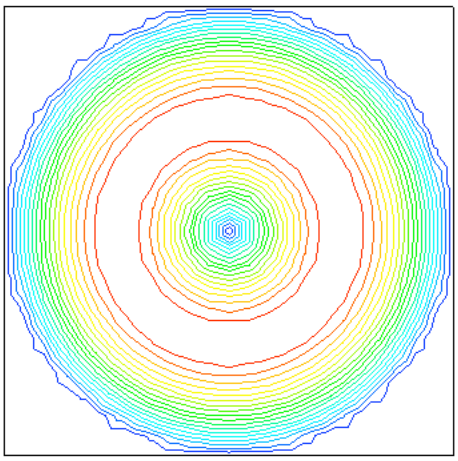
Fig. 7 Contaminant transport: Mesh with different refinements.



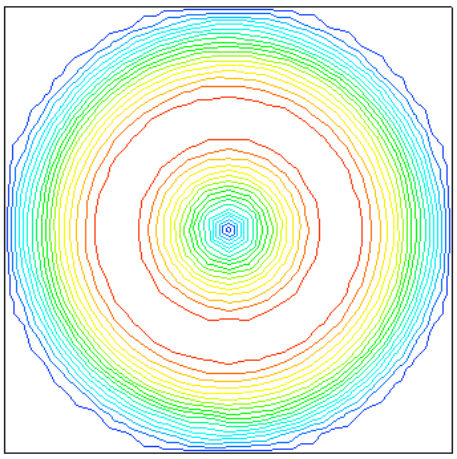
(a) exact solution on mesh 0.



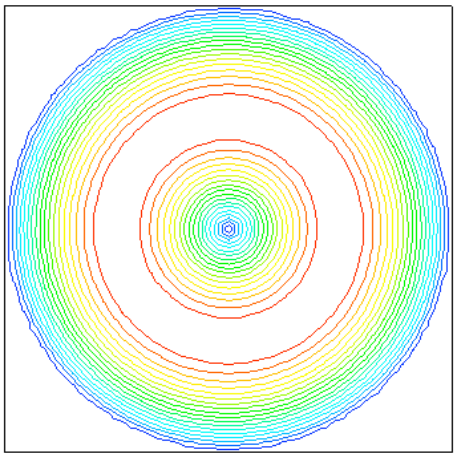
(b) numerical solution on mesh 0.



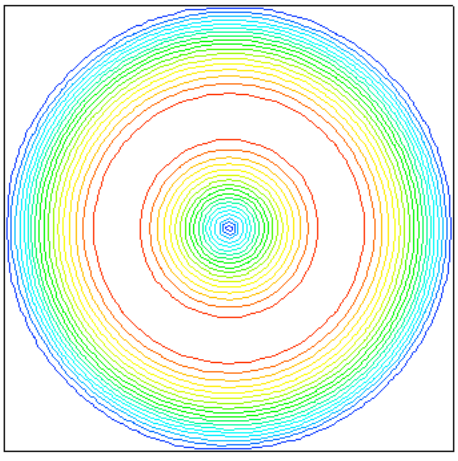
(c) exact solution on mesh 1.



(d) numerical solution on mesh 1.



(e) exact solution on mesh 2.



(f) numerical solution on mesh 2.

Fig. 8 Contaminant transport: Contour plots for exact solutions and numerical solutions.

Table 1 Contaminant transport: $\Delta t = 0.1$, TOL = 1.e-4 for all mesh

Mesh	Element size	Number of elements	Number of time steps	Error
Mesh 0	0.1	244	97	2.188E-02
Mesh 1	0.05	948	92	1.053E-02
Mesh 2	0.025	3704	89	5.331E-03
Mesh 3	0.0125	14762	89	2.456E-03

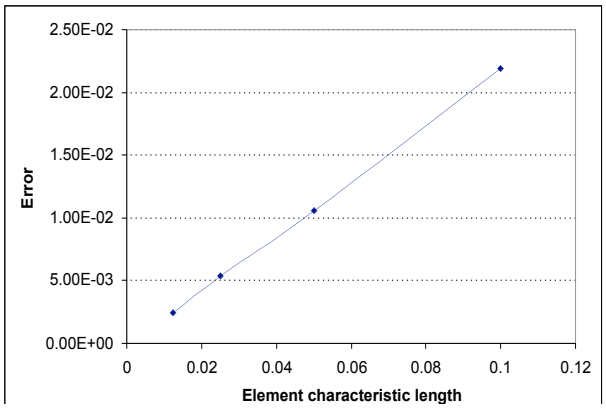


Fig. 9 Contaminant transport: Linear relationship between the error and the element characteristic length.

