INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
*Int. J. Numer. Meth. Fluids* 2007; **55**:177–203
Published online 28 February 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1454

# Development of a hybrid finite volume/element solver for incompressible flows

Shuangzhang Tu[§] and Shahrouz Aliabadi[*, †, ‡]

*Northrop Grumman Center for HPC of Ship Systems Engineering, Jackson State University, MS e-Center Box 1400, 1230 Raymond Road, Jackson, MS 39204, U.S.A.*

## SUMMARY

In this paper, we report our development of an implicit hybrid flow solver for the incompressible Navier–Stokes equations. The methodology is based on the pressure correction or projection method. A fractional step approach is used to obtain an intermediate velocity field by solving the original momentum equations with the matrix-free implicit cell-centred finite volume method. The Poisson equation derived from the fractional step approach is solved by the node-based Galerkin finite element method for an auxiliary variable. The auxiliary variable is closely related to the real pressure and is used to update the velocity field and the pressure field. We store the velocity components at cell centres and the auxiliary variable at cell vertices, making the current solver a staggered-mesh scheme. Numerical examples demonstrate the performance of the resulting hybrid scheme, such as the correct temporal convergence rates for both velocity and pressure, absence of unphysical pressure boundary layer, good convergence in steady-state simulations and capability in predicting accurate drag, lift and Strouhal number in the flow around a circular cylinder. Copyright © 2007 John Wiley & Sons, Ltd.

Received 8 September 2006; Revised 24 November 2006; Accepted 29 December 2006

KEY WORDS: incompressible NSE; projection method; hybrid finite volume/element method; staggered mesh; Jacobian-free GMRES; matrix-free LU-SGS

[*]Correspondence to: Shahrouz Aliabadi, Northrop Grumman Center for HPC of Ship Systems Engineering, Jackson State University, MS e-Center Box 1400, 1230 Raymond Road, Jackson, MS 39204, U.S.A.
[†]E-mail: saliabadi@jsums.edu
[‡]Professor and Director.
[§]Assistant Professor.

Contract/grant sponsor: Army High Performance Computing Research Center; contract/grant numbers: DAAD19-01-2-0014, DAAD19-03-D-0001
Contract/grant sponsor: Northrop Grumman Ship Systems

## 1. INTRODUCTION

This paper describes the development of a numerical method to solve the incompressible Navier–Stokes equations (NSE), which can be expressed as

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

$$\rho \left[ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right] = \rho \mathbf{g} + \nabla \cdot \boldsymbol{\sigma} \tag{2}$$

with

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}})$$

on domain $(\mathbf{x}, t) \in \Omega \times (0, T)$. Here $\Omega$ is the spatial domain with boundary $\partial\Omega$. The time domain is represented by $(0, T)$. In Equations (1) and (2), $\rho$, $\mathbf{u}$, $p$, $\mathbf{g}$ and $\mu$ are the fluid density, velocity, pressure, gravitational force and dynamic viscosity, respectively. Equations (1) and (2) are completed by an appropriate set of boundary and initial conditions. Based on the velocity, the entire boundary can be generally categorized into $\partial\Omega = \Gamma_D \cap \Gamma_n \cap \Gamma_N$ (the notation follows that in [1]) where

$\Gamma_D$: $\mathbf{u} = \mathbf{u}_b$, i.e. all components of velocity are specified, e.g. inflow boundary .
$\Gamma_n$: $\mathbf{u} \cdot \mathbf{n} = (\mathbf{u} \cdot \mathbf{n})_b$, i.e. only the velocity component normal to the boundary is specified, e.g. symmetry boundary, and
$\Gamma_N$: $\boldsymbol{\sigma} \cdot \mathbf{n} = (\boldsymbol{\sigma} \cdot \mathbf{n})_b$, e.g. outflow boundary.

In Equations (1) and (2), the unknowns are the fluid velocity and the pressure. Numerous numerical methods for solving Equations (1) and (2) have been proposed and studied over the last four decades. Generally, the methods can be grouped into two broad categories, *coupled methods* and *segregated methods*.

In the *coupled approach*, the momentum equations and the continuity equations are solved simultaneously. The artificial compressibility (AC) method pioneered by Chorin [2] is such a coupled method. In this method, an artificial pressure time derivative is added to the continuity equation, eliminating the singularity when the original continuity equation is discretized and the artificial term will vanish upon convergence in the steady-state simulations. If the dual-time-stepping technique is used, the AC method is able to solve unsteady problems. Another method that was designed for low-Mach number flows can also be used to solve incompressible flows. This method preconditions the original compressible NSE to modify the eigenvalues of the hyperbolic equation system [3]. Note that this method does not start from the incompressible NSE. In the finite element (FE) community, many researchers use stabilized finite element methods (FEMs) to solve the incompressible NSE simultaneously. These stabilization techniques including two main approaches, the streamline upwinded/Petrov Galerkin (SUPG) [4] combined with the pressure stabilized Petrov Galerkin (PSPG) [5, 6], and the Galerkin least square (GLS) [7] method, play a crucial role in the success of this type of methods.

Among the category of the *segregated methods*, the pressure correction method is the most widely used for solving the incompressible NSE. The pressure correction method uses a fractional step approach to first obtain an intermediate velocity field which is generally not divergence-free. A Poisson equation is usually solved for the pressure correction. The final velocity is corrected to satisfy the discrete divergence-free constraint. The pressure correction method is also viewed as one of the so-called projection methods. Guermond *et al*. provide a comprehensive overview of the projection methods in [8] where other projection methods such as the velocity correction method and consistent splitting method are also reviewed (cf. [8] and references therein). Other variants of the projection method not covered in [8] includes the SIMPLE family methods pioneered by Caretto *et al*. [9] and the PISO method proposed by Issa [10].

A successful numerical solver for incompressible NSE must address the following issues:

- *Accuracy*: A numerical scheme should be at least second-order accurate in both space and time. In addition, the claimed spatial and temporal accuracy must be numerically verified. This can be done by conducting convergence rate tests.
- *Efficiency*: Implicit schemes are preferred since larger timesteps can be used without loss of accuracy and stability. Since implicit schemes result in large, sparse and usually ill-conditioned linear systems, robust and efficient preconditioners should be employed to accelerate the convergence of any linear system solvers.
- *Applicability*: Most engineering applications involve complex domains. Unstructured meshes are indispensable to discretize complex domains. Therefore, a numerical solver must be capable of handling unstructured meshes. In addition, a 2D solver must be able to be easily extended for 3D applications. Due to this reason, the vorticity-stream function formulation is not considered as practical because of its 2D limitation.

