Simulation of unsteady incompressible flows using temporal adaptation and dynamic remeshing

Mohamed S. Ebeida[a*], Roger L. Davis[b], and Roland W. Freund[c]

[a] *Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA*;
[b] *Department of Mechanical and Aeronautical Engineering, University of California, Davis, CA, USA*;
[c] *Department of Mathematics, University of California, Davis, CA, USA*

An implicit finite-element flow solver based on the Galerkin finite-element method is employed to study unsteady laminar flow past single and multiple objects. A fast dynamic remeshing technique is used to control the distribution of the mesh nodes during the unsteady simulation, thus minimizing (or even eliminating) the need for adding artificial dissipation terms. The quad-dominant mesh generator coupled with this solver is based on a Cartesian mesh with a conforming spatial decomposition. The dynamic remeshing technique preserves the quality of the elements during the refinement/coarsening process. A mixed Galerkin finite-element discretization is used to generate the nonlinear system corresponding to the Navier-Stokes equations. After approximating the time derivative by means of finite differences and applying Picard's iteration, one obtains linear systems of equations that need to be solved at each time step. The GMRES method combined with a least-squares commutator as a preconditioner is employed for the solution of these linear systems. The time step is controlled and adapted using an error estimation of the computed flow variables with respect to time. Results of several numerical simulations are presented and validated against experimental and numerical studies to demonstrate the viability of this new approach.

Keywords: incompressible flow; unsteady flow; Galerkin finite-element; stable discretization; dynamic remeshing; time-step control; Krylov method; GMRES; TFQMR; implicit method

1 Introduction

Incompressible flows play an important role in many areas, including aerodynamics, hydrodynamics, and the simulation of manufacturing processes. Over the years, many techniques for the numerical solution of the incompressible Navier-Stokes equations have been developed; see, e.g., Gunzburger and Nicolaides (1993), Gresho and Sani (1998a,b), Hafez (2003), Elman *et al.* (2005). In particular, the Galerkin finite-element method provides a rigorous and robust means for devising high-order algorithms for unstructured grids. Methods of this type can be employed to perform highly accurate simulations of flows around complex geometries, using grids that are readily generated. Such high-order algorithms are attractive due to their ability to produce a smaller error using relatively coarse meshes (Fidkowski *et al.* (2005), Nejat and Ollivier-Gooch (2008)). However, high-order methods are more expensive than conventional second-order methods, on a per degree of freedom basis (Adjerid *et al.* (1995), Ainsworth (2004)). At the same time, second-order techniques have to employ denser meshes than higher-order methods in order to achieve comparable accuracy. Thus, for second-order algorithms, the use of adaptive grids is crucial in order to reduce computational costs.

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; see, e.g., Babuška *et al.* (1986), Flaherty *et al.* (1989), Eriksson *et al.* (1998), Rannacher (1999). Adaptive methods offer a means for tackling

*Corresponding author. Email: msebeida@andrew.cmu.edu

complex flow problems at a reasonable cost and, at the same time, controlling the accuracy of the numerical simulations. While spectacular results have been achieved for compressible flow problems, progress for unsteady viscous incompressible flows has been more modest; see, e.g., Hétu and Pelletiert (1992). For unsteady problems, a fast dynamic remeshing algorithm is required. Mapping of variables from one mesh to another should be conservative and done in an efficient way, otherwise the vorticity generated during the simulation will be dissipated and the solution quality will deteriorate.

The method presented in this paper is based on a fast dynamic remeshing technique coupled to a finite-element solver for unsteady viscous incompressible flows. The time-step during the unsteady simulation is controlled and hence adapted via an error estimation procedure. The proposed adaptive methodology is designed to transform an existing finite-element flow solver into an adaptive one. To this end, the spatial and temporal adaptivity are kept completely separate from the flow solver.

The paper is organized as follows. In Section 2, we review the governing equations and the standard weak formulation using mixed finite elements and Picard's iteration. Also, we discuss various ways to handle the sparse convection tensor. In Section 3, we discuss the solution of the resulting nonsymmetric linear systems by means of a suitably preconditioned GMRES algorithm and compare the convergence of various iterative methods that are employed to efficiently implement the preconditioner. The dynamic remeshing technique is presented briefly in Section 4, and the control of the time step during the unsteady simulation is discussed in Section 5. The methodology is validated in Section 6 by solving unsteady problems around single and multiple bodies to clearly quantify improvements due to adaptivity. Finally, Section 7 provides some concluding remarks.

2 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 Gresho and Sani (1998a) and Elman *et al.* (2005).

2.1 *Unsteady incompressible Navier-Stokes equations*

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

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

Equation (1) is posed on a given domain Ω, together with suitable conditions on the boundary $\partial\Omega$ of Ω. We assume that these boundary conditions are of the form

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

where $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. Here, $\mathbf{n}$ denotes the outward-pointing normal to the boundary and $\mathbf{w}$ is assumed to be a given function on the Dirichlet part, $\partial\Omega_D$, of the boundary. Note that we need to specify the pressure p only on the Neumann part, $\partial\Omega_N$, of the boundary. We remark that in the case of pure Dirichlet boundary conditions, i.e. $\partial\Omega_D = \partial\Omega$, the pressure part p of the solution of the Navier-Stokes problem described by equations (1) and (2) is only unique up to a hydrostatic constant.

The major difficulty in solving equation (1) lies in the coupling of the velocity and the pressure through the incompressibility constraint. Unlike the case of compressible flows, where the pressure is obtained from an equation of state, the pressure field in incompressible flows establishes itself instantaneously and must therefore be integrated implicitly in time. This reflects the assumption that the speed of sound through incompressible flows is infinite.

2.2 *Weak formulation*

Let

$$L_2(\Omega) := \left\{ u : \Omega \mapsto \mathrm{R} \;\middle|\; \int_\Omega u^2 < \infty \right\}$$

denote the space of square-integrable Lebesgue functions, and let

$$H^1(\Omega) := \left\{ u : \Omega \mapsto \mathrm{R} \;\middle|\; u,\, \frac{\partial u}{\partial x},\, \frac{\partial u}{\partial y} \in L_2(\Omega) \right\}$$

denote the corresponding Sobolev space of functions that have square-integrable derivatives. The usual solution space and test space are given by

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

respectively. Then, the standard weak formulation of equations (1) and (2) is the problem to find $\mathbf{u} \in H^1_E$ and $\mathrm{p} \in L_2(\Omega)$ such that

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

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 in equation (3), the nonlinear convection term is represented by the trilinear form

$$c : H^1_E \times H^1_E \times H^1_{E_0} \mapsto \mathrm{R}. \quad c(\mathbf{z}, \mathbf{u}, \mathbf{v}) := \int_\Omega (\mathbf{z} \cdot \nabla \mathbf{u}) \cdot \mathbf{v}.$$