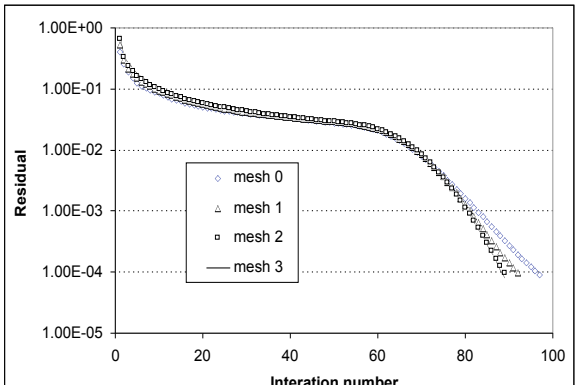


Fig. 10 Contaminant transport: Convergence history.

Contaminant Dispersion Behind a Building: This problem has been studied to validate the numerical results with experimental data, reported in Ref. 15. In the experiment, a 10 cm cube representing a model building was mounted on a 1.2 x 2.4 m flat baseplate. The model was mounted with the front edge 2.3 cm downstream from the leading edge of the baseplate. To generate a uniform inflow velocity, the plate was towed in a towing tank 1.2 m deep, 2.5 m wide and 25 m long. The details of this experiment can be found in Refs. 15-17.

The source tube, with diameter of 0.09 H (H is the building height) was located behind the building, on the ground-level centerline at a downswind distance of 0.25 H from the base of the building (see Fig. 11). The ground-level concentration profile was obtained using 25 sampling ports, equally spaced and located on the ground between the building and 6 H downwind along the a-a and b-b lines in Fig. 11. The vertical concentration was measued along the c-c line in Fig. 11 using a vertical rake of 10 tubes.

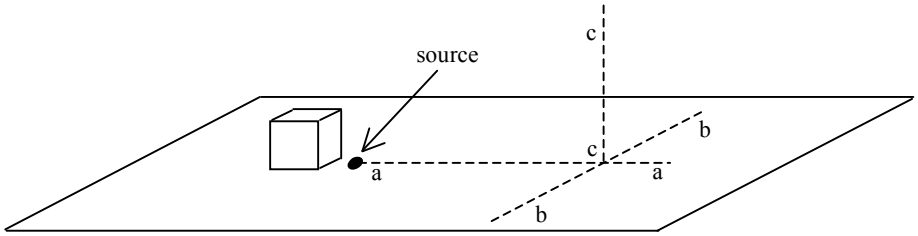


Fig. 11 Contaminant dispersion behind a building: Problem Description.

The numerical simulations reported here correspond for the case with Froude number of ∞ (uniform temperature distribution).[16] In the experiment, it is obsered that under uniform temperature distribution, the critical Reynolds number, where the flow becomes Reynolds-number-independent, is around 11,000 to 13,000. The large-eddy simulations are carried out for the Reynolds number of 13,000 using the Smagorinsky turbulence model.[25] The simulations are carried out for 500 time steps, which correspond to almost one hour of release time. The time accurate simulations are averaged for the last 20 minutes of the actual release time to obtain the statistical soultion.

The computed concentration is compared with the experimental data downstream of the building. For the comparison, the normalized concentration is defined as $x = cU_\infty H^2/\dot{m}$, where U_∞ is the inflow velocity (7 m/s) and $\dot{m}$ is the tracer release rate (kg/s). During the experiment, a considerable amount of scatter was observed in the data as the experiments were repeated.

We computed this problem using three unstructured finite element meshes made of tetrahedral elements. The coarse mesh has 696,430 elements. The refined mesh has 6,353,079 elements and the very refined mesh has 50,824,632 elements.

The snapshot of the flow field is shown in Fig. 12, Movie 2 and Movie 3. In this figure, the ground-level concentration, viewed from the top at two instances is shown. As expected, there are many vorticities and instabilities in the flow field.