Over the years, the authors of this paper have developed a set of incompressible flow solvers using SUPG/PSPG stabilized FEM for incompressible free-surface flows [11] and contaminant dispersion flows [12]. Though we have achieved tremendous success in solving real engineering problems, we realized that the solver can be improved. The main drawback with the stabilized FE solver is the difficulty with the determination of the stabilization parameters. The stabilization parameters in the SUPG/PSPG stabilized FE solver rely heavily on the definition of the characteristic element length. For isotropic unstructured meshes, the characteristic length is well defined. However, for high Reynolds number flows where the high aspect ratio elements are usually used inside the boundary layer, the element length is not well defined. Inappropriate amount of stabilization, too much or too little, causes the loss of accuracy inside the boundary layer. The finite volume (FV)-based solver *CaMEL_Aero* [13] we developed for compressible flows, however, is very insensitive to the aspect ratio of elements and has been shown to be able to predict accurate aerodynamic forces for high-Reynolds number flows. Motivated by this realization, we would like to combine the finite volume method (FVM) and the FEM to develop a new hybrid solver for incompressible flows. The new solver is expected to take advantage of both methods and avoid the shortcomings.

In this paper, we do not aim to propose new formulations. The formulation we are using is actually the pressure-correction (projection) method similar to that of Timmermans *et al.* [14]. The aim of this paper is to present our implementation of the projection method with a hybrid numerical discretization for these formulations. We use the cell-centred FVM to solve the momentum equation for the intermediate velocity and the node-based Galerkin FEM to solve the Poisson equation for the pressure correction. Note that in our implementation, the pressure does not directly enter the momentum equation. Instead, an auxiliary variable closely related to the pressure takes the place of pressure in the momentum equations. Because the velocity components and the auxiliary variable are placed at different locations on the mesh, the current hybrid scheme can be viewed as a staggered-mesh scheme. However, our staggered mesh deployment is distinct from the more traditional staggered mesh scheme where the pressure is put on the cell centre and the velocity is put on the cell interfaces. In our deployment, the velocity unknowns are put at the cell centres and the auxiliary variable is put at the mesh vertices. This deployment makes it convenient to compute the gradients of the auxiliary variable using local FE basis function, which is required for solving the momentum equations. Numerical experiments show that the current staggered deployment also avoids the notorious velocity–pressure odd–even decoupling phenomenon. Another key feature of the current hybrid scheme is that it allows the same order of spatial discretization for velocity and pressure. In other words, the velocity takes a second-order accurate reconstruction in the FV framework and the pressure uses linear basis functions in the FE framework. The pressure is updated carefully to ensure it is free of any unphysical boundary layer. Numerical results will show that the current hybrid implementation is superconvergent in terms of the spatial convergence rates for both velocity and pressure. The correct temporal convergence rates can also be obtained for both velocity and pressure.

The rest of this paper is organized as follows. In Section 2, we describe the solution procedure in detail. Section 3 provides key components of our FV solver for the momentum equations. Following that is Section 4 where we present a brief description of the FE-based Poisson solver. In Section 5, we provide more information on how to compute the velocity gradients inside each cell and the solution and its derivatives at vertices. Three test cases are presented in Section 6 to demonstrate the performance of the current solver. Finally in Section 7, we summarize this paper with final concluding remarks.

## 2. SOLUTION PROCEDURE

In this section, we describe the solution procedure in detail. First, we explain the storage locations of the unknowns. Illustrated in Figure 1, cell centres store the velocity components and pressure and mesh nodes store an auxiliary variable $q$ that will be explained later. The auxiliary variable is closely related to the pressure and stored at a different location from the velocity components. For this reason, we consider our implementation a staggered-grid scheme. However, it must be pointed out that the current scheme is distinct from other conventional staggered grid schemes where velocity components are usually stored at cell edges and the pressure at cell centres.

Briefly, the solution procedure follows a segregated approach to decouple the pressure from the velocity. The velocity field is updated by solving the momentum equation provided that a known pressure field is given as a source term, through a cell-centred FV discretization that will be described in next section. The pressure does not directly enter the momentum equation. Instead,

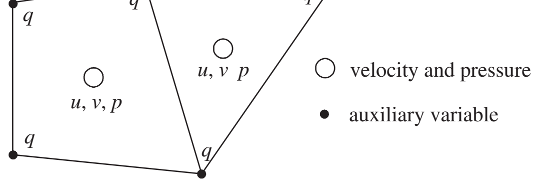


Figure 1. Storage locations for unknowns on an unstructured mesh.

an auxiliary variable, which is closely related to the pressure, takes the place of pressure in the momentum equation, providing pressure gradient information. Because only pressure gradients instead of the pressure itself are needed in the momentum equation, we put the auxiliary variable on the vertices of cells. This deployment provides a convenient way to evaluate the pressure gradient using the local FE basis functions without the need of reconstruction as required for the velocity field. The incremental value of the auxiliary variable is computed by solving a Poisson equation to update the velocity field. After the final velocity field is determined, the pressure can be updated according to the auxiliary variable and the velocity divergence field. The rest of this section explains the solution procedure in more detail.

## *2.1. Updating the velocity field*

To facilitate our analysis, we rewrite the momentum equation (2) as

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \nabla p = \rho(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u}) + \mu\nabla^2\mathbf{u} \tag{3}$$

where we utilize the fact that the divergence of the viscous stress tensor is $\mu(\nabla^2\mathbf{u} + \nabla(\nabla\cdot\mathbf{u}))$, which is just $\mu\nabla^2\mathbf{u}$ when $\nabla\cdot\mathbf{u} = 0$.

Because the pressure field is not available at the same time level as the velocity when solving the momentum equation, we use the fractional step method to first compute an intermediate velocity $\tilde{\mathbf{u}}$

$$\rho\frac{\alpha_1\tilde{\mathbf{u}} + \alpha_0\mathbf{u}^n + \alpha_{-1}\mathbf{u}^{n-1}}{\Delta t} + \nabla q^n = \rho(\mathbf{g} - \tilde{\mathbf{u}}\cdot\nabla\tilde{\mathbf{u}}) + \mu\nabla^2\tilde{\mathbf{u}} \tag{4}$$

where the time-dependent term of Equation (3) has been discretized using the backward difference formula (BDF). For first-order time accurate scheme (BDF1), $\alpha_1 = 1.0$, $\alpha_0 = -1.0$ and $\alpha_{-1} = 0.0$ and for second-order time accurate scheme (BDF2), $\alpha_1 = 1.5$, $\alpha_0 = -2.0$ and $\alpha_{-1} = 0.5$. In Equation (4), an obvious choice for $q$ is $p$, but for the time being, $q$ can be considered as an auxiliary variable that is closely related to the real pressure $p$. A correction step is used to update the velocity field to obtain the final velocity at the end of the timestep, $\mathbf{u}^{n+1}$, which can be expressed as

$$\rho\alpha_1\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} + \nabla q' = 0 \tag{5}$$

where $q'$ is the incremental value of $q$. $\mathbf{u}^{n+1}$ can therefore be updated according to Equation (5), i.e.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho\alpha_1}\nabla q' \tag{6}$$

We expect $\nabla\cdot\mathbf{u}^{n+1} = 0$ at the end of the timestep. Hence, Equation (6) happens to be the Hodge decomposition of the intermediate velocity field, $\tilde{\mathbf{u}}$, into the sum of a divergence-free vector field, $\mathbf{u}^{n+1}$, and a curl-free vector field, $\Delta t\nabla q'/(\rho\alpha_1)$. Due to this, the correction step is also referred to as a projection step. Before Equation (6) can be utilized, we must first obtain the updated $q'$ field.