The other terms in equation (3) are described by bilinear forms; for example, the diffusion term is represented by

$$a : H^1_E \times H^1_{E_0} \mapsto \mathrm{R}, \quad a(\mathbf{u}, \mathbf{v}) := \frac{1}{Re} \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}.$$

2.3 *Mixed finite-element approximation*

A discrete approximation of the weak formulation is obtained by replacing the infinite-dimensional spaces H^1_E, $H^1_{E_0}$, and $L_2(\Omega)$ in equation (3) by suitably chosen finite-dimensional subspaces $X^h_E \subset H^1_E$, $X^h_0 \subset H^1_{E_0}$, 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

$$\begin{aligned} \int_\Omega \frac{\partial \mathbf{u}_h}{\partial t} \cdot \mathbf{v} + \int_\Omega (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \cdot \mathbf{v} + \frac{1}{Re} \int_\Omega \nabla \mathbf{u}_h : \nabla \mathbf{v} + \int_\Omega p_h (\nabla \cdot \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \quad \text{for all} \quad \mathbf{v} \in X_0^h, \\ \int_\Omega q (\nabla \cdot \mathbf{u}_h) = 0 \quad \text{for all} \quad q \in M^h(\Omega). \end{aligned} \tag{4}$$

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

$$\mathbf{u}_h = \sum_{j=1}^{n_u} u_j \boldsymbol{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\partial} u_j \boldsymbol{\phi}_j \quad \text{and} \quad \sum_{j=1}^{n_u} u_j \boldsymbol{\phi}_j \in X_0^h.$$

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

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

Employing the backward Euler method to approximate the time derivative in equation (4), one obtains a system of of nonlinear equations, which in tensor notation is given as follows:

$$\begin{aligned} q_{ij} \left(\frac{u_j^n - u_j^{n-1}}{\Delta t} \right) + c_{ijk} u_j^n u_k^n + a_{ij} u_j^n + b_{il} q_l = q_{ij} f_j, \\ b_{lj} u_j^n = 0, \end{aligned} \tag{5}$$

where $i, j = 1, 2, \ldots, n_u$ and $l = 1, 2, \ldots, n_p$. Moreover, the superscript n is used to donate the solution at the n-th time step. The operators q_{ij}, a_{ij}, b_{ji} represent sparse matrices defined as follows:

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

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

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

Solution of the nonlinear system of equations (5) can be carried out efficiently using Picard's method or Newton's method, both of which are iterative procedures. The operators defined above are independent of the solution variables. They only depend on the test functions and the employed mesh. Hence these operators are calculated once and updated for each new mesh. Storage of these operators might introduce a problem for large mesh sizes. In this case, the 3D 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. For this

node, the 3D tensor stores a dense square matrix of size $n_s \times n_s$, while the other operators only store a sparse row vector with only n_u nonzero entries. In our simulations, the number of nodes in the mesh can go up to 10^5 using a personal computer with 2 Gigabyte RAM without facing a storage problem with storing all the operators. For larger problems, this remains an issue that we will discuss in the following section.

In order to avoid stability issues, we chose to work with mixed finite elements for the velocity components and the pressure. More precisely, for rectangular elements, we use biquadratic approximations (Q_2) for the velocity components and bilinear approximations (Q_1) for the pressure field. A similar approach is followed for the triangular elements, where we use quadratic approximations (P_2) for the velocity components and linear approximations (P_1) for the pressure field. Isoparametric mapping is employed to enable the integration over the corresponding reference elements. Gauss quadrature is then used to evaluate the integrals within the various operators. For more details about the construction of the Galerkin system, we refer the reader to Ebeida (2008).

2.4 *Nonlinear iteration*

Picard's method is a classical linearization procedure that starts with an 'initial guess', $(\mathbf{u}^{n,0}, p^{n,0})$, and then constructs a sequence of iterates $\{\,(\mathbf{u}^{n,m}, p^{n,m})\,\}$ until 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 can be stored 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 m-th nonlinear iteration, 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:

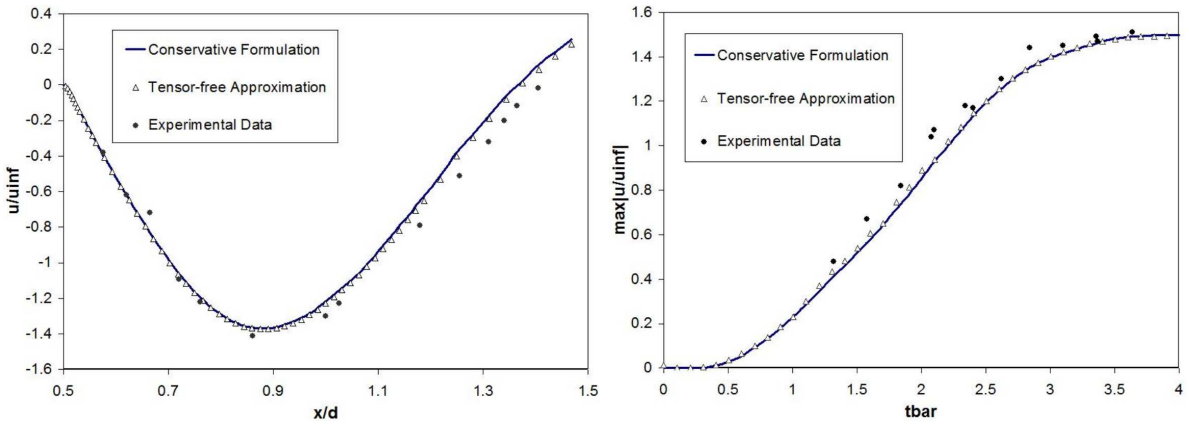
$$c_{ij}^m \approx \mathbf{u}_i^{n,m} \cdot \int_\Omega \boldsymbol{\phi}_i \cdot \nabla \boldsymbol{\phi}_j.$$

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 reduces the storage requirements significantly and saves the computational time consumed by the tensor-vector multiplication required during the conservative approach mentioned above.

We validated this non-conservative approximation using the unsteady flow over a cylinder at Reynolds number $Re = 1200$ with an impulsive initial condition, Figure 1 shows the velocity distribution in the wake at non-dimensional time $\bar{t} := tU/L = 2.9$ and the evolution of the maximum velocity in the wake during the first four non-dimensional time units $\bar{t}$. Here, L and U denote the characteristic length and velocity, respectively, for this flow problem. The results obtained using the tensor-free approximation are compared to the results obtained using the conservative formulation as well as the experimental data published by Nagata *et al.* (1985). The comparison shows that the solutions obtained using both approaches are nearly the same and in good agreement with the experimental data.



(a) Velocity distribution in wake at $tU/L = 2.9$

(b) Evolution of max velocity in wake with time

Figure 1. Validating the convection operator approximation using the flow over a cylinder at $Re = 1200$ using the results obtained from the conservative formulation and the experimental data obtained by Nagata *et al.* (1985).

