Unsteady Incompressible Flow Simulation Using Galerkin Finite Elements with Spatial/Temporal Adaptation

Mohamed S. Ebeida[*]
Carnegie Mellon University, Pittsburgh, PA 15213

Roger L. Davis[†] and Roland W. Freund[‡]
University of California, Davis, CA 95616

A new adaptive technique for the simulation of unsteady incompressible flows is presented. The initial mesh is generated based on a Cartesian grid with spatial decomposition and a simple optimization step to define the boundaries of the domain. This technique is fast and produces a quad-dominant mesh, while preserving the quality of the elements. Adaptive mesh refinement is performed based on the gradient of the vorticity from the previous time step. The time step is controlled and hence adapted using error estimation of the flow variables with respect to time. A Galerkin finite-element discretization is used to generate the nonlinear system corresponding to the Navier-Stokes equations. The solution of the linearized system is carried out using the GMRES method with a least-squares commutator as a preconditioner. Numerical experiments for various test cases illustrate the strength of this new approach.

Nomenclature

$\mathbf{q}$	Vector-valued function representing the dimensionless velocity of the fluid
$\mathbf{f}$	Vector-valued function representing the dimensionless body forces acting on the fluid
p	Scalar function representing the dimensionless pressure
Re	Reynolds number
Ω	Two-dimensional domain used in the simulation of the problem
$\partial\Omega$	Boundary of Ω
$\partial\Omega_D$	Part of $\partial\Omega$ with a Drichlet condition
$\partial\Omega_N$	Part of $\partial\Omega$ with a Neumann condition
x, y	Cartesian coordinates in Ω
$L_2(\Omega)$	Space of square-integrable Lebesgue functions: $L_2(\Omega) := \{\, u : \Omega \mapsto \mathbb{R} \mid \int_\Omega u^2 < \infty \,\}$
$H^1(\Omega)$	Sobolev space $H^1(\Omega) := \{u : \Omega \mapsto \mathbb{R} \mid u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L_2(\Omega)\,\}$

I. Introduction

Over the last two decades, adaptive methods have stirred much interest in the engineering community. For compressible flow applications, such methods are crucial because of the pressing need for accurate computation of shock waves[1–3]. Adaptive methods offer a means of tackling complex flow problems at a reasonable cost and of controlling the accuracy of numerical simulations. Although spectacular results have been achieved for compressible flow problems, less has been accomplished for unsteady viscous incompressible flows. For unsteady problems, we need a fast remeshing algorithm. Mapping of variables from one mesh to

[*]Post-doc Researcher; AIAA Member; Department of Mechanical Engineering; email: msebeida@andrew.cmu.edu
[†]Professor; AIAA Associate Fellow; Department of Mechanical and Aeronautical Engineering; email: davisrl@ucdavis.edu
[‡]Professor; Department of Mathematics; email: freund@math.ucdavis.edu

another should be done in an efficient way, otherwise the vorticity generated during the simulation will be dissipated and the solution quality will deteriorate.

This paper presents an adaptive spatial/temporal finite-element method for unsteady viscous incompressible flow problems. Such flows present special challenges for adaptive methods. Because of the elliptic nature of the Navier-Stokes equations a fully coupled approach is used.

The paper is organized as follows. In the section about governing equations, we describe the approximation of the standard weak formulation using mixed finite elements and Picard's nonlinear iteration. Next, we discuss various ways to deal with the sparse convection tensor, the solution of the linearized system using a suitably preconditioned GMRES algorithm, and the adaptive strategy in space and time. The methodology is then validated by solving problems with experimental data to clearly quantify improvements due to adaptivity. Finally, the method is applied to various flow problems for which experimental data is available.

II. Governing Equations

In this section, we recall the governing equations and the approximation of the standard weak formulation using mixed finite elements. For more details, we refer the reader to Elman et al.[4] and Gresho and Sani[5].

A. Unsteady Incompressible Navier-Stokes Equations

The unsteady incompressible Navier-Stokes equations in their non-dimensional form can be written as follows:

$$\begin{aligned} \frac{\partial \mathbf{q}}{\partial t} + \mathbf{q}\cdot\nabla\mathbf{q} - \frac{1}{Re}\nabla^2\mathbf{q} + \nabla p &= \mathbf{f}, \\ \nabla\cdot\mathbf{q} &= \mathbf{0}. \end{aligned} \tag{1}$$

Here, the relative contributions of convection and diffusion are defined by the *Reynolds number*,

$$Re = \frac{UL}{\nu},$$

where L denotes a characteristic length scale for the domain Ω, U is a reference value for the velocity, and $\nu > 0$ is the *kinematic viscosity* of the fluid. If L and U are suitably chosen, then the condition $Re \leq 1$ means that Eq. (1) is diffusion-dominated and the flow solution can be shown to be uniquely defined. In contrast, if $Re \gg 1$, then the flow problem is convection-dominated. In this case, the convection term, $\mathbf{q}\cdot\nabla\mathbf{q}$, because of its nonlinearity, makes the problem much more complicated.

Eq. (1) is posed on Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

$$\mathbf{q} = \mathbf{w} \quad \text{on} \quad \partial\Omega_D \quad \text{and} \quad \frac{1}{Re}\frac{\partial\mathbf{q}}{\partial n} - \mathbf{n}\,p = \mathbf{0} \quad \text{on} \quad \partial\Omega_N. \tag{2}$$

Here, $\mathbf{n}$ denotes the outward-pointing normal to the boundary. In view of Eq. (2), we need to specify the pressure only on $\partial\Omega_N$. If $\partial\Omega_D = \partial\Omega$, then the pressure solution of the Navier-Stokes problem described by Eqs. (1) and (2) is only unique up to a hydrostatic constant.