Taking the divergence of Equation (6) and considering $\nabla\cdot\mathbf{u}^{n+1} = 0$, we obtain

$$\nabla^2 q' = \frac{\rho\alpha_1}{\Delta t}\nabla\cdot\tilde{\mathbf{u}} \tag{7}$$

which is a Poisson equation for $q'$. We want to enforce the normal component of $\tilde{\mathbf{u}}$ and $\mathbf{u}^{n+1}$ to be the correct boundary conditions, i.e.

$$\mathbf{n}\cdot\tilde{\mathbf{u}} = \mathbf{n}\cdot\mathbf{u}^{n+1} = (\mathbf{n}\cdot\mathbf{u})_b \tag{8}$$

where $(\mathbf{n}\cdot\mathbf{u})_b$ is the prescribed normal component of the velocity on the boundary $\Gamma_D$ and $\Gamma_n$. Besides, Equations (5) and (8) imply

$$\mathbf{n}\cdot\nabla q' = 0 \quad \text{on } \Gamma_D \text{ and } \Gamma_n \tag{9a}$$

In addition, we also specify

$$q' = [\mu\mathbf{n}\cdot(\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}})\cdot\mathbf{n}]_b - q^n \quad \text{on } \Gamma_N \tag{9b}$$

which comes from the stress-free assumption on $\Gamma_N$. Actually $q' = 0$ on $\Gamma_N$ is more commonly used in the literature.

Equation (7) together with the boundary conditions Equations (8) and (9) provide a well-posed Poisson equation that can be solved using any Poisson solver. A FE-based Poisson solver will be described in Section 4.

After $q'$ field is available, Equation (6) can be invoked to update the velocity field. The resulting velocity field will be divergence-free and considered as the final solution. Also the auxiliary variable will be updated as $q^{n+1} = q^n + q'$.

## 2.2. *Updating the pressure field*

The real pressure, $p^{n+1}$ can be seemingly updated according to

$$p^{n+1} = q^{n+1} \equiv q^n + q' \tag{10}$$

However, $q$ suffers from an unphysical homogeneous Neumann boundary condition Equation (9a) on $\Gamma_D$ and $\Gamma_n$. If $p$ is updated through Equation (10), $p$ will inherit this defect. This artificial pressure Neumann boundary condition causes the loss of accuracy in the pressure field. A direct effect is that the temporal convergence rate of pressure will not reach the intended value (e.g. second order for BDF2). To cure this, we need to find an alternative way to update the pressure. The new way must yield correct physical pressure Neumann boundary condition. Timmermans *et al.* [14] propose such a cure. In their approach the pressure is updated *via*

$$p^{n+1} = p^n + p' - \mu\nabla\cdot\tilde{\mathbf{u}} \tag{11}$$

Note that in Equation (11), $p$ instead of $q$ is used in the momentum equation, which is different from our current approach. Guermond and Shen [15] prove that Equation (11) yields about 1.5th order temporal accuracy in pressure even if the velocity is fully second-order accurate in time.

In our implementation, we use the auxiliary variable $q$ in the momentum equation. Therefore, our variant to Equation (11) becomes

$$p^{n+1} = q^n + q' - \mu\nabla\cdot\tilde{\mathbf{u}} \tag{12}$$

Recall that $p$ is stored at the cell centre and $q$ at the cell vertex (cf. Figure 1). In practice, we update $p$ according to Equation (47) using FE basis functions that will be given in Section 4. The convergence tests we will conduct in Section 6 indicate that Equation (12) yields fully second-order convergence in time for both velocity and pressure. Both Equations (11) and (12) lead to correct pressure Neumann boundary conditions on $\Gamma_D$ and $\Gamma_n$, which can be obtained by taking the dot product of the unit outward boundary normal vector with the original momentum equation (3), i.e.

$$\mathbf{n}\cdot\nabla p^{n+1}=\mathbf{n}\cdot\left(\rho\left(-\frac{\partial\mathbf{u}^{n+1}}{\partial t}+\mathbf{g}-\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1}\right)+\mu\nabla^2\mathbf{u}^{n+1}\right) \tag{13}$$

Note that on $\Gamma_D$ and $\Gamma_n$, either $\partial\mathbf{u}/\partial t=0$ (e.g. inflow) or $\mathbf{n}\cdot\mathbf{u}=0$ (e.g. symmetry plane), so the true pressure Neumann boundary condition can be simplified to

$$\mathbf{n}\cdot\nabla p^{n+1}=\mathbf{n}\cdot[\rho(\mathbf{g}-\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1})+\mu\nabla^2\mathbf{u}^{n+1}] \tag{14}$$

Keep in mind that Equation (14) is the physical pressure Neumann boundary condition. Now we proceed to prove Equation (12) leads to a pressure Neumann boundary condition that closely approximates Equation (14). From Equation (12), we can derive

$$\nabla q'=\nabla p^{n+1}-\nabla q^n+\nabla(\mu\nabla\cdot\tilde{\mathbf{u}}) \tag{15}$$

Substituting Equation (15) into Equation (5) and taking the sum of Equations (4) and (5) to obtain

$$\rho\frac{\alpha_1\mathbf{u}^{n+1}+\alpha_0\mathbf{u}^n+\alpha_{-1}\mathbf{u}^{n-1}}{\Delta t}+\nabla p^{n+1}=\rho(\mathbf{g}-\tilde{\mathbf{u}}\cdot\nabla\tilde{\mathbf{u}})+\mu\nabla^2\tilde{\mathbf{u}}-\nabla(\mu\nabla\cdot\tilde{\mathbf{u}}) \tag{16}$$

By invoking the identity $\nabla^2\tilde{\mathbf{u}}\equiv\nabla(\nabla\cdot\tilde{\mathbf{u}})-\nabla\times\nabla\times\tilde{\mathbf{u}}$ and also noticing from Equation (5) that $\nabla\times\nabla\times\tilde{\mathbf{u}}=\nabla\times\nabla\times\mathbf{u}^{n+1}$, Equation (16) can be rewritten as

$$\rho\frac{\alpha_1\mathbf{u}^{n+1}+\alpha_0\mathbf{u}^n+\alpha_{-1}\mathbf{u}^{n-1}}{\Delta t}+\nabla p^{n+1}=\rho(\mathbf{g}-\tilde{\mathbf{u}}\cdot\nabla\tilde{\mathbf{u}})-\mu\nabla\times\nabla\times\mathbf{u}^{n+1} \tag{17}$$

Taking the dot product of the unit outward boundary normal vector with Equation (17) and using the same reasoning to derive Equation (14) yields

$$\mathbf{n}\cdot\nabla p^{n+1}=\mathbf{n}\cdot[\rho(\mathbf{g}-\tilde{\mathbf{u}}\cdot\nabla\tilde{\mathbf{u}})-\mu\nabla\times\nabla\times\mathbf{u}^{n+1}] \tag{18}$$

On the other hand, by invoking the identity $\nabla^2\mathbf{u}^{n+1}\equiv\nabla(\nabla\cdot\mathbf{u}^{n+1})-\nabla\times\nabla\times\mathbf{u}^{n+1}$ and $\nabla\cdot\mathbf{u}^{n+1}=0$, Equation (14) can be rewritten as

$$\mathbf{n}\cdot\nabla p^{n+1}=\mathbf{n}\cdot[\rho(\mathbf{g}-\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1})-\mu\nabla\times\nabla\times\mathbf{u}^{n+1}] \tag{19}$$

which is the true physical Neumann boundary condition for pressure. Because Equations (18) and (19) utilize the operator $\nabla\times\nabla\times$, they are referred to as the *rotational form* in [15].