3 Solution of the linearized system

In this section, we discuss the solution of the nonsymmetric linear systems that arise within the nonlinear iteration.

3.1 *Preconditioning of the nonsymmetric linear systems*

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

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

We remark that the submatrix F is nonsymmetric. Thus the coefficient matrix of the linear system in equation (6) is also nonsymmetric. Moreover, the matrix is indefinite, and iterative techniques combined with efficient preconditioning are required to achieve satisfactory convergence; see, e.g., Saad (2003). In our simulations, we employ preconditioned GMRES (Saad and Schultz (1986)), which is a Krylov subspace method for the solution of general nonsymmetric linear systems.

To motivate our choice of preconditioning, we first look at the block LU-decomposition of the coefficient matrix, K, of equation (6):

$$K := \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = L_0 U_0, \quad \text{where} \quad L_0 := \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \quad \text{and} \quad U_0 := \begin{bmatrix} F & B^T \\ 0 & -BF^{-1}B^T \end{bmatrix}.$$

If we would choose U_0 as a preconditioner, then all the eigenvalues of the preconditioned matrix $L = KU_0^{-1}$ are equal to 1 and have Jordan blocks of size at most 2. This implies that the preconditioned GMRES algorithm would never need more than two iterations to compute the solution of the linear system in equation (6). Unfortunately, using U_0 is not an option since it requires the action of the inverse of the Schur complement $S := BF^{-1}B^T$, which is usually too expensive to compute. Instead, we employ a suitable approximation to the Schur complement. To this end, we use the so-called least-squares commutator preconditioner (Elman *et al.* (2007)).

It approximates the Schur complement S as follows:

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

Here, $Q = [q_{ij}]$ is the mass matrix defined in Section 2. The remaining step is to apply the action of the inverse of the preconditioner. For this step, 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 $\hat{Q} = \mathrm{diag}(Q)$. The resulting approximation M_S to S is thus defined as follows:

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

The inverse of the matrix M_S is given by

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

and the right-preconditioned linear system is as follows:

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

The implementation of this Least-Squares Commutator Preconditioner (LSCP) involves two discrete Poisson solves, matrix-vector products with the sparse 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.

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

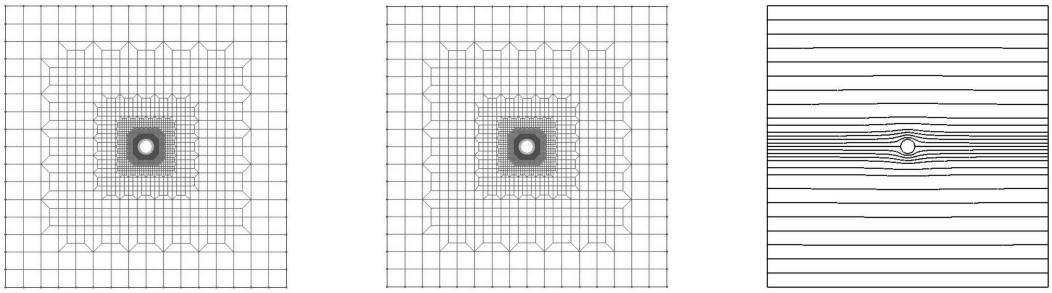
where $\mathbf{r}$ represents the residual vector of equation (6).

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 a trick due to Eisenstat (1981), 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 (1993) with directional block Gauss-Seidel as a preconditioner.

3.2 *Test problems for the iterative solvers*

The LSCP involves solving two discrete Poisson equations with a symmetric positive definite coefficient matrix and a convection-diffusion equation with a nonsymmetric matrix. Hence, different methods are employed for solving these two different systems. In this section, we use two test problems to illustrate some properties of the iterative methods that can be used to solve the linear systems arising within the application of the LSCP preconditioner.

In particular, our gaol is to show the dependence of the convergence rate of these methods on the number of nodes in the mesh used in the discretization. In the case of solving the convection-diffusion equation, we also want to check the effect of the Reynolds number on the convergence of the utilized iterative method.



(a) Mesh 1 (8,604 Nodes) (b) Mesh 3 (101,702 Nodes) (c) Final solution

Figure 2. Potential flow over a circular cylinder using the stream function.

3.2.1 A symmetric positive definite linear system

Consider the problem

$$-\nabla^2 \psi = 0,$$

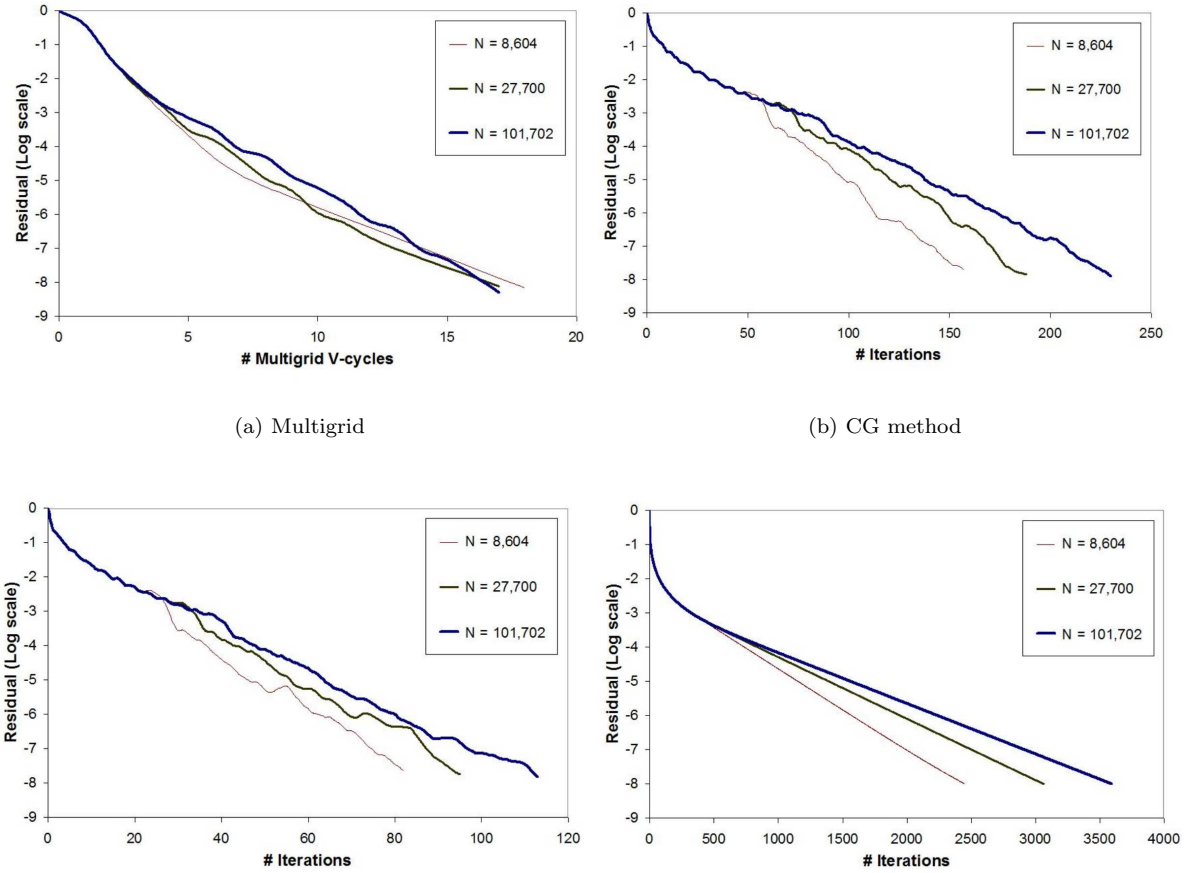
which arises in the simulation of inviscid flow around a circular cylinder using the stream function. The corresponding linear system is symmetric positive definite. We used three meshes (with $8,604$, $27,700$, $101,702$ nodes) to discretize this equation. Figure 2 shows the smallest and the largest of the three meshes employed, as well as the final solution. When creating the meshes, more nodes were added near the boundary representing the circular cylinder. This is the reason why the two meshes shown in Figure 2 look the same in the far field. The corresponding discrete Poisson problem is solved using multigrid (MG), Conjugate Gradients (CG), Preconditioned Conjugate Gradients (PCG) with SSOR as a preconditioner, and the classical Gauss-Seidel iterative method. As shown in Figure 3, the convergence rate of MG was not affected by increasing the number of nodes in the mesh, which is one of the key properties of MG. CG on the other hand is slightly affected by the increased number of nodes. PCG with SSOR preconditioning nearly cuts the number of iterations in half, without essentially any additional costs due to the use of Eisenstat's trick. Comparing MG and CG to the classical Gauss-Seidel method shows the superiority of these non-stationary methods.