B. Weak Formulation

Let

$$H_E^1 := \left\{ \mathbf{u} \in H^1(\Omega)^2 \;\middle|\; \mathbf{u} = \mathbf{w} \text{ on } \partial\Omega_D \right\} \quad \text{and} \quad H_{E_0}^1 := \left\{ \mathbf{v} \in H^1(\Omega)^2 \;\middle|\; \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega_D \right\}$$

denote the usual solution space and test space, respectively. Then, the standard weak formulation of Eqs. (1) and (2) is to find $\mathbf{u} \in H_E^1$ and p $\in L_2(\Omega)$ such that

$$\begin{aligned} \int_\Omega \frac{\partial\mathbf{u}}{\partial t}\cdot\mathbf{v} + \int_\Omega(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + \frac{1}{Re}\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v} + \int_\Omega p\,(\nabla\cdot\mathbf{v}) &= \int_\Omega \mathbf{f}\cdot\mathbf{v} \quad \text{for all} \quad \mathbf{v}\in H_{E_0}^1, \\ \int_\Omega q\,(\nabla\cdot\mathbf{u}) &= 0 \quad \text{for all} \quad q \in L_2(\Omega). \end{aligned} \tag{3}$$

Here $\nabla\mathbf{u}:\nabla\mathbf{v}$ denotes the component-wise scalar product, which in two dimensions is given by

$$\nabla\mathbf{u}:\nabla\mathbf{v} := \nabla\mathbf{u}_x\cdot\nabla\mathbf{v}_x + \nabla\mathbf{u}_y\cdot\nabla\mathbf{v}_y.$$

Note that the nonlinear convection term is represented by the trilinear form $c : H^1_{E_0} \times H^1_{E_0} \times H^1_{E_0} \mapsto \mathbb{R}$ defined as follows:

$$c(\mathbf{z}, \mathbf{u}, \mathbf{v}) := \int_\Omega (\mathbf{z} \cdot \nabla \mathbf{u}) \cdot \mathbf{v}.$$

The remaining terms are described by bilinear forms; for example, the diffusion term is represented by $a : H^1_{E_0} \times H^1_{E_0} \mapsto \mathbb{R}$ defined as follows:

$$a(\mathbf{u}, \mathbf{v}) := \frac{1}{Re} \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}.$$

C. Mixed Finite-Element Approximation

A discrete weak formulation is defined using finite-dimensional subspaces $X^h_0 \subset H^1_{E_0}$ and $M^h \subset L_2(\Omega)$. The discrete problem then is to find $\mathbf{u}_h \in X^h_E$ and $p_h \in M^h$ such that

$$\begin{aligned} \int_\Omega \frac{\partial \mathbf{u}_h}{\partial t} \cdot \mathbf{v} + \int_\Omega (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \cdot \mathbf{v} + \frac{1}{Re} \int_\Omega \nabla \mathbf{u}_h : \nabla \mathbf{v} + \int_\Omega p_h \, (\nabla \cdot \mathbf{v}) &= \textstyle\int_\Omega \mathbf{f} \cdot \mathbf{v} \quad \text{for all} \quad \mathbf{v} \in X^h_0, \\ \int_\Omega q \, (\nabla \cdot \mathbf{u}_h) &= 0 \quad \text{for all} \quad q \in M^h(\Omega). \end{aligned} \tag{4}$$

Implementation entails defining appropriate bases for the finite-element spaces, leading to a nonlinear system of algebraic equations. Using a set of vector-valued basis functions $\{\boldsymbol{\phi}_j\}$, so that

$$\mathbf{u}_h = \sum_{j=1}^{n_u} u_j \boldsymbol{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\partial} u_j \boldsymbol{\phi}_j \quad \text{and} \quad \sum_{j=1}^{n_u} u_j \boldsymbol{\phi}_j \in X^h_0.$$

We fix the coefficients u_j, $j = n_u + 1, n_u + 2, \ldots, n_u + n_\partial$, so that the second term interpolates the boundary data on $\partial\Omega_D$. We also introduce a set of pressure basis functions $\{\psi_k\}$ and set

$$p_h = \sum_{k=1}^{n_p} p_k \psi_k.$$

Using the backward Euler method for the time derivative and substituting into Eq. (4), one obtains the following system of nonlinear equations in tensor notation:

$$\begin{aligned} q_{ij} \left(u^n_j - u^{n-1}_j\right) + c_{ijk} u^n_j u^n_k + a_{ij} u^n_j + b_{il} q_l &= s_{ij} f_j, \\ b_{lj} u^n_j &= 0. \end{aligned} \tag{5}$$

Here, n represents discretized time, $i, j = 1, 2, \ldots, n_u$, and $l = 1, 2, \ldots, n_p$. The operators q_{ij}, a_{ij}, b_{ji}, represent sparse matrices defined as follows:

$$\begin{aligned} q_{ij} \in \mathbb{R}^{n_u \times n_u} &= \text{mass matrix} = \frac{1}{\Delta t} \int_\Omega \boldsymbol{\phi}_j \cdot \boldsymbol{\phi}_i, \\ a_{ij} \in \mathbb{R}^{n_u \times n_u} &= \text{diffusion operator} = \int_\Omega \nabla \boldsymbol{\phi}_j : \nabla \boldsymbol{\phi}_i, \\ b_{lj} \in \mathbb{R}^{n_p \times n_u} &= \text{divergence operator} = \int_\Omega \psi_l (\nabla \cdot \boldsymbol{\phi}_j). \end{aligned}$$