As can be seen, the only difference between Equations (18) and (19) is the advection term. For Stokes flows where advection is absent, Equations (18) and (19) are exactly the same. For more general flows, we observe the following for typical portions of the boundary $\Gamma_D$ and $\Gamma_n$.

- On solid walls where the no-slip boundary condition $\mathbf{u}^{n+1}=\tilde{\mathbf{u}}=0$ is usually applied, Equations (18) and (19) are the same.
- On symmetry boundaries that are aligned with the coordinate axis, $\mathbf{n}\cdot(\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1})=\mathbf{n}\cdot(\tilde{\mathbf{u}}\cdot\nabla\tilde{\mathbf{u}})=0$. So Equations (18) and (19) are still the same.

- On inflow boundaries, Equations (18) and (19) are not the same, but Equation (18) is a reasonably good approximation to Equation (19) because at inflow, the velocity gradients are usually small.

Note that on $\Gamma_N$ Dirichlet boundary condition (cf. Equation (9b)) instead of the Neumann boundary condition is used for pressure.

Based on the observations above, we can conclude that the pressure updating strategy using Equation (12) truly yields correct pressure Neumann boundary conditions. The numerical test presented in Section 6 will verify this conclusion.

For clarity, let us summarize the entire solution procedure as follows.

Given an initial velocity field $\mathbf{u}$ (not necessarily divergence-free) and pressure field $q$, do

*Step 1*: Solve Equation (4) to obtain the intermediate velocity field $\tilde{\mathbf{u}}$.

*Step 2*: Solve Equation (7) to obtain the incremental auxiliary variable $q'$.

*Step 3*: Update the pressure $p^{n+1}$ through Equation (12).

*Step 4*: Update the velocity $\mathbf{u}^{n+1}$ through Equation (6).

Go to Step 1 to start the next timestep.

As can be seen from the above, the solution procedure is very efficient. Inside each physical timestep, only an advection–diffusion-typed momentum equation (Step 1) and a Poisson equation (Step 2) are solved. Step 3 and Step 4 are done with trivial cost.

## 3. MATRIX-FREE FINITE VOLUME SOLVER FOR MOMENTUM EQUATIONS

The momentum equation (4) is a fully implicit non-linear advection–diffusion equation system with pressure gradients and gravity as source terms. We use the cell-centred FV method to discretize this system and solve the discretized system with the Newton–Raphson iterative method. This corresponds to Step 1 in the solution procedure described in the previous section. In this paper, we follow exactly the same procedure as in our FV compressible flow solver, *CaMEL_Aero* [13]. The rest of this section reviews some key ingredients of the FV solver.

### 3.1. Finite volume discretization

After the computational domain is discretized into conforming (without hanging nodes) cells (hybrid triangular/quadrilateral cells are allowed), the FV formulation can be written for each control volume. Since the current FV solver is cell-centred, i.e. the control volume is the cell itself, the velocity unknowns are stored at the cell centroids (cf. Figure 1). Following the standard FV discretization, we can obtain the semi-discrete form for cell $i$

$$\left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}\right)_i + \mathbf{R}(\mathbf{u}) = 0$$

where $\mathbf{R}(\mathbf{u}) = \frac{1}{|\Omega_i|}\int_{\partial\Omega_i} \mathbf{H}\cdot\mathbf{n}\,\mathrm{d}\Gamma$. Here $\mathbf{H}$ includes both inviscid and viscous fluxes, $\mathbf{n}$ is the outward unit normal vector of the faces surrounding cell $i$ and $|\Omega_i|$ is the volume of cell $i$. Assuming the control volume is composed of piecewise linear facets, we can use the one-point integration rule for the boundary integral. The boundary integral can be approximated with

$$\mathbf{R}(\mathbf{u}) = \frac{1}{|\Omega_i|}\sum_{k=1}^{n_\mathrm{f}} (\mathbf{H}\cdot\mathbf{n}\Delta s)_k \tag{20}$$

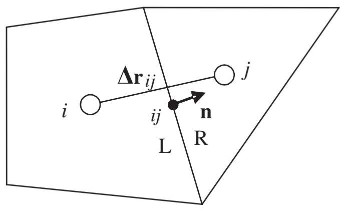


Figure 2. Stencil to compute the fluxes across the cell interface.

where $n_\mathrm{f}$ is the number of faces of the control volume and $\Delta s$ the area (length in 2D) of $k$th face of cell $i$. The following two sub-sections provide some details in computing the inviscid and viscous fluxes across the face shared by cell $i$ and cell $j$ (cf. Figure 2). Note that in Figure 2 the location of the interface is denoted by $ij$.

*3.1.1. Inviscid flux.* The inviscid flux takes the general form of the flux vector splitting scheme widely used as an approximate Riemann solver in compressible flow solvers.

$$\mathbf{F}_\mathrm{INV} = \lambda_{ij}^{+}\mathbf{u}_{ij}^{\mathrm{L}} + \lambda_{ij}^{-}\mathbf{u}_{ij}^{\mathrm{R}} \tag{21}$$

where the second-order accurate reconstruction is used to compute the left state and the right state of the solution at the interface

$$\begin{aligned}\mathbf{u}_{ij}^{\mathrm{L}} &= \mathbf{u}_i + (\mathbf{x}_{ij} - \mathbf{x}_i)\cdot\nabla\mathbf{u}_i\\ \mathbf{u}_{ij}^{\mathrm{R}} &= \mathbf{u}_j + (\mathbf{x}_{ij} - \mathbf{x}_j)\cdot\nabla\mathbf{u}_j\end{aligned} \tag{22}$$

and $\lambda_{ij}^{\pm} = (\lambda_{ij} \pm |\lambda_{ij}|)/2$ representing the positive and negative eigenvalues of the inviscid flux Jacobian matrix. A unique solution, $\mathbf{u}_{ij}$, at the cell interface must be obtained to evaluate $\lambda_{ij} = \mathbf{n}\cdot\mathbf{u}_{ij}$, which is the advection velocity normal to the interface. We use the following formula to compute $\mathbf{u}_{ij}$:

$$\mathbf{u}_{ij} = 0.5\mathbf{u}_m + (\mathbf{x}_{ij} - \mathbf{x}_m)\cdot\nabla\mathbf{u}_m \tag{23}$$