3.2.2 A nonsymmetric linear system

Consider the problem

$$u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} - \frac{1}{Re}\nabla^2 \rho = 0.$$

This is a convection-diffusion problem where the wind velocity is specified everywhere in a box domain with Drichlet and Neumann boundary conditions. The corresponding linear system is nonsymmetric. We used three meshes (with $8,673$, $17,381$, $34,793$ nodes) to discretize this equation. When creating these meshes, more nodes were added near the boundary representing the solid surfaces. Figure 4 shows the smallest of the three meshes, as well as the final solution for Reynolds numbers $Re = 10^2$, $Re = 10^3$, and $Re = 10^4$. For this case, we want to check the influence of increasing the number of mesh points as well as increasing the value of Re. Figure 5 shows the convergence rates for the different iterative methods at a fixed Reynolds number while increasing the number of the nodes in the mesh. On the other hand, Figure 6 illustrates the effect of varying the Reynolds number on the convergence rates for the different iterative methods using the same mesh. Note that successive line over-relaxation (SLOR) was implemented in MG as a smoother instead of the damped Jacobi method utilized while solving



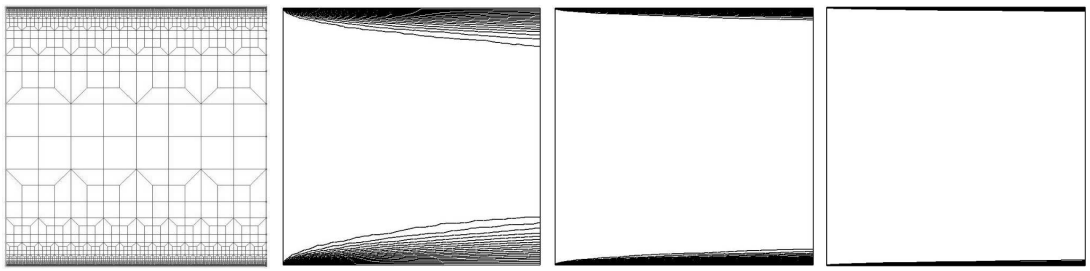
(a) Multigrid

(b) CG method

(c) PCG method with SSOR

(d) Gauss-Seidel method

Figure 3. Convergence history for different methods applied to the discrete Poisson equation.



(a) Mesh with 8673 nodes

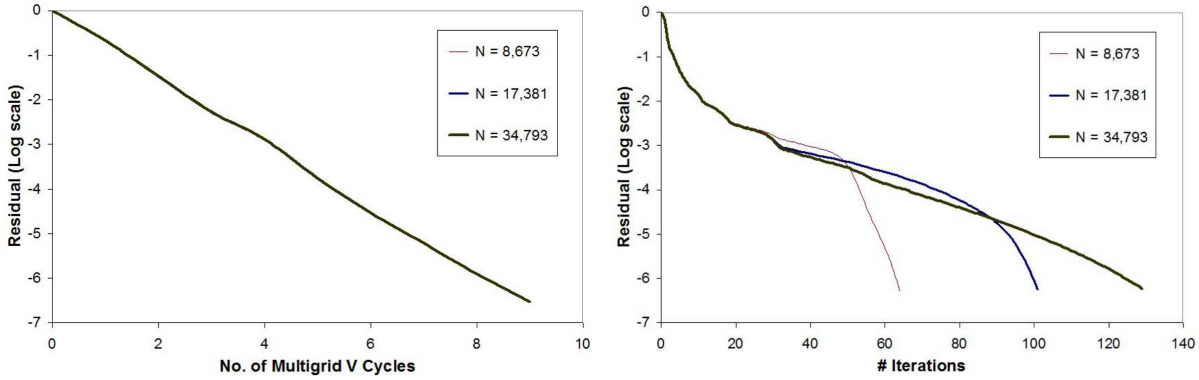
(b) $Re = 10^2$

(c) $Re = 10^3$

(d) $Re = 10^4$

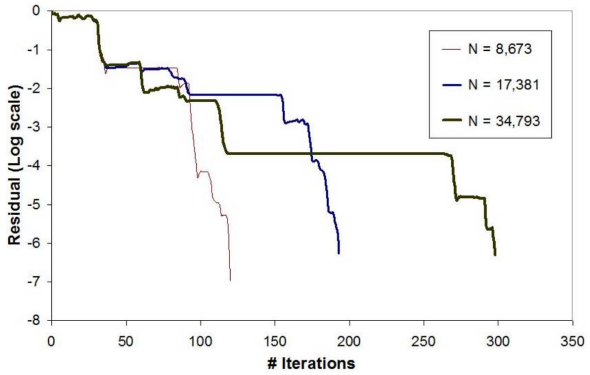
Figure 4. Mesh + Solution of the discrete convection-diffusion problem.

the symmetric positive definite system. SLOR was also utilized in GMRES and TFQMR as a preconditioner. The results show that MG is the optimal method to solve such a problem. The convergence rate of MG was slightly affected by increasing the Reynolds number and was not affected at all by increasing the number of nodes in the mesh. As one would expect, GMRES performs better compared to TFQMR but as the number of nodes increases, the number of iterations increases as well.