The operator c_{ijk} is a sparse three-dimensional tensor defined as follows:

$$c_{ijk} \in \mathbb{R}^{n_u \times n_u \times n_u} = \text{convection operator} = \int_\Omega (\boldsymbol{\phi}_k \cdot \nabla \boldsymbol{\phi}_j) \cdot \boldsymbol{\phi}_i.$$

Solution of the nonlinear system of equations, Eq. (5), can be carried out efficiently using Picard's method or Newton's method, both of which require iteration. The operators defined above are independent of the solution variables, and they only depend on the test functions and the mesh. If we can store these operators during the nonlinear iteration, then we would be able to save discretization time. The sparse matrices usually do not represent a problem. The sparse tensor is the one that might consume a lot of memory, although it is also sparse. To illustrate this fact, let n_s be the number of points that live in the stencil of any internal point i. We have to store a dense square matrix of size $n_s \times n_s$, while for the other operators we need to store a sparse row vector with only n_u nonzero entries. We have run simulations using up to 10^5 grid points using a personal computer with 2 Gigabyte RAM without facing a problem with storing all the operators. For larger problems, this remains an issue that we will discuss in the following section.

For the finite-element basis functions, we chose to work with stable rectangular elements (Q_2-Q_1), where we use biquadratic approximation for the velocity components, bilinear approximation for the pressure, and stable triangular elements ($P_2 - P_1$), where we use quadratic approximation for the velocity components and linear approximation for the pressure.

D. Nonlinear Iteration

Picard's method is a classical linearization procedure where we start with an 'initial guess' $(\mathbf{u}^{n,0}, p^{n,0})$ and construct a sequence of iterates $\{(\mathbf{u}^{n,m}, p^{n,m})\}$ 'hoping' it converges to the solution of the weak formulation. In this approach we approximate the nonlinear convection term as follows:

$$c(\mathbf{u}^n, \mathbf{u}^n, \mathbf{v}) \approx c(\mathbf{u}^{n.m}, \mathbf{u}^{n,m+1}, \mathbf{v}).$$

We then define the operator

$$c_{ij}^m = c_{ijk}\mathbf{u}_k^{n,m},$$

which is a sparse matrix that we can store easily. However, in order to construct it, we have either to discretize the convection operator using $\mathbf{u}^{n,m}$ at the beginning of every nonlinear iterate $m+1$, or we can store the tensor c_{ijk} and get the matrix c_{ij}^m by a tensor-vector product operation. The first approach consumes time and the second one consumes memory. In order to resolve this issue, we chose to approximate the convection operator as follows:

$$c_{ij}^m \approx \mathbf{u}_i^{n,m} \cdot \int_\Omega \boldsymbol{\phi}_i \cdot \nabla \boldsymbol{\phi}_j.$$

We refer to this approximation as a 'tensor-free' approach. It leads to a sparse matrix given by

$$c_{ij}^* = \int_\Omega \boldsymbol{\phi}_i \cdot \nabla \boldsymbol{\phi}_j,$$

which can be stored easily and possibly save even the time of the tensor-vector multiplication in the second approach mentioned above.

To test this approximation, we carried out two simulations for the unsteady flow over a cylinder at $Re = 1200$. In the first one, we stored the tensor, and in the second, we used the tensor-free approach. We compared the results with experimental data[8]. As Figure 1 shows, the solutions obtained using both approaches are nearly the same.

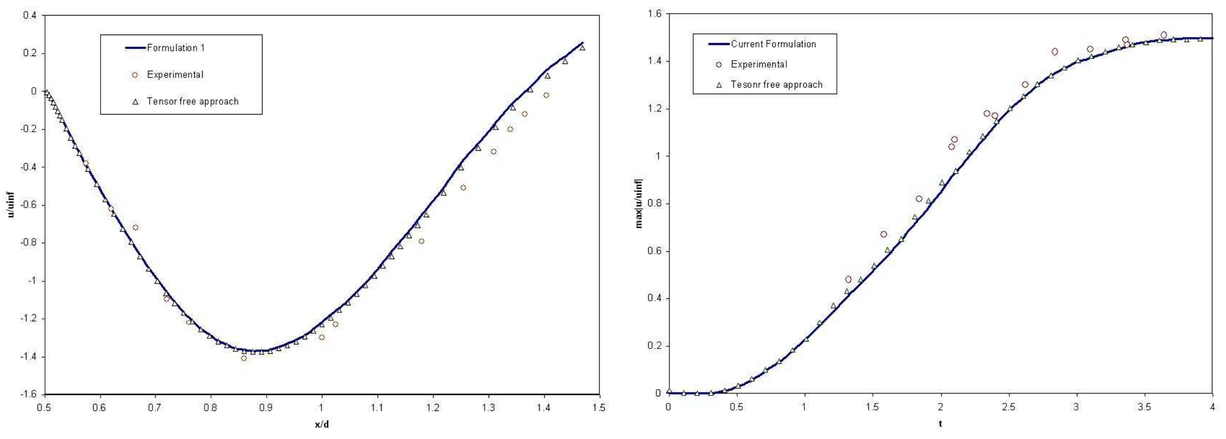
III. Solution of the Linearized System

The linear system we need to solve within each iteration of Picard's method has the following generic form:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f}^* \\ g \end{bmatrix} \tag{6}$$

Such systems are indefinite and call for special iterative techniques in order to achieve convergence[9]. The coefficient matrices are also nonsymmetric. We solve these systems using GMRES, which is a Krylov subspace method for the solution of nonsymmetric systems. However, for GMRES to be viable, we need an efficient preconditioner to ensure GMRES will converge in a reasonable number of iterations. To motivate such a preconditioner, we first look at the block LU-decomposition of the coefficient matrix, K, of Eq. (6):

$$K := \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = LU, \quad \text{where} \quad L := \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \quad \text{and} \quad U := \begin{bmatrix} F & B^T \\ 0 & -BF^{-1}B^T \end{bmatrix}.$$



(a) Velocity distribution in wake (b) Evoultion of max velocity in wake with time

Figure 1. Validating the convection operator approximation using the flow over a cylinder at $Re = 1200$.