where '$m$' denotes the centre point between cell centroids '$i$' and '$j$'. In Equation (22), $\mathbf{u}_m = 0.5(\mathbf{u}_i + \mathbf{u}_j)$ and $\mathbf{x}_m = 0.5(\mathbf{x}_i + \mathbf{x}_j)$ obviously. However, there are two choices for $\nabla\mathbf{u}_m$. One is $\nabla\mathbf{u}_m = 0.5(\nabla\mathbf{u}_i + \nabla\mathbf{u}_j)$ which is simply the arithmetic average of the gradients at two adjacent cell centroids. The other is $\nabla\mathbf{u}_m = \nabla\mathbf{u}_{ij}$ which is the gradient at the interface and remains to be defined later. The former choice has been more conventionally used in the literature. However, we claim that the latter yields more accurate results for unevenly spaced grids. This can be confirmed by considering 1D unevenly spaced grids with zero gradients (e.g. spatially first-order scheme) in two adjacent cells. In this case, $\nabla\mathbf{u}_m = \nabla\mathbf{u}_{ij}$ that is generally non-zero provides some correction value to $0.5(\mathbf{u}_i + \mathbf{u}_j)$, which is more accurate if the grid is not evenly spaced.

*3.1.2. Viscous flux.* The viscous flux is computed according to

$$\mathbf{F}_\mathrm{VIS} = \mathbf{n}\cdot(\mu\nabla\mathbf{u}_{ij}) \tag{24a}$$

or

$$\mathbf{F}_{\mathrm{VIS}} = \mathbf{n}\cdot[\mu(\nabla\mathbf{u}_{ij} + \nabla\mathbf{u}_{ij}^{\mathrm{T}})] \tag{24b}$$

if the stress-form is used for viscous terms.

Therefore, the velocity gradient across the cell interface is required to compute the viscous flux. For unstructured meshes, there is no explicit line structure as in structured meshes. Hence, obtaining the gradient of a velocity component requires special attention. Mathur and Murphy [16] adopted a sophisticated formula to compute the gradient at the face *ij*. Denoting a component of $\mathbf{u}$ with $u$, we have

$$\nabla u_{ij} = \frac{u_j - u_i}{\Delta\mathbf{r}_{ij}\cdot\mathbf{n}}\mathbf{n} + \left(\overline{\nabla u_{ij}} - \frac{\overline{\nabla u_{ij}}\cdot\Delta\mathbf{r}_{ij}}{\Delta\mathbf{r}_{ij}\cdot\mathbf{n}}\mathbf{n}\right) \tag{25}$$

where $\overline{\nabla u_{ij}} = \frac{1}{2}(\nabla u_i + \nabla u_j)$, $\Delta\mathbf{r}_{ij}$ is the displacement vector connecting cell centroids $i$ and $j$, and $\mathbf{n}$ is the unit normal of the face *ij* (cf. Figure 2). The idea behind Equation (25) is that the gradient at the face is divided into two components. The first component is the gradient in the direction normal to the face and computed by the first term of Equation (25). The second component is computed by averaging the gradients at cell centroids and removing the component normal to the face.

Note that $\nabla\mathbf{u}_{ij}$ computed according to Equation (25) serves two purposes. One is to provide information for computing the viscous flux in Equation (24). The other is to provide the information required in Equation (23) to compute the unique solution at the cell interface.

## 3.2. Implicit time integration

We adopt the implicit backward Euler difference formula (BDF) for the time integration

$$\frac{\alpha_1\mathbf{u}^{n+1} + \alpha_0\mathbf{u}^n + \alpha_{-1}\mathbf{u}^{n-1}}{\Delta t} + \mathbf{R}(\mathbf{u}^{n+1}) = 0 \tag{26}$$

First-order (BDF1) or second-order (BDF2) scheme can be chosen depending on the values of the coefficients $\alpha_1$, $\alpha_0$ and $\alpha_{-1}$. We can use $\mathbf{G}(\mathbf{u})$ to denote the left-hand side of Equation (26). Therefore, Equation (26) can be expressed as

$$\mathbf{G}(\mathbf{u}) = 0 \tag{27}$$

which is a non-linear equation system. To solve Equation (27), we use the standard Newton–Raphson iterative method

$$\mathbf{J}\delta\mathbf{u} = -\mathbf{G}(\mathbf{u}) \tag{28}$$

where $\mathbf{J}$ is the Jacobian matrix and can be computed as

$$\mathbf{J} \equiv \frac{\partial\mathbf{G}}{\partial\mathbf{u}} = \frac{\alpha_1}{\Delta t}\mathbf{I} + \frac{\partial\mathbf{R}}{\partial\mathbf{u}} = \frac{\alpha_1}{\Delta t}\mathbf{I} + \tilde{\mathbf{J}} \tag{29}$$

where $\tilde{\mathbf{J}}$ denotes the contribution of $\mathbf{J}$ from the spatial flux terms and $\mathbf{I}$ is the identity matrix.

*Remarks*

- For transient simulations, BDF1 or BDF2 can be chosen. The user provides a proper value of timestep. During each physical timestep, several sub-iterations are performed to drive the unsteady residual (defined as the root-mean-square (RMS) of $\mathbf{G}(\mathbf{U})$ in Equation (27) over the whole computational domain) to reach the pre-specified tolerance (measured as the order of magnitude dropped). Three order drop of the magnitude of the residual during each timestep is usually sufficient for the accuracy.
- For steady-state simulations, BDF1 is always used. Inside each physical timestep, only one sub-iteration is performed. Sufficient number of timesteps are run to drive the steady residual (defined as the RMS of $\mathbf{R}(\mathbf{U})$ in Equation (20) over the whole computational domain) to reach the pre-specified tolerance.

### 3.3. *Jacobian-free GMRES solver*

Inside each non-linear Newton–Raphson iteration, a linear system described as Equation (28) must be solved. This linear system is usually huge, sparse and ill-conditioned. The generalized minimal residual method (GMRES) [17] has been widely used to solve this kind of linear system. Because the GMRES algorithm involves only matrix–vector multiplication, it is unnecessary to form the Jacobian matrix explicitly. We are able to approximate the matrix–vector product using [18]

$$\tilde{\mathbf{J}}\mathbf{v} \approx [\mathbf{R}(\mathbf{u}+\varepsilon\mathbf{v}) - \mathbf{R}(\mathbf{u})]/\varepsilon \tag{30}$$

where $\mathbf{R}$ is evaluated according to Equation (20). Equation (30) can be shown to be the first-order Taylor series expansion approximation to the multiplication of the Jacobian matrix, $\tilde{\mathbf{J}}$, and the Krylov vector, $\mathbf{v}$. Obviously, only the spatial contribution $\tilde{\mathbf{J}}$ in Equation (29) needs to be approximated in such a way because the time-dependent term can be evaluated exactly.

In Equation (30), the choice of $\varepsilon$ is a balance between the approximation accuracy and the floating point rounding error. It is as much of art as science. We use the following formula to obtain $\varepsilon$:

$$\varepsilon = \delta \frac{\|\mathbf{u}\|_2}{\|\mathbf{v}\|_2} \tag{31}$$

where $\delta$ is usually taken as the square root of the machine zero, which is about $10^{-8}$–$10^{-7}$ on most platforms.