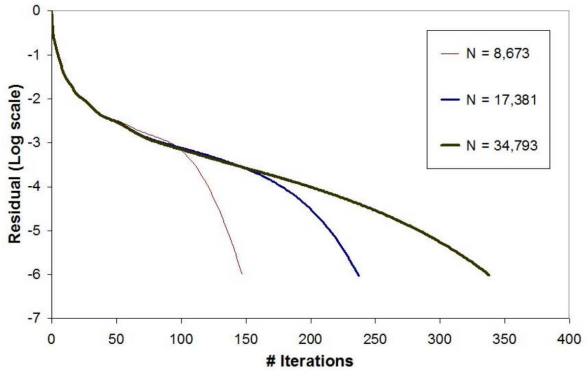


(a) Multigrid

(b) GMRES



(c) TFQMR



(d) SLOR

Figure 5. Convergence history for different methods applied to the discrete convection-diffusion equation at $Re = 10^4$.

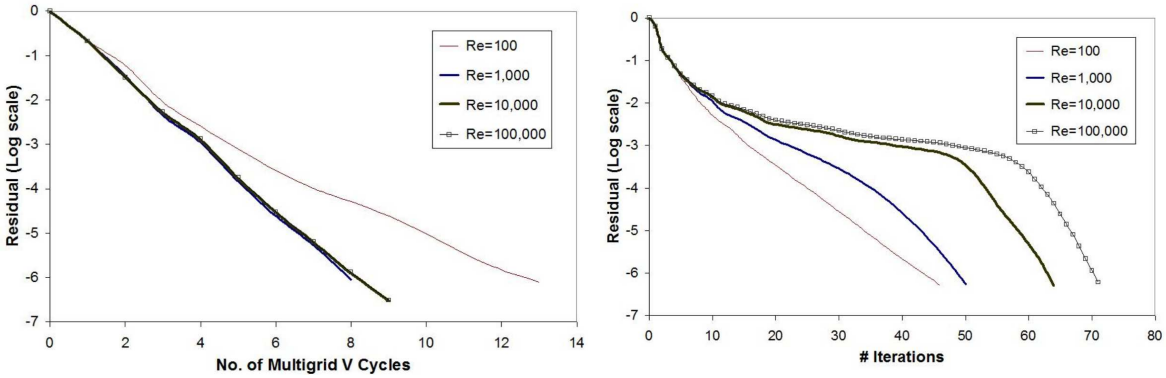
4 Adaptive dynamic remeshing

Our solver is coupled with a fast adaptive dynamic remeshing technique. We start the simulation with a base mesh $\mathcal{M}_b$, which has just enough nodes to define the boundaries of the domain. This mesh is crack-free and quad-dominant, and most of the quadrilaterals are squares. We refer to the non-square elements as transition elements. For more details about generating such a base mesh, we refer the reader to the previous work of Ebeida and Davis (2008).

In our implementation we used the gradient of the vorticity as the error-related function. Note that during the first time step of the dynamic remeshing algorithm, $\mathcal{M}_b$ and $\mathcal{M}_s$ are the same. Hence after we obtain a refined mesh corresponding to that time step, we repeat the simulation using the refined mesh. 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 mesh. An example for the output of the mesh adaptation algorithm is illustrated in Figure 7, using one refinement level.

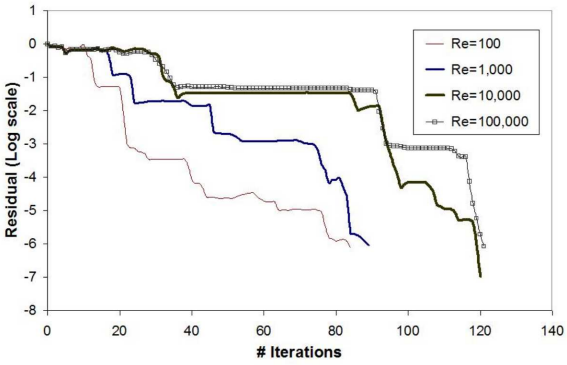
We remark that a transition element is formed by an incomplete refinement of a square element. This incomplete refinement is restricted here to either one or two neighboring edges of the square element. In the first case, the square is transformed into a block of two quads and one triangle. In the later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

Figure 8(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 Figure 8(b). The technique used to fix this

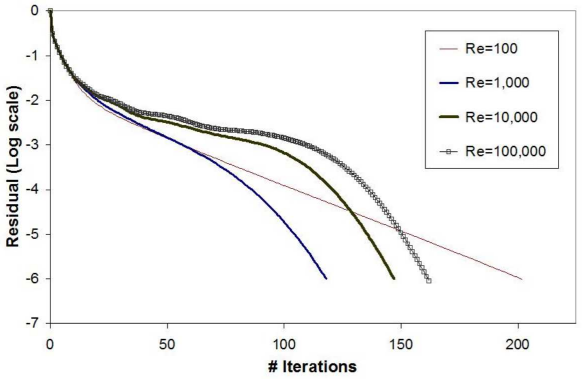


(a) Multigrid

(b) GMRES

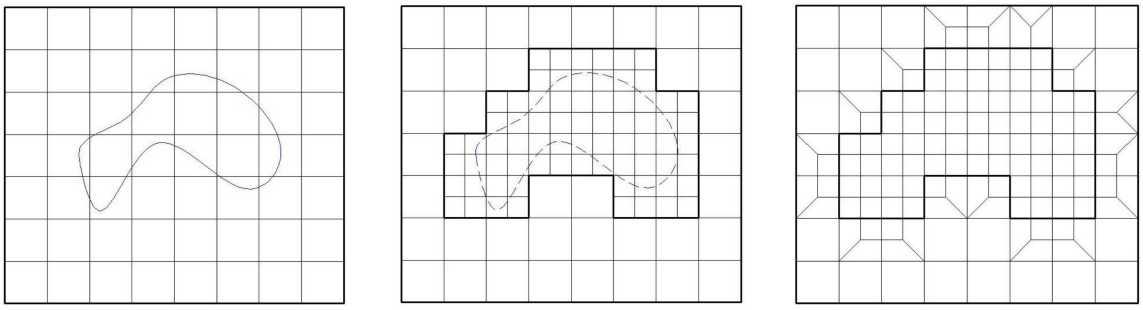


(c) TFQMR



(d) SLOR

Figure 6. Convergence history for different methods applied to the discrete convection-diffusion using 8673 nodes.



(a) Uniform mesh and region to be refined

(b) Refinement results in hanging nodes

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

Figure 7. One refinement level applied to a subregion of the uniform base mesh.

problem is to merge all the transition elements for that level. In other words, each block of transition elements is transformed into four square elements, as shown in Figure 8(c).