If we would choose U as a preconditioner, then all the eigenvalues of the preconditioned matrix $L = KU^{-1}$ are equal to 1 and have Jordan blocks of size at most 2. This implies that the GMRES algorithm would need only two steps to compute the solution to the preconditioned problem, independent of the mesh size or Reynolds number. Unfortunately, using U is prohibitive since it requires the action of the inverse of the Schur complement $S := BF^{-1}B^T$, which is usually too expensive to compute. So we use a suitable approximation to the Schur complement instead. We chose to obtain such an approximation by using the so-called least-squares commutator preconditioner. It approximates the Schur complement matrix as follows:

$$S = BF^{-1}B^T \approx (BQ^{-1}B^T)(BQ^{-1}FQ^{-1}B^T)^{-1}(BQ^{-1}B^T).$$

Here Q is the mass matrix defined in section II-C. The remaining step is to apply the action of the inverse of the preconditioner. For that reason, it is not practical to work with Q^{-1} since it is a dense matrix. Instead, the mass matrix Q is replaced with the diagonal approximation $\hat{Q} = \text{diag}(Q)$. The resulting approximation M_S to S is thus defined as follows:

$$M_S := (B\hat{Q}^{-1}B^T)(B\hat{Q}^{-1}F\hat{Q}^{-1}B^T)^{-1}(B\hat{Q}^{-1}B^T). \tag{7}$$

The inverse of the matrix M_S is given by

$$M_S^{-1} := (B\hat{Q}^{-1}B^T)^{-1}(B\hat{Q}^{-1}F\hat{Q}^{-1}B^T)(B\hat{Q}^{-1}B^T)^{-1} \tag{8}$$

and the right-preconditioned linear system is as follows:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} F & B^T \\ 0 & -M_S \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{u}^* \\ p^* \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix}, \quad \begin{bmatrix} \mathbf{u}^* \\ p^* \end{bmatrix} = \begin{bmatrix} F & B^T \\ 0 & -M_S \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix}. \tag{9}$$

The least-squares commutator preconditioning involves two discrete Poisson solves, matrix-vector products with the matrices B, B^T, F, and (the diagonal matrix) $\hat{Q}^{-1}$, and one discrete convection-diffusion solve.

Note that the use of an inexact solver has no impact on the asymptotic convergence behavior of Picard's method, but it saves computational time. By inexact solver we mean that we run the GMRES algorithm until the residual is reduced by only 2 orders of magnitude, i.e.

$$\frac{||\mathbf{r}||_2}{||\mathbf{r}_0||_2} \leq 10^{-2},$$

where $\mathbf{r}$ represents the residual vector of Eq. (6).

Since we are interested in fast iterative solvers, we solve the discrete Poisson equation using the Preconditioned Conjugate Gradient method (PCG) with SSOR as a preconditioner. We remark that by employing Eisenstat's trick[7], the SSOR preconditioner can be implemented very efficiently. This implementation has nearly the same cost as the Conjugate Gradient (CG) algorithm without preconditioning. For the solution of the discrete convection-diffusion equation we use the Transpose-Free Quasi Minimal Residual (TFQMR) algorithm by Freund[6] with directional block Gauss-Seidel as a preconditioner.

IV. Mesh Adaption

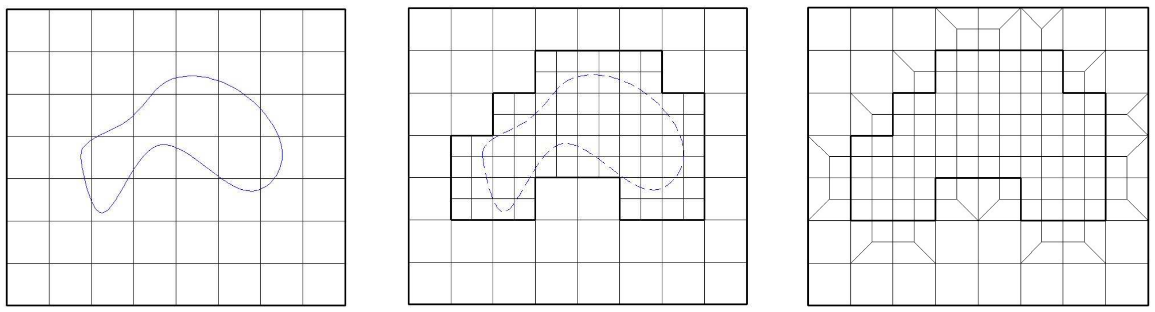
Our solver is coupled with a fast adaptive grid generator. We start the simulation with a base mesh (Mesh A) that has enough points just to define the boundaries of the domain. This mesh is quad-dominant. Moreover, most of the quad elements of this mesh are squares. We refer to the non-square elements as transition elements. For more details about the grid generator, we refer the reader to the previous work of Ebeida and Davis[10].