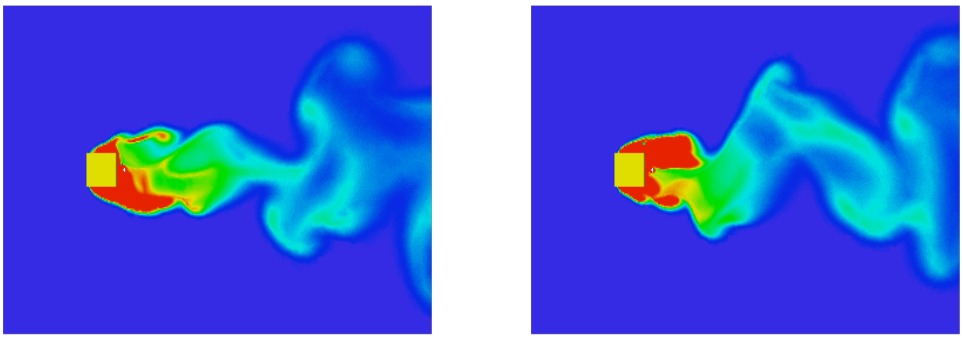


Fig. 12 Contaminant Dispersion Behind a Building: Ground level concentration of contaminant at two instances. See also Movie 2 and Movie 3.

Figures 13, 14, and 15 show the comparison between the computed concentration profiles downstream of the building with the experimental data. In these figures, the upper and lower band of the experiment shows the scatter observations as the experiments were repeated. Also, for the comparison, the results obtained by Zhang[16] are also provided in these figures.

The results reported by Zhang overestimate the concentration level. Our finite element results are all within the experimental range, except the coarse mesh results which over predicts the concentration level near the source. The

mesh refinement seems to resolve this problem. The refined and very refined meshes provide soultions that predict the concentration level within the experimental range, even in the region close to the source location.

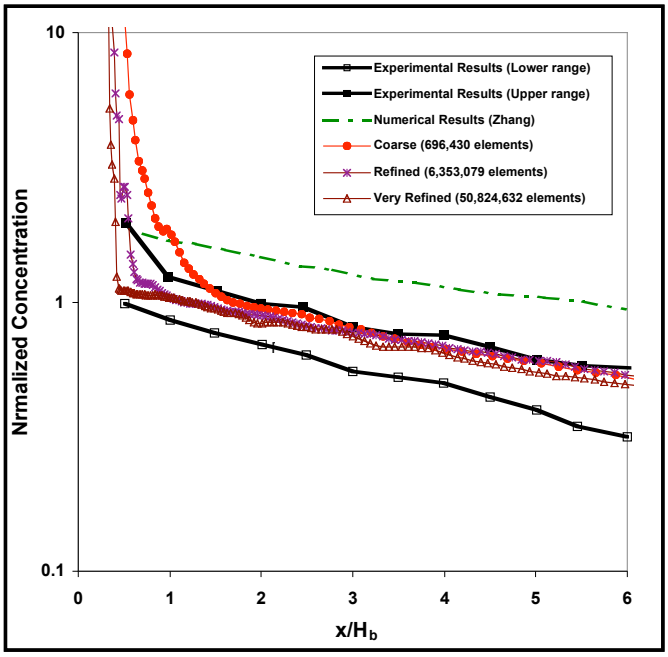


Fig. 13 Contaminant dispersion behind a building: Normalized ground level concentration along the center line (a-a line in the Fig. 11).

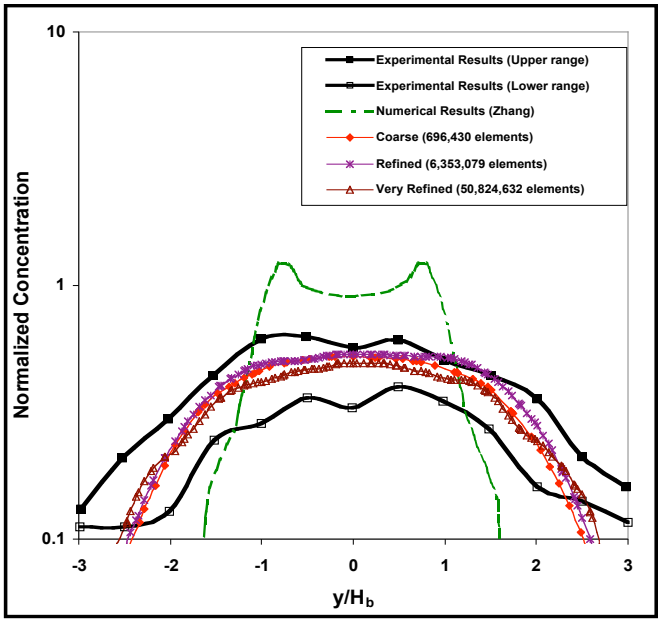


Fig. 14 Contaminant dispersion behind a building: Normalized lateral ground level concentration along the b-b line in the Fig. 11.