This approximation has the following advantages: (i) avoid the difficulty and cost in forming the Jacobian matrix. For high-order FV solvers, the analytic evaluation of the Jacobian matrix is not readily available because of the big stencil involved; (ii) save a significant amount of memory for storing the Jacobian matrix. Even though the sparse Jacobian matrix could be stored in a compressed way [17], the storage saving is still significant. Of course, the disadvantage of not forming the Jacobian matrix is: we have to evaluate the function $\mathbf{R}$ a number of times depending on the size of the Krylov space.

### 3.4. *Matrix-free LU-SGS preconditioning*

The convergence performance of the GMRES algorithm is highly related to the preconditioning. In *CaMEL_Aero* [13], we adopt the lower-upper symmetric Gauss–Seidel (LU-SGS) method [19] as a preconditioning technique. As mentioned in the previous sub-section, it is not easy to form the Jacobian matrix analytically in high-order FV solvers. In contrast, the Jacobian matrix of the low-order (assuming the solution is constant inside the control volume, i.e. first-order accurate) flux function can be trivially obtained and is more diagonally dominant and more compact than the high-order Jacobian matrix. Therefore, the low-order Jacobian matrix is a good candidate as preconditioner to the high-order Jacobian matrix. In this paper, we directly follow what we did in our compressible solve, *CaMEL_Aero* [13], with slight changes. In *CaMEL_Aero*, we use the simplest and most dissipative flux function, the local Lax–Friedrich (LF) flux to establish the preconditioning matrix. The local LF flux normal to the cell interface can be expressed as

$$\mathrm{Flux}_{\mathrm{LF}} = \mathbf{T}^{-1}\{\tfrac{1}{2}[\mathbf{F}(\mathbf{Tu}_i) + \mathbf{F}(\mathbf{Tu}_j) - \lambda^*(\mathbf{Tu}_j - \mathbf{Tu}_i)]\} \tag{32}$$

where $i$ and $j$ are the indices of the left and right adjacent cells of the face, respectively, $\mathbf{T}$ is the orthonormal rotation matrix of the face, and $\lambda^*$ is the velocity normal to the interface. $\lambda^*$ is determined from the arithmetic average of the left and the right states $|\mathbf{Tu}_i + \mathbf{Tu}_j|/2$. The difference between the current implementation and in *CaMEL_Aero* is that the speed of sound does not enter the evaluation of $\lambda^*$ in the current incompressible flow environment.

For the flux function described as Equation (32), the Jacobian matrix can be easily computed. And, with the face-based data structure, the resulting Jacobian matrix can be conveniently separated into block diagonal part, lower block triangular part and upper block triangular part

$$\mathbf{J}_{\mathrm{low}} = \mathbf{D} + \mathbf{L} + \mathbf{U} \tag{33}$$

where the subscript low indicates that the Jacobian matrix comes from the low-order dissipative flux function. Assuming that $j<i$ in Equation (32), we obtain the $\mathbf{L}$ operator for cell $i$ contributed by cell $j$

$$\mathbf{L}_{ij} = \frac{1}{2|\Omega_i|}\mathbf{T}^{-1}\left[\frac{\partial \mathbf{F}(\mathbf{Tu}_j)}{\partial(\mathbf{Tu}_j)} - \lambda^*\mathbf{I}\right]\Delta s\mathbf{T} \tag{34}$$

where $\Delta s$ is the area of the face shared by cells $i$ and $j$. If $j>i$ in Equation (32), then the $\mathbf{U}$ operator is obtained. The diagonal block for row $i$ of $\mathbf{J}_{\mathrm{low}}$ can be expressed as

$$\mathbf{D}_i = \left(\frac{\alpha_1}{\Delta t} + \frac{1}{2|\Omega_i|}\sum_{k=1}^{n_{\mathrm{f}}} \lambda_k^* \Delta s_k\right)\mathbf{I} \tag{35}$$

which can be represented by a single scalar for each cell. To derive the above equation, we have used the fact that $\sum_{k=1}^{n_{\mathrm{f}}} \mathbf{n}_k \Delta s_k = \mathbf{0}$ for closed polygons. Also, the time-dependent term is included in $\mathbf{D}$. Note that the viscous contributions to $\mathbf{L}_{ij}$ and $\mathbf{D}_i$ can be included in $\lambda^*$ in the form of $\mu/(\rho\|\Delta\mathbf{r}_{ij}\cdot\mathbf{n}\|)$ (cf. Equation (25) and assuming $\overline{\nabla u_{ij}} = \mathrm{const.}$ when evaluating the Jacobian.).

The preconditioning matrix is taken as the approximate LU-SGS factorization of $\mathbf{J}_{\mathrm{low}}$, namely

$$\mathbf{P} = (\mathbf{D} + \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}) \tag{36}$$

By comparing Equation (36) and Equation (33), we know that an extra term $\mathbf{LD}^{-1}\mathbf{U}$ appears in Equation (36). For diagonally dominant and narrowband matrix, $\mathbf{LD}^{-1}\mathbf{U}$ is negligible. The purpose of factorization (36) is to eliminate the need to invert the preconditioning matrix $\mathbf{P}$. Right-preconditioning is usually adopted in GMRES algorithm because with right-preconditioning, the same residual as the original non-preconditioned system is minimized in the GMRES algorithm.

By applying the right-preconditioning to the Jacobian-free GMRES solver, we obtain the new form of Equation (30).

$$\tilde{\mathbf{J}}\mathbf{P}^{-1}\mathbf{v} \approx [\mathbf{R}(\mathbf{u}+\varepsilon\mathbf{P}^{-1}\mathbf{v}) - \mathbf{R}(\mathbf{u})]/\varepsilon \tag{37}$$

It has to be stressed that the function $\mathbf{R}$ in Equations (20), (30) and (37) is evaluated following the second-order reconstruction procedure. Numerical experience has shown that using the Jacobian matrix resulting from the lower-order, more dissipative flux function to precondition the Jacobian matrix resulting from higher-order, more accurate flux functions will not compromise the final solution. Even with this simple preconditioning matrix, the convergence of the GMRES solver can be significantly improved.

Before Equation (37) can be implemented, $\mathbf{v}^{\#} = \mathbf{P}^{-1}\mathbf{v}$ must be solved first. This can be done by solving

$$\mathbf{P}\mathbf{v}^{\#} = \mathbf{v} \tag{38}$$

for $\mathbf{v}^{\#}$. Substituting Equation (36) into Equation (38) yields

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\mathbf{v}^{\#} = \mathbf{v} \tag{39}$$

which can be solved in two steps in which the block forward sweep

$$(\mathbf{D}+\mathbf{L})\mathbf{v}^{*} = \mathbf{v} \tag{40}$$

is followed by the block backward sweep

$$(\mathbf{D}+\mathbf{U})\mathbf{v}^{\#} = \mathbf{D}\mathbf{v}^{*} \tag{41}$$

In Equations (40) and (41), the $\mathbf{L}$ and $\mathbf{U}$ operators are computed when needed, thus completely eliminating the need to store the preconditioning matrix. Therefore, the current FV solver for the momentum equation is truly matrix-free.