Algorithm: (Mesh Adaption)
Input: Mesh A, mesh B, solution variables defined at every node of mesh B from the last time step, $n_R \in \mathbb{N}$, $r_L \in \mathbb{R}^{n}_{R}$:=user-specified thresholds for refinement levels.

```
1. Calculate the gradient of the vorticity $|\nabla\omega|$ using mesh B.
2. Map $|\nabla\omega|$ from mesh A to mesh B.
3. For $i = 1, 2, \ldots, n_R$ do:
         Merge transition elements for that refinement level with its corresponding fine region.
         Refine any edge connecting two nodes $(j, k)$ if the following condition is satisfied:
                    $\max\{\, |\nabla\omega|_j,\ |\nabla\omega|_k \,\} > r_L(i)$.
         Interpolate the values of solution variables for the new point from mesh B.
   end
4. Set mesh B to be the new refined mesh.
```

Output: new refined mesh B.

For the first time step, we run the simulation using the base mesh and after convergence, we set mesh B to be the same as mesh A. We then run the mesh adaption technique and repeat the simulation for that time step. So the first time step is simulated twice. The first one gives an approximate solution for the sake of the spatial adaptation, and the second gives the solution using the adapted grid. An example for the output of the mesh adaption algorithm is illustrated in Figure 2, using one refinement level.



(a) Uniform grid and region to be refined

(b) Refinement results in hanging nodes

(c) Final mesh after elimination of all the hanging nodes

Figure 2. One refinement level applied to a subregion of the uniform base mesh.

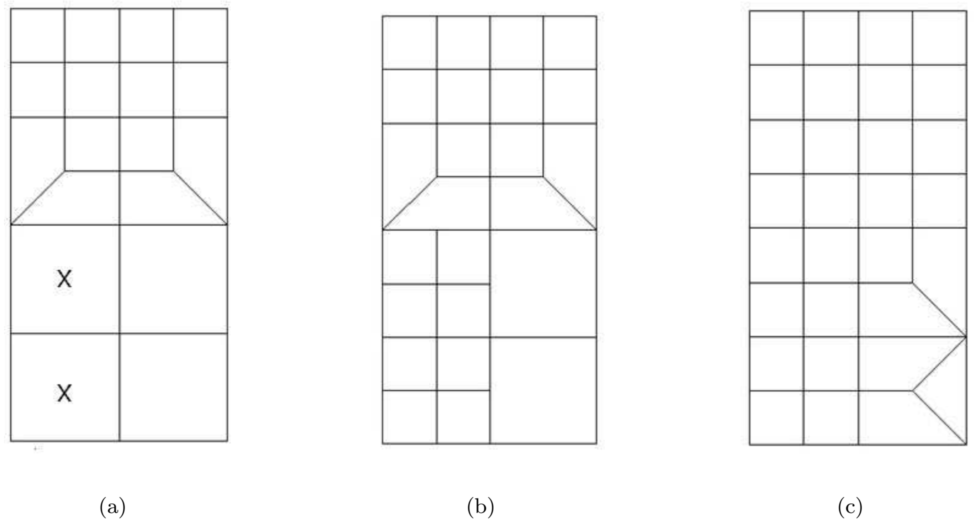


Figure 3. Merging transition elements.

We remark that a transition element is formed by an incomplete refinement of a square element. This incomplete refinement is restricted here to either one or two neighboring edges of the square element. In the first case, the square is transformed into a block of 2 quads and 1 triangle. In the later case, the square is transformed into 3 quads.

Figure 3(a) shows that the transition elements will always lie between two levels of square elements. So if we decide to refine a square element in the coarse region (marked with an '$\times$'), a mesh quality problem will result as illustrated in 3(b). The technique used to fix this problem is to merge all the transition elements for that level. In other words, each block of transition elements is transformed into 4 square elements, as shown in Fig 3(c).

This algorithm will be employed for refining and coarsening the mesh elements from the last refinement step, although it is always refining the base mesh. Interpolating the variables from the last refined mesh to the new refined one is crucial for preventing the dissipation of the vortices within the flow. Currently, we do not adapt the mesh at every time step, but rather we chose to run this algorithm once within a non-dimensional time interval of length 0.1.

V. Time-Step Control

In this section, we discuss an algorithm for the adaption of the time step, following the technique described in[11]. Once the Reynolds number is specified, the grid is generated, and the test functions are chosen, the solution of Eqs. (1) and (2) depends only on the time step. Let h denote the time step at time t. Let $\mathbf{u}(t;h)$ be the computed approximation of the exact velocity $\mathbf{u}^*(t)$. If we know the order p of the method used in approximating the time derivatives, then we have the following error estimate:

$$\mathbf{u}\left(t;\frac{h}{2}\right) - \mathbf{u}^*(t) \approx \frac{\mathbf{u}(t;h) - \mathbf{u}\left(t;\dfrac{h}{2}\right)}{2^p - 1}.$$

At each time step, we start with an initial guess $H > 0$ for the desired time step h and calculate the quantity

$$r_H := \left(\frac{2^p}{2^p - 1} \frac{\left\| \mathbf{u}(t_0 + H; H) - \mathbf{u}\left(t_0 + H; \dfrac{H}{2}\right) \right\|_2}{\epsilon} \right)^{\dfrac{1}{p+1}} \approx \frac{H}{h},$$

which approximates the ratio H/h. If $r_H \gg 2$, then the error is large. On the other hand, if r_H is too small then we might want to increase the time step in order to reduce the simulation time. In the numerical experiments reported in this paper, we chose to work with $\epsilon = 10^{-6} \times ||\mathbf{u}(t_0)||_2$ and an acceptable range of $1.0 < r_H < 5.0$. If a value of r_H exists in that range, we proceed to the next time step. If not, we set $H = 2h$ and re-simulate the last time step. Figure 4 shows the fast evolution of the time step H for a flow over a cylinder at $Re = 1200$ with impulsive initial condition. The algorithm started with $H = 2.6 \times 10^{-4}$ and at approximately $t = 6.0$ the algorithm set H to be 0.077.