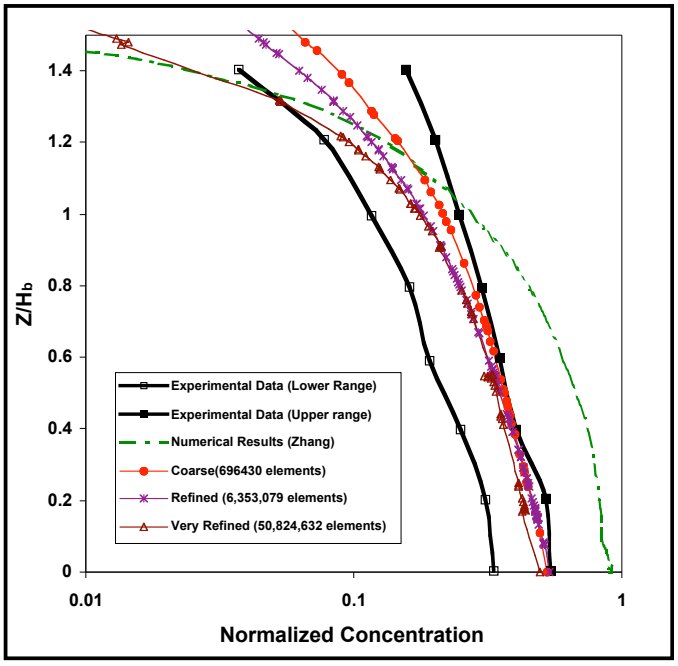


Fig. 15 Contaminant Dispersion Behind a Building: Normalized vertical concentration along the c-c line in the Fig. 11.

VII. Conclusion

An incompressible flow solver based on stabilized finite element formulations is developed to predict the concentration of contaminants in complex three-dimensional domains. The flow solver has efficiently been parallelized and vectorized on the Cray X1 supercomputer. The sustained computational speed is around 3.5-4.5 Gigaflops per processor of the Cray X1. The accuracy of this flow solver is demonstrated with three numerical examples. This flow solver is very fast and reasonably accurate.

VIII. Acknowledgment/Disclaimer

This work is funded in part by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory contract numbers DAAD19-01-2-0014 and DAAD19-0three-dimensional-0001. Partial support for this paper was made possible through support provided by DoD High Performance Computing Modernization Program (HPCMP), Programming Environment & Training (PET) activities through Mississippi State University under the terms of Agreement No. GS04T01BFC0060. Views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army and Department of Defense position, policy, or decision unless so designated by other official documentation and no official endorsement should be inferred.

References

[1]Aliabadi, S., Johnson, A., Abedi, J., Bota, K., and Zellars, B., "Finite Element Simulation of Hydrodynamics Problems Using Unstructured Meshes with more than One Billion Elements," *Proceedings of 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, 2002.

[2]Aliabadi, S., Abedi, J., and Zellars, B., "Parallel Finite Element Simulation of Mooring Forces on Floating Objects," *International Journal for Numerical Methods in Fluids*, Vol. 41, 2003, pp. 809-822.

[3]Aliabadi, S., Abedi, J., Zellars, B., Bota, K., and Johnson, A., "Simulation Technique for Wave Generation," *Communications in Numerical Methods in Engineering*, Vol. 19, 2003, pp. 349-359.

[4]Aliabadi, S., Johnson, A., Abatan, A. A., Abedi, J., Yeboah, Y., and Bota, K., "Stabilized Finite Element Formulation of Buoyancy Driven Incompressible Flows," *Journal of Communications in Numerical Methods in Engineering (CNME)*, Vol. 18, Issue 5, 2002, pp. 315-324.

[5]Johnson, A. A., and Tezduyar, T. E., "Simulation of Multiple Spheres Falling in a Liquid-Filled Tube," *Computer Methods in Applied Mechanics and Engineering*, Vol. 134, 1996, pp. 351-373.

[6]Aliabadi, S. K., and Tezduyar, T. E., "Parallel Fluid Dynamics Computations in Aerospace Applications," *International Journal for Numerical Methods in Fluids*, Vol. 21, 1995, pp. 783-805.

[7]Johan, Z., Hughes, T. J. R., and Shakib, F., "A Globally Convergence Matrix-Free Algorithm for Implicit Time-Marching Schemes Arising in Finite Element Analysis in Fluids," *Computer Methods in Applied Mechanics and Engineering*, Vol. 87, 1991, pp. 281-304.