## 4. MATRIX-FREE FINITE ELEMENT SOLVER FOR THE POISSON EQUATION

Within each physical timestep, we have to solve a Poisson equation (7) for $q'$, the incremental auxiliary variable. Since the Poisson equation is of elliptic type, it can be best discretized using the node-based Galerkin FE method.

With the help of linear nodal basis functions, we can express $q'$ at any location in the domain using the solutions at nodes

$$q' = \sum_{i=1}^{nn} \phi_i q'_i \tag{42}$$

The basis function is Lagrangian defined in each element. Therefore, the resulting $q'$ field is elementwise linear for triangles or bilinear for quadrilaterals with $C^0$ continuity on element borders.

The variational form of Equation (7) is obtained by multiplying Equation (7) with the shape function and integrating by parts

$$\int_\Omega \nabla\phi\cdot\nabla q'\,\mathrm{d}\Omega = \frac{\rho\alpha_1}{\Delta t}\int_\Omega \nabla\phi\cdot\tilde{\mathbf{u}}\,\mathrm{d}\Omega + \int_\Gamma \phi\left(\mathbf{n}\cdot\nabla q' - \frac{\rho\alpha_1}{\Delta t}\mathbf{n}\cdot\tilde{\mathbf{u}}\right)\mathrm{d}\Gamma \tag{43}$$

Taking the boundary condition Equation (9a) into account, we have

$$\int_{\Omega} \nabla\phi \cdot \nabla q' \,\mathrm{d}\Omega = \frac{\rho\alpha_1}{\Delta t} \int_{\Omega} \nabla\phi \cdot \tilde{\mathbf{u}} \,\mathrm{d}\Omega - \frac{\rho\alpha_1}{\Delta t} \int_{\Gamma_D+\Gamma_n} \phi\mathbf{n} \cdot \tilde{\mathbf{u}} \,\mathrm{d}\Gamma \tag{44}$$

Note that on $\Gamma_N$, $q'$ is specified according to Equation (9b) and hence $\Gamma_N$ does not appear in Equation (44). Also note that usually only the inflow boundary will provide a non-zero contribution to the boundary integral in Equation (44). The discretization expressed in Equation (44) leads to a linear equation system with the stiffness matrix as the coefficient matrix. The system is again solved by the GMRES solver. To save memory, we are not forming the global stiffness matrix. Instead, we follow a matrix–free implementation in [20]. This implementation is memory efficient at the price of repeated computation of the matrix–vector product. At the current status of implementation, we use the simple diagonal preconditioner as in [20]. We have to admit that this simple preconditioning is not effective enough in practical applications. We have to use a relatively large size (say 100) of the Krylov space in the GMRES solver to ensure sufficient convergence of the Poisson system.

Recall that the velocity unknowns are stored at element centres while the auxiliary variable is stored at vertices. After $q'$ is obtained through Equation (44), the velocity at the centre of element $i$ can be updated (i.e. *corrected* or *projected*) according to

$$\mathbf{u}_i^{n+1} = \tilde{\mathbf{u}}_i - \frac{\Delta t}{\rho\alpha_1} \sum_{k=1}^{\mathrm{nen}} q'_k \nabla\phi_k \tag{45}$$

where nen is the number of nodes of element $i$.

The node-based auxiliary variable is simply updated *via*

$$q^{n+1} = q^n + q' \tag{46}$$

The pressure at the centre of element $i$ can be updated in a similar way to Equation (42) (also cf. Equation (12))

$$p_i^{n+1} = \sum_{k=1}^{\mathrm{nen}} [q_k^n + q'_k - \mu(\nabla \cdot \tilde{\mathbf{u}})_k]\phi_k \tag{47}$$

The pressure field updated in this way is free of unphysical oscillations and unphysical boundary layers, as will be verified through numerical tests in Section 6. As can be seen in Equation (47), the accuracy of the pressure depends not only on the accuracy of the auxiliary variable, but also on the accuracy of the node-based evaluation of the velocity divergence. $\nabla \cdot \tilde{\mathbf{u}}$ must be computed in a reliable way to ensure the accuracy of the pressure. We will address this issue in the next section.

## 5. MORE IMPLEMENTATION DETAILS

For completeness and clarity, we provide some more implementation details in this section. These details include the way to compute the velocity gradients inside each cell and a reliably way to compute the solution and its derivatives at vertices.

### *5.1. Velocity gradients inside the cell*

To compute the velocity gradients inside each cell, either least square method or Gauss theorem can be used. In our earlier implementation [21], we use the least square method based on the solution at vertices of the cell. However, we found that the Gauss theorem implementation is more efficient on distributed-memory computers and yields as good results as the least square method and adopted it in our more recent implementation [13]. The Gauss theorem states that for each component of the velocity $\mathbf{u}$, denoted by $u$, at cell $i$

$$\nabla u_i = \frac{1}{|\Omega_i|} \sum_{k=1}^{n_\mathrm{f}} u_k A_k \mathbf{n}_k \tag{48}$$

where $u_k$ is the interpolated solution at the $k$th face centre. $u_k$ is obtained by taking the arithmetic average of the interpolated solutions at vertices. $A_k$ and $\mathbf{n}_k$ are the area and outward unit normal of $k$th face, respectively. Hence, the key is to compute the solution at vertices accurately. Since the solution is only known at cell centres, we must use some type of interpolation method to obtain the solution at mesh nodes. We used the pseudo-Laplacian averaging in our compressible solver [13, 21]. However, we realize the method described in the next sub-section is more useful because it not only provides the solution itself at nodes, but also the gradient information at nodes. Recall that from Equation (47), we need the gradient information to compute the velocity divergence.

### *5.2. Solution and its derivatives at vertices*

The interpolation method is based on the least square minimization and was first proposed by Zhu and Zienkiewicz [22]. This method has been proven to lead to superconvergent recovery of nodal derivatives. We assume the solution at the small vicinity region surrounding a specific node varies linearly (cf. Figure 3), i.e.

$$u = a + bx + cy \tag{49}$$

The three unknown coefficients in Equation (49) are determined through the least square method, i.e. minimizing the sum of the squares of $u(\mathbf{x}_i) - u_i^h$ where $\mathbf{x}_i$ is the location of the centre of cell $i$ surrounding the vertex, $u(\mathbf{x}_i)$ is computed according to Equation (49) and $u_i^h$ is the known solution at the centre of cell $i$. Standard least square method leads to the following symmetric

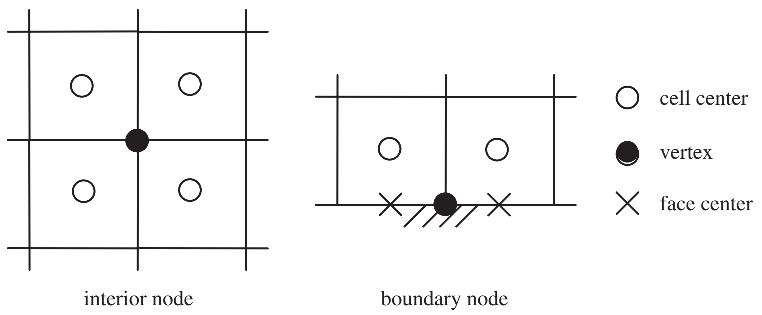


Figure 3. Stencil to compute the solution and its derivatives at a vertex.