This algorithm is 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

Algorithm 1 (Adaptive Dynamic Remeshing)

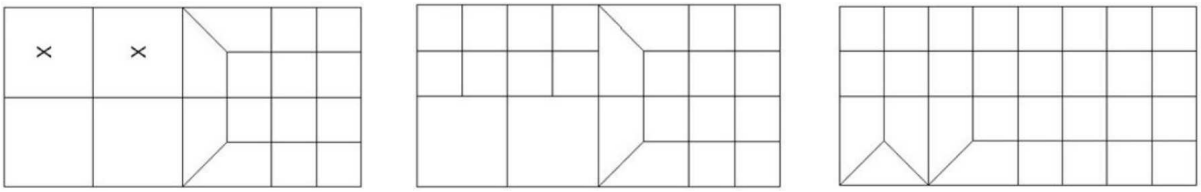
Input: Coarse base mesh $\mathcal{M}_b$, refined solution mesh $\mathcal{M}_s$, discrete error-related function f defined on $\mathcal{M}_s$, $r_L \in \mathrm{R}^{n_R}$:=user-specified thresholds for n_R refinement levels.

Create a copy of $\mathcal{M}_b$ and denote it $\mathcal{M}_o$.

for $i = 1$ to n_R **do**
  Project f from $\mathcal{M}_s$ to $\mathcal{M}_{i-1}$.
  Merge all the transition elements $\mathcal{M}_{i-1}$ of between refinement levels, $i-1$ and i.
  Refine any element in $\mathcal{M}_{i-1}$ at the refinement level $i-1$ if the maximum value of f over that element exceeds the threshold $r_L(i)$.
  Create new transition elements between the refinement levels i and $i-1$ to obtain a refined conformal mesh $\mathcal{M}_i$.
end for

Map the solution variables using a conservative interpolation scheme from $\mathcal{M}_s$ to $\mathcal{M}_{n_R}$.

Output: New refined mesh, $\mathcal{M}_{n_R}$, with a more convenient node distribution for simulating the next time step.



(a) Two coarse elements to be refined

(b) Hanging nodes might be created

(c) Merging of transition elements

Figure 8. Merging of transition elements during refinement of Coarse level elements to avoid the creation of hanging nodes.

within the flow. Currently, we do not adapt the mesh at every time step, but rather we chose to run this algorithm whenever the number of the mesh elements that should be refined exceeds 4% of the total number of elements.

5 Time-step control

In this section, we discuss an algorithm for the adaption of the time step, following the technique described in Stoer and Bulirsch (2002). Once the Reynolds number is specified, the grid is generated, and the test functions are chosen, the solution of equations (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:

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

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

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

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 9 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.

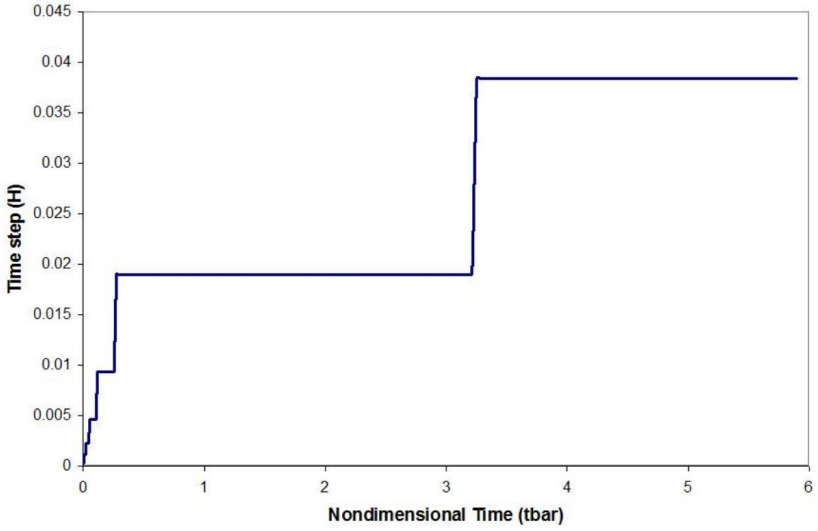


Figure 9. Evolution of time steps at the beginning of the simulation of flow over a cylinder at $Re = 1200$ with impulsive initial condition.

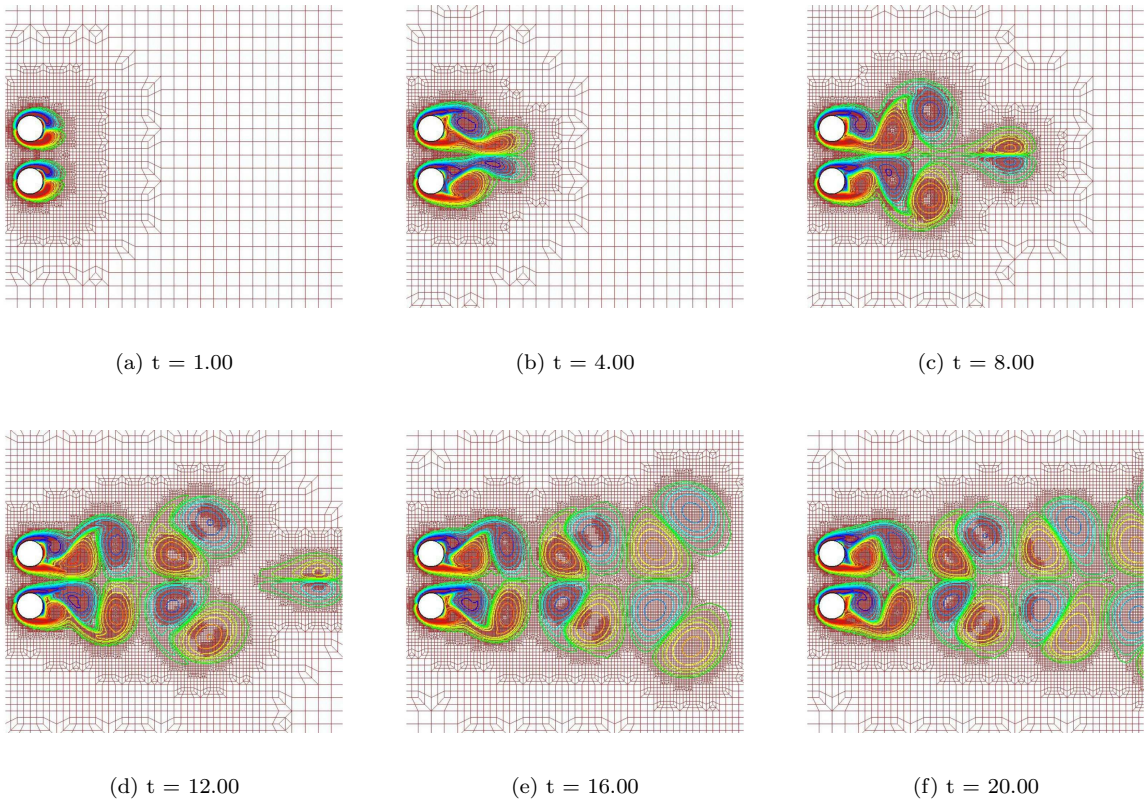
6 Numerical results

In this section, we present numerical results for two test problems to validate our proposed methodology.