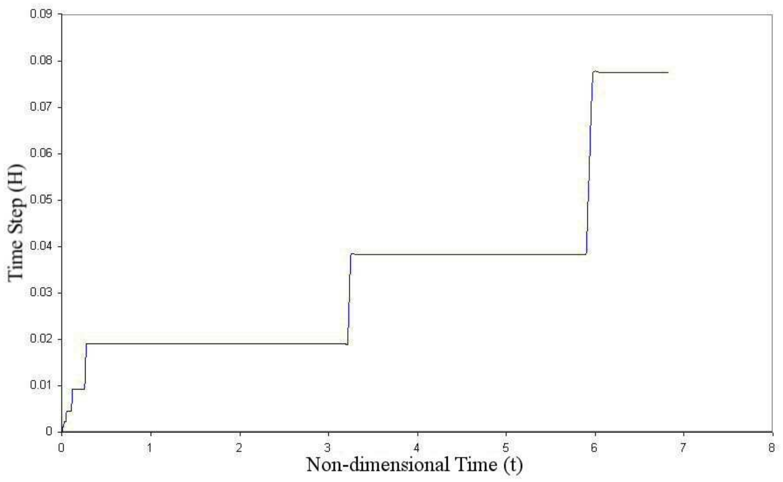


Figure 4. Evolution of time steps at the beginning of the simulation of flow over a cylinder at $Re = 1200$ with impulsive initial condition.

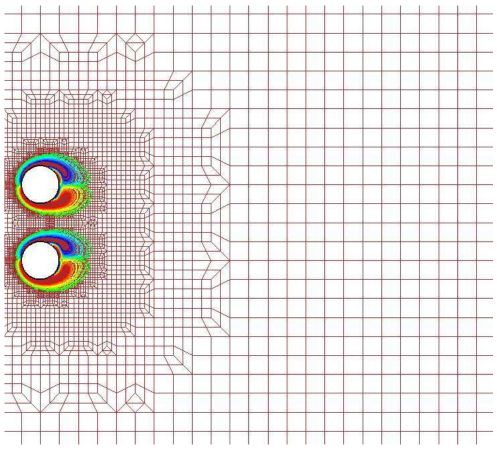
VI. Test Cases

A. Unsteady Laminar Flow Over Two Vertical Cylinders at $Re = 200$

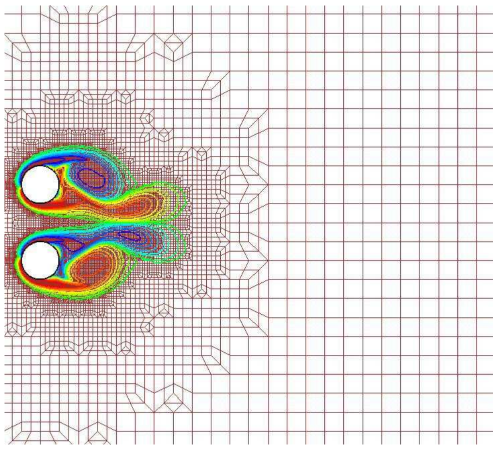
The goal of this simulation is to illustrate the ability of our code to simulate flow over multiple objects and to accurately resolve the interaction of the vortices generated around each cylinder. Figure 5 shows the interaction of the vortex shedding around each cylinder and how the vortices impact with each other at the centerline preserving the symmetry of the flow. Vorticity contours overlaid onto the computational grid are shown in Figure 5 for different instants in time (in seconds) from $t = 1$ to $t = 20$. At $t = 1$, the viscous flow of each cylinder is just beginning to shed. At $t = 4$, the vortices near the centerline begin to interact. The adaptation algorithm has correctly identified the vortices along with the interaction region and has refined the grid in those regions. The contours and grid shown at $t = 8$, 12, 16, and 20 further illustrate how the adaptation algorithm has identified the multiple vortices in the domain and kept the grid refined in those regions. It should also be noted that the solution and grid remained symmetric about the centerline between the two cylinders indicating high spatial and temporal accuracy.

B. Unsteady Laminar Flow Over a NACA 0012 Airfoil at $Re = 800$, $\alpha = 20°$

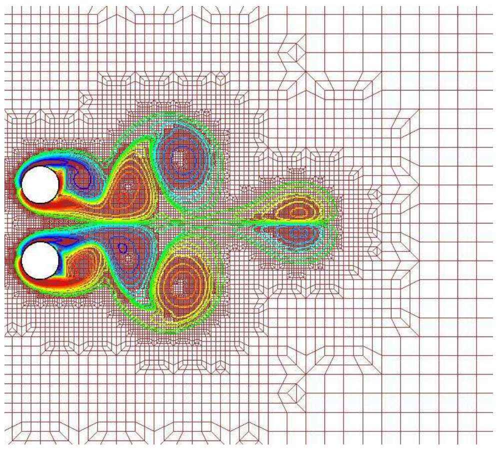
Figure 6 shows the evolution of the vorticity and the grid produced during a simulation of the unsteady incompressible flow around a NACA 0012 airfoil at $Re = 800$ and an angle of attack of 20 degrees. We picked this case to test the capability of capturing the starting vortex and the steady vortex shedding. Figure 6(a) shows the starting vortex generated near the trailing edge. Figure 6(b) illustrates how the grid is adapted to track the starting vortex while it moves toward the outflow boundary. Figure 6(c) shows the starting vortex while crossing the outflow boundary in a smooth way. Figures 6(d) and 6(e) display the evolution of the airfoil vortex street.