equation system:

$$\begin{bmatrix} N & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i y_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_i u_i \\ \sum_i u_i x_i \\ \sum_i u_i y_i \end{bmatrix} \tag{50}$$

where $N$ is the number of cells surrounding the vertex. Equation (50) can then be solved for the coefficients $a$, $b$ and $c$. The derivatives of $u$ are simply

$$u_x = b \quad \text{and} \quad u_y = c \tag{51}$$

For interior nodes, the linear assumption is sufficient to provide accurate recovery of the solution and its derivatives. However, we found for boundary nodes, it is essential to assume the solution varies bilinearly for better accuracy of solution derivatives, i.e.

$$u = a + bx + cy + dxy \tag{52}$$

To provide sufficient supporting stencil, we can add the face centres as shown in Figure 3 where two cell centres plus two face centres are adequate for bilinear reconstruction. The derivatives of $u$ can be computed according to

$$u_x = b + dy \quad \text{and} \quad u_y = c + dx \tag{53}$$

If the stencil is not sufficient to support the bilinear reconstruction, which is possible at the corners of the computational domain, we should have to resort to the linear reconstruction.

## 6. NUMERICAL EXAMPLES

In this section, we present several numerical experiments to demonstrate the performance of the current hybrid scheme.

### *6.1. Case I: a case with analytical solution*

This problem has been studied in [23, 24] to validate an incompressible flow solver. We use this problem to numerically verify the spatial and temporal convergence rates of the current hybrid solver.

According to [23], the two-dimensional incompressible NSE are numerically solved on a unit square domain $\Omega = [0, 1] \times [0, 1]$. The Reynolds number is 100. The analytical solution is given by

$$\begin{aligned} u(t, x, y) &= (t+1)^2 x^2 (1-x)^2 (2y - 6y^2 + 4y^3) \\ v(t, x, y) &= (t+1)^2 y^2 (1-y)^2 (-2x + 6x^2 - 4x^3) \\ p(x, y) &= x^2 - y^2 \end{aligned} \tag{54}$$

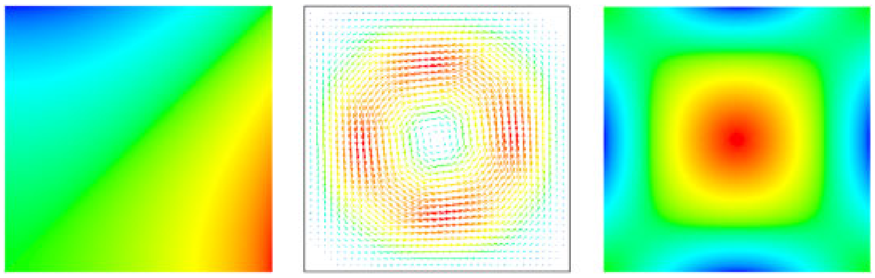


Figure 4. Initial solutions of Case I. Left: pressure; mid: velocity and right: vorticity.

Substituting the exact solution into the incompressible NSE, we can obtain the required body force field, $\mathbf{g}=(g_1, g_2)$, as follows

$$\begin{aligned} g_1(t,x,y) = {} & 2(t+1)x^2(1-x)^2(2y-6y^2+4y^3) \\ & + u(t+1)^2(2y-6y^2+4y^3)(2x-6x^2+4x^3) \\ & + v(t+1)^2x^2(1-x)^2(2-12y+12y^2) \\ & - 0.01(t+1)^2(2y-6y^2+4y^3)(2-12x+12x^2) \\ & - 0.01(t+1)^2x^2(1-x)^2(-12+24y)+2x \end{aligned} \tag{55a}$$

$$\begin{aligned} g_2(t,x,y) = {} & 2(t+1)y^2(1-y)^2(-2x+6x^2-4x^3) \\ & + u(t+1)^2y^2(1-y)^2(-2+12x-12x^2) \\ & + v(t+1)^2(2y-6y^2+4y^3)(-2x+6x^2-4x^3) \\ & - 0.01(t+1)^2y^2(1-y)^2(12-24x) \\ & - 0.01(t+1)^2(-2x+6x^2-4x^3)(2-12y+12y^2)-2y \end{aligned} \tag{55b}$$

The initial conditions at $t=0$ are the following velocity and pressure fields:

$$\begin{aligned} u_0 &= x^2(1-x)^2(2y-6y^2+4y^3) \\ v_0 &= y^2(1-y)^2(-2x+6x^2-4x^3) \\ p_0 &= x^2-y^2 \end{aligned} \tag{56}$$

and no-slip condition ($\mathbf{u}=0$) is applied on all boundaries. As can be seen from Figure 4, the velocity field is an anticlockwisely rotating vortex. The body force acts as the driving force to increase the velocity magnitude with time. The pressure field remains unchanged with time.

We want to use this case to demonstrate the rates of spatial and temporal convergence of the present hybrid scheme. Figure 5 shows the quadrilateral and triangular meshes we used for this purpose. The triangular mesh has the same characteristic element size as the quadrilateral one. First

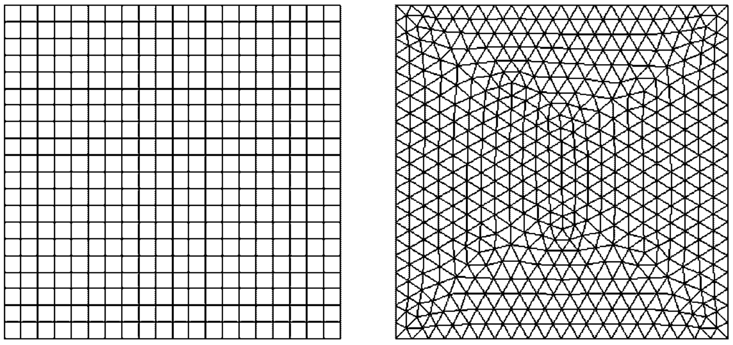


Figure 5. Meshes for Case I. Left: $20 \times 20$ quadrilateral mesh; right: unstructured triangular mesh with the same edge size.

Table I. Rate of spatial convergence on quadrilateral meshes.

| Mesh size $h$ | $\|\text{err}_m\|$ | $r$ | $\|\text{err}_p\|$ | $r$ |
|---|---|---|---|---|
| 0.05 | 0.53235E−07 | — | 0.25007E−07 | — |
| 0.025 | 0.37310E−08 | 3.83 | 0.30310E−08 | 3.04 |
| 0.0125 | 0.29247E−09 | 3.67 | 0.22693E−09 | 3.74 |
| 0.00625 | 0.19940E−10 | 3.87 | 0.13155E−10 | 4.11 |

$\Delta t