[8]Barakat, Z. H., and Clark, A. J., "Analytical and Experimental Study of Transient Laminar Natural Convection Flows in Partially Filled Liquid Containers," *Proceedings of Third International Heat Transfer Conference*, Vol. 2, 1966, pp. 152-162.

[9]Tezduyar, T., Aliabadi, S., Behr, M., Johnson, A., Kalro, V., and Litke, M., "Flow Simulation and High Performance Computing," *Computational Mechanics*, Vol. 18, 1996, pp. 397-412.

[10]Aliabadi, S., and Tezduyar, T., "Stabilized-Finite-Element/Interface-Capturing Technique for Parallel Computation of Unsteady Flows with Interfaces," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, 2000, pp. 243-261.

[11]Kulacki, F. A., and Emara, A. A., "Steady and Transient Thermal Convection in a Fluid Layer with Uniform Volumetric Energy Sources," *Journal of Fluid Mechanics*, Vol. 55, Part 2, 1977, pp. 271-287.

[12]Murakami, S., Mochida, A., and Hibi, K., "3-D Numerical Simulation of Air-Flow Around a Cubic Model by Means of Large Eddy Simulation," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 25, 1987, pp. 291-305.

[13]Dawson, P., Stock, D. E., and Lamb, B., "The Numerical Simulation of Airflow and Dispersion in Three-Dimensional Atmospheric Recirculation Zones," *Journal of Applied Meteorology*, Vol. 30, 1991, pp. 1005-1024.

[14]Trend, D. S., and Eyler, L. L., "A Three-Dimensional Time-Dependent Computer Program for Hydrothermal Analysis," *Numerical Methods and Input Instructions*, Vol. 1, PNL-4348, Pacific Northwest Laboratory, Battelle, Washington, 1989.

[15]Snyder, W. H., "Some Observation of the Influence of Stratification on Diffusion in Building Wakes," *Stably Stratified Flows: Flow and Dispersion over Topography*, edited by I. P. Castro and N. J. Rockliff, Clarendon Press, Oxford, 1994, pp. 310-324.

[16]Zhang, Y. Q., Arya, S. P., and Snyder, W. H., "A Comparison of Numerical and Physical Modeling of Stale Atmospheric Flow and Dispersion Around a Cubical Building," *Atmospheric Environment*, Vol. 30, 1996, pp. 1327-1345.

[17]Smith, W. S., Reisner, J. M., and Kao, C. Y. J., "Simulation of Flow Around a Cubical Building: Comparison With Towing-Tank Data and Assessment of Radioactivity Induced Thermal Effects," *Atmospheric Environment*, Vol. 35, 2001, pp. 3811-3821.

[18]Tezduyar, T., Aliabadi, S., and Behr, M., "Enhanced-Discretization Interface-Capturing Technique (EDICT) for Computation of Unsteady Flows with Interfaces," *Computer Methods in Applied Mechanics and Engineering*, Vol. 155, 1998, pp. 235-248.

[19]Aliabadi, S., "Parallel Finite Element Method in Aerospace Applications," Ph.D. Thesis, Dept. of Aerospace and Mechanics, Univ. of Minnesota, 1994.

[20]Saad, Y., and Schultz, M., "GMRES: Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856-896.

[21]Johnson, A., "Computational Fluid Dynamics Applications on the Cray X1 Architecture: Experiences, Algorithms, and Performance Analysis," *Cray User Group Conference 2003 Proceedings*, May 2003. Data also available online at http://www.ahpcrc.org (cited May 2004).

[22]Carey, G. F., and Krishnan, R., "Penalty Approximation of Stokes Flow," *Computational Methods in Applied Mechanical Engineering*, Vol. 35, 1982, pp. 169-206.

[23]Valli, A. M. P., "Control Strategies for Timestep Selection in Finite Simulation of Incompressible Flows and Coupled Heat and Mass Transfer," Technical Report, Programa de Engenharia Civil, COPPE, 2001.

[24]Brooks, A. N., and Hughes, T. J. R., "Streamline Upwind/Petrov-Galerkin Formulation for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations," *Computational Methods in Applied Mechanical Engineering*, Vol. 32, 1982, pp.199-259, 1982.

[25]Kato, C., and Ikegawa, M., "Large Eddy Simulation of Unsteady Turbulent Wake of a Circular Cylinder Using the Finite Element Method," *Advances in Numerical Simulation of Turbulent Flows*, FED-Vol. 117, ASME, New York, 1991, pp.49-56.