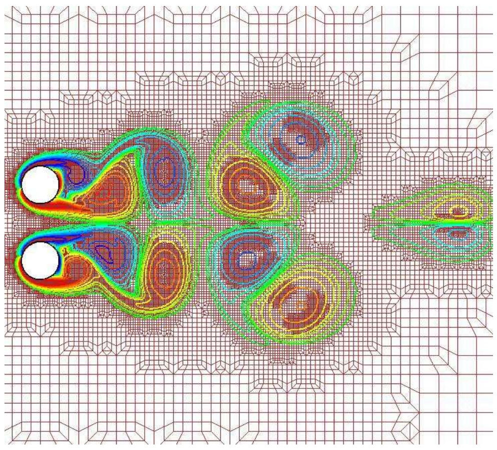
(a) t = 1.00



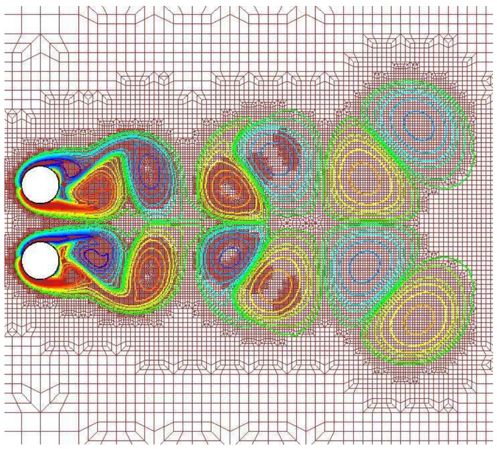
(b) t = 4.00



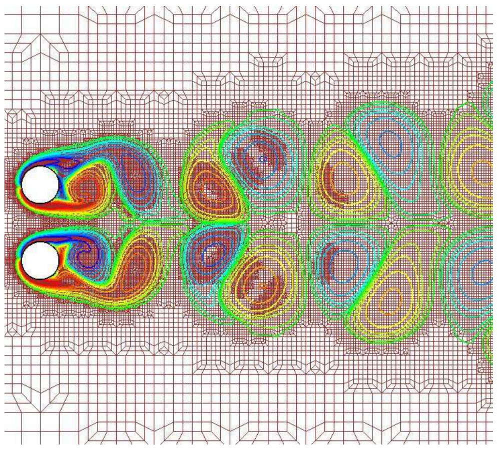
(c) t = 8.00



(d) t = 12.00

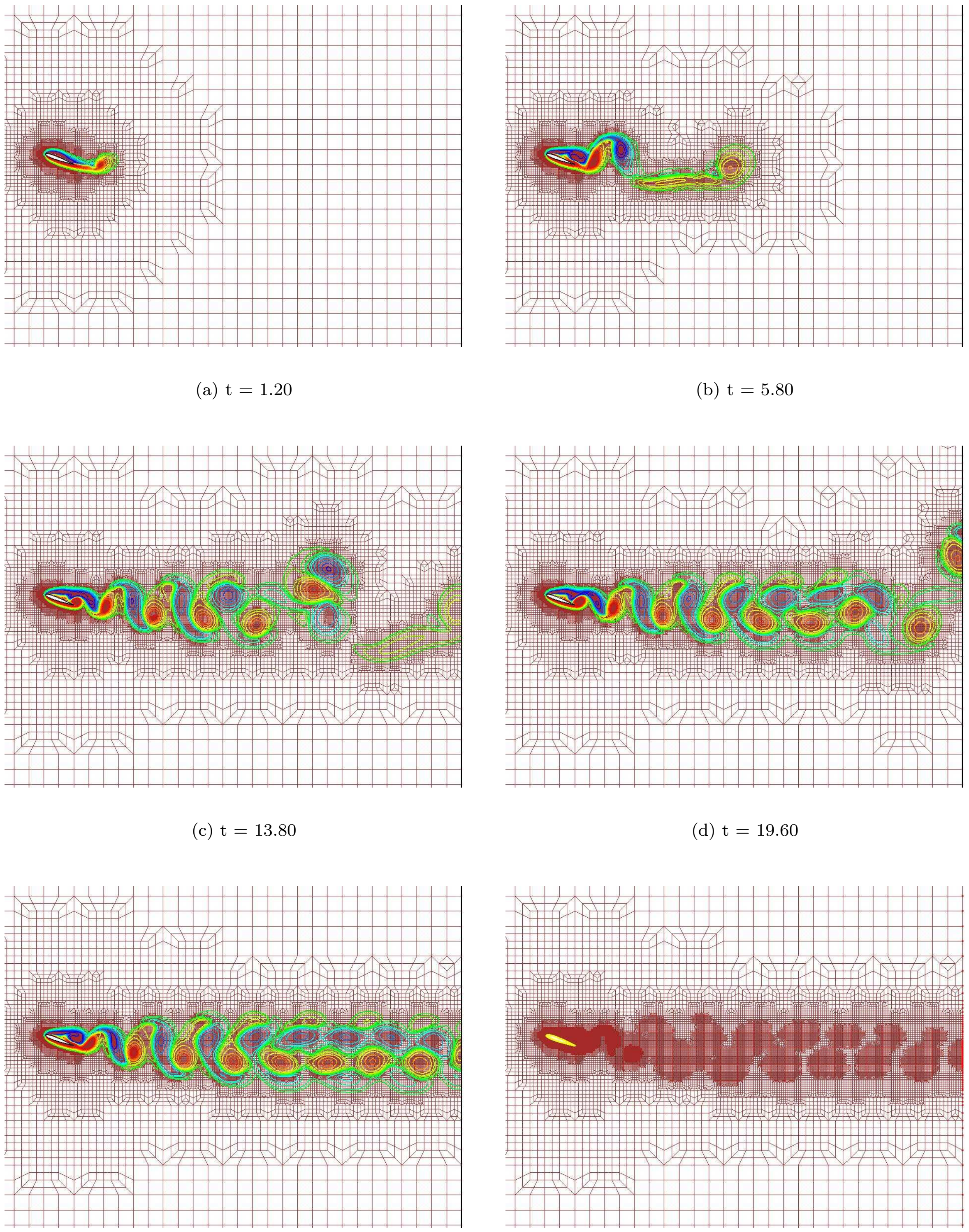


(e) t = 15.80



(f) t = 20.00

Figure 5. Evolution of grid and vorticity contours for flow over two vertical cylinders at $Re = 200$.



(a) t = 1.20

(b) t = 5.80

(c) t = 13.80

(d) t = 19.60

(e) t = 27.80

(f) t = 27.80 - Grid Only

Figure 6. Evolution of grid and vorticity contours for flow over a NACA 0012 airfoil at $Re = 800$, $\alpha = 20°$**.**

VII. Summary and Future Work

We presented a new Galerkin finite-element technique for the simulation of unsteady incompressible flows. This technique employs a fast dynamic remeshing procedure to adaptively redistribute the grid points based on the gradient of the vorticity calculated using the solution variables from the previous time step. A varying time step is chosen based on an error estimation algorithm. Our technique allows the use of larger domains without dramatically increasing the number of grid points. Efficient Galerkin finite-element discretization is accomplished through the storage of the different operators associated with the Navier-Stokes equations as sparse tensors. The use of fast iterative solvers, such as preconditioned GMRES, CG, and TFQMR, is crucial for incompressible flows. Our technique still needs to be tested using unsteady flows at high Reynolds numbers. Also, the extension to three-dimensional flows needs to be investigated.

References

[1]Flaherty, I. E., Paslow, P. J., Sheppard, M. S., and Vasilakis, J. D. (eds), *Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1989.

[2]Babuška, I., Zienkiewicz, O. C., Gago, J., and de A. Oliveira, E. R. (eds), *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, Wiley, Chichester, England, UK, 1986.

[3]Zienkiewicz, O. C., Gago, J. P., and Kelly, D. W., *The Hierarchical Concepts in Finite Element Analysis*, Comp. Struct., Vol. 16, 1983, pp. 53–65.