6.1 *Cross-flow over two parallel cylinders at* $Re = 200$

The goal of this simulation is to illustrate the ability of our approach to simulate flow over multiple objects and to accurately resolve the interaction of the vortices generated around each cylinder. Figure 10 shows the interaction of the vortex shedding around each cylinder and how the vortices impact each other at the centerline while preserving the symmetry of the flow. Vorticity contours overlaid onto the computational mesh are shown in Figure 10 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 mesh in those regions. The contours and mesh 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 mesh refined in those regions. It should also be noted that the solution and the mesh remained symmetric about the centerline between the two cylinders indicating high spatial and temporal accuracy. The computed time history of the total lift and the total drag coefficients are presented in Figure 11. The oscillatory pattern corresponds to the shedding of the vortices and shows an important feature of this flow (Farrant *et al.* (2000)): the vortex shedding starts initially in symmetrical anti-phase mode and then after some time, it shifts to be in phase.



(a) t = 1.00 (b) t = 4.00 (c) t = 8.00

(d) t = 12.00 (e) t = 16.00 (f) t = 20.00

Figure 10. Evolution of mesh and vorticity contours for flow over two vertical cylinders at $Re = 200$.

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

Figure 12 shows the evolution of the vorticity and the mesh produced during a simulation of the unsteady incompressible flow around a NACA 0012 airfoil at $Re = 800$ and an angle of attack of $\alpha = 20°$. The domain of this problem is 11 units in length and 20 units in height. These are non-dimensional units relative to the chord length of the airfoil. The trailing edge is located at the origin. The lower left corner of the domain is located at $(-5, -10)$. For this case we set $Re = 800$, which forces the flow to be unsteady. The initial condition for the simulation was an impulsive flow at $t = 0$. As Figure 12 shows, an initial vortex is generated near the trailing edge of the airfoil followed by a sequence of vortices. The vortex shedding is the reason for the fluctuations in the generated lift and drag. The initial vortex reaches the boundaries of the domain at non-dimensional time $tU/L = 12.0$, while the transition phase decays at $tU/L = 30.0$. Here, L and U denote the characteristic length and velocity, respectively, for this flow problem. We picked this case to demonstrate the capability of the dynamic remeshing technique to capture the starting vortex and the steady vortex shedding. The computed time history of the total lift and the total drag coefficients are presented in Figure 13. Note that the oscillatory pattern is

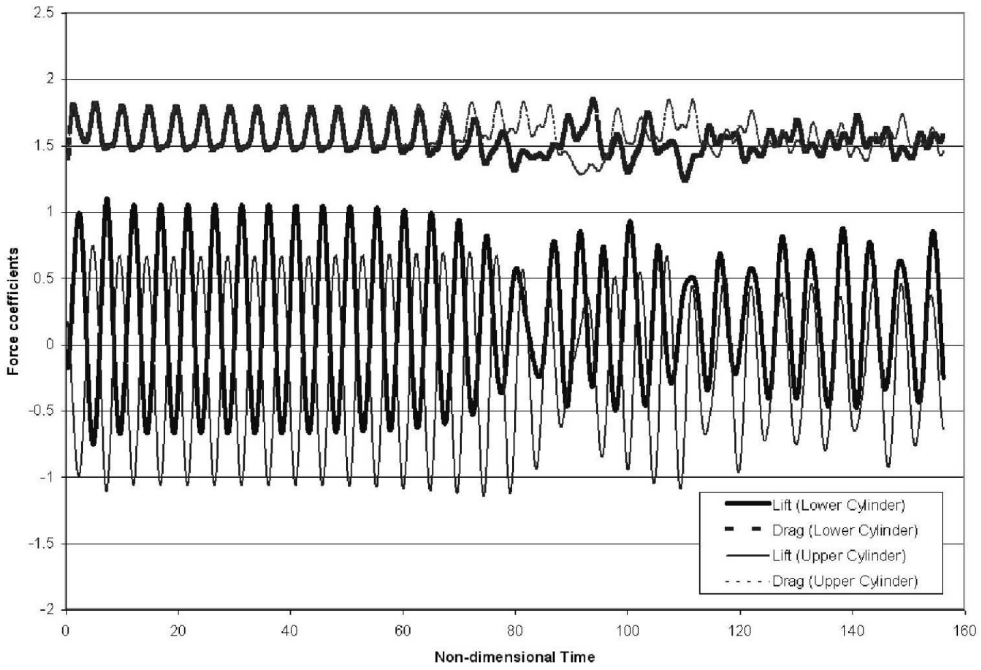


Figure 11. Evolution of the force coefficient with time for the flow over two vertical cylinders at $Re = 200$.

due to the shedding of the vortices. The computed corresponding Strouhal number, based on the foil chord, is approximately 0.54.

7 Concluding remarks

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 mesh 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 nodes in the mesh. 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, which we plan to do in future work. Also, we will investigate the extension of the approach in this paper to three-dimensional flows.

References

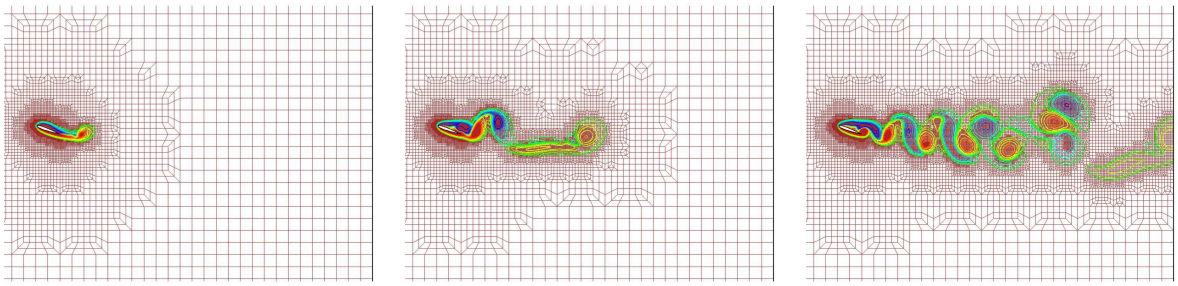
Adjerid, S., Aiffa, M. and Flaherty, J.E., 1995. High-order finite element methods for singularly perturbed elliptic and parabolic problems. *SIAM J. Appl. Math.*, 55 (2), 520–543.

Ainsworth, M., 2004. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *J. Comput. Phys.*, 198 (1), 106–130.

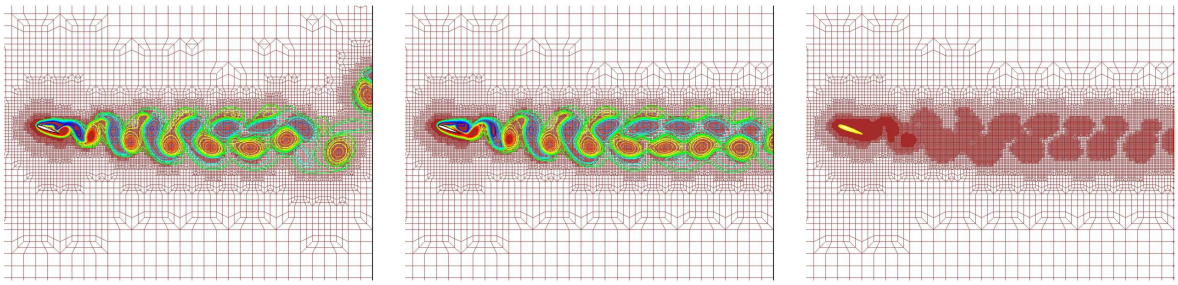
Babuška, I., Zienkiewicz, O.C., Gago, J. and de A. Oliveira, E.R., eds., 1986. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations.* Chichester, UK: Wiley.

Ebeida, M.S. and Davis, R., 2008. Fast adaptive hybrid mesh generation based on quad-tree decomposition. *In*: *Proceedings of the 38th Fluid Dynamics Conference and Exhibit.* AIAA Paper 2008–4141.

Ebeida, M.S., 2008. Simulation of unsteady incompressible turbulent flows using Galerkin finite element and adaptive grids. Thesis (PhD). University of California, Davis, CA.



(a) t = 1.20 (b) t = 5.80 (c) t = 13.80



(d) t = 19.60 (e) t = 27.80 (f) t = 27.80 - Mesh Only

Figure 12. Evolution of mesh and vorticity contours for the flow over a NACA 0012 airfoil at $Re = 800$, $\alpha = 20°$.

Eisenstat, S.C., 1981. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Statist. Comput.*, 2 (1), 1–4.

Elman, H., Silvester, D. and Wathen, A., 2005. *Finite elements and fast iterative solvers*. Oxford, UK: Oxford University Press.

Elman, H., Howle, V.E., Shadid, J., Silvester, D. and Tuminaro, R., 2007. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. *SIAM J. Sci. Comput.*, 30 (1), 290–311.

Eriksson, K., Johnson, C. and Larsson, S., 1998. Adaptive finite element methods for parabolic problems VI: Analytic semigroups. *SIAM J. Numer. Anal.*, 35 (4), 1315–1325.

Farrant, T., Tan, M. and Price, W.G., 2000. A cell boundary element method applied to laminar vortex-shedding from arrays of cylinders in various arrangements. *J. Fluids Struct.*, 14 (3), 375–402.

Fidkowski, K.J., Oliver, T.A., Lu, J. and Darmofal, D.L., 2005. *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 207 (1), 92–113.

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

Freund, R.W., 1993. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14 (2), 470–482.

Gresho, P.M. and Sani, R.L., 1998a. *Incompressible Flow and the Finite Element Method. Volume 1: Advection-Diffusion*. Chichester, UK: Wiley.

Gresho, P.M. and Sani, R.L., 1998b. *Incompressible Flow and the Finite Element Method. Volume 2: Isothermal Laminar Flow*. Chichester, UK: Wiley.

Gunzburger, M.D. and Nicolaides, R.A., eds. , 1993. *Incompressible Computational Fluid Dynamics: Trends and Advances*. Cambridge, UK: Cambridge University Press.

Hafez, M.M., ed. , 2003. *Numerical Simulations of Incompressible Flows*. Singapore: World Scientific Publishing.

Hétu, J.F. and Pelletiert, D.H., 1992. Fast, adaptive finite element scheme for viscous incompressible flows. *AIAA J.*, 30 (11), 2677–2682.

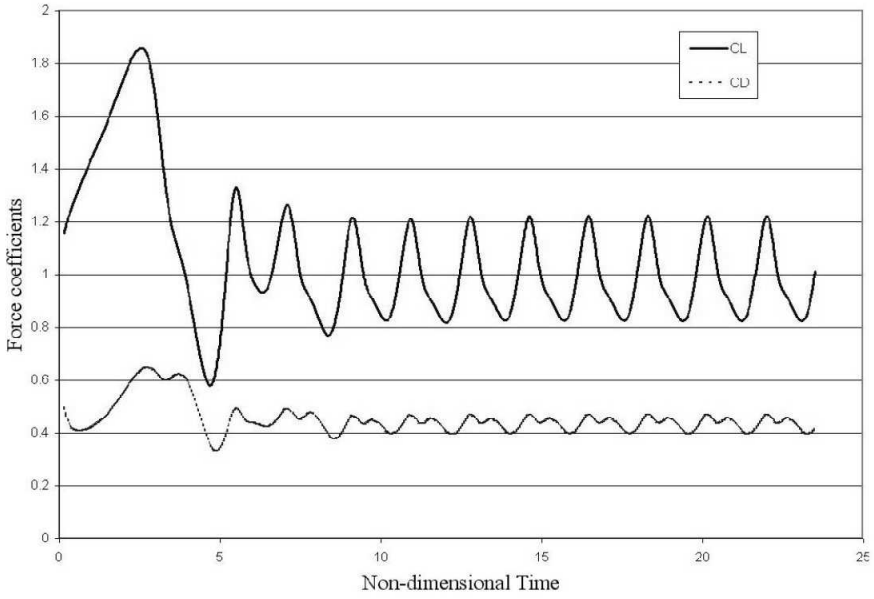


Figure 13. Evolution of the force coefficient with time for the flow over a NACA 0012 airfoil at $Re = 800$, $\alpha = 20°$.

Nagata, H., Funada, H. and Matsui, T., 1985. Unsteady flows in the vortex region behind a circular cylinder started impulsively. *Bull. JSME*, 28, 2608–2616.

Nejat, A. and Ollivier-Gooch, C., 2008. Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton-GMRES solver for the Euler equations. *J. Comput. Phys.*, 227 (4), 2366–2386.

Rannacher, R., 1999. Adaptive Galerkin finite element methods for partial differential equations. *J. Comput. Appl. Math.*, 128, 205–233.

Saad, Y., 2003. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia, PA: SIAM.

Saad, Y. and Schultz, M.H., 1986. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7 (3), 856–869.

Stoer, J. and Bulirsch, R., 2002. *Introduction to Numerical Analysis*. 3rd ed. New York, Berlin, Heidelberg: Springer-Verlag.