[4]Elman, H., Silvester, D., and Wathen A., *Finite Elements and Fast Iterative Solvers*, Oxford University Press, 2005.

[5]Gresho, P. M. and Sani, R. L., *Incompressible Flow and The Finite Element Method*, John Wiley and Sons 1998.

[6]Freund, R. W., *A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems*, SIAM J. Sci. Comput., Vol. 14, pp. 470–482, 1993.

[7]Eisenstat, S. C., *Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods*, SIAM J. Sci. Statist. Comput., Vol. 2, pp. 1–4, 1981.

[8]Nagata, H., Funada, H., and Matsui, T., *Unsteady Flows in the Vortex Region Behind a Circular Cylinder Started Impulsively: 2nd Report. Velocity Fields and Circulations*, Bull. JSME., Vol. 28, pp. 2608–2616, 1985.

[9]Saad, Y., *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, PA, 2003.

[10]Ebeida, M. and Davis, R., *Fast Adaptive Hybrid Mesh Generation Based On Quad-Tree Decomposition*, AIAA Paper 2008-4141, 38th Fluid Dynamics Conference and Exhibit, 2008.

[11]Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis*, 3rd edition, Springer-Verlag, New York, Berlin, Heidelberg, 2002.

[12]Turek, S., *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[13]Engelman, M. S. and Jamnia, M.-A., *Transient Flow Past a Circular Cylinder: A Benchmark Solution*, Int. J. Num. Meth. Fluids, Vol. 11, pp. 985–1000, 1990.

[14]Soto, O., Löhner, R., and Cebral, J., *An Implicit Monolithic Time Accurate Finite Element Scheme for Incompressible Flow Problems*, AIAA Paper 2001-2616, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, CA, 2001.

[15]Norberg, C., *Fluctuation Lift on a Circular Cylinder: Review and New Measurements*, J. Fluids Struct., Vol. 17, pp. 57–96, 2003.

[16]Löhner, R., *A Fast Finite Element Solver for Incompressible Flows*, AIAA Paper 1990-398, 28th Aerospace Sciences Meeting, Reno, NV, 1990.

[17]Tezduyar, T. E., Mittal, S., Ray, S. E., and Shih, R., *Incompressible Flow Computations With Stabilized Bilinear and Linear Equal-Order Interpolation Velocity-Pressure Elements*, Comput. Methods Appl. Mech. Engrg., Vol. 95, pp. 221–242, 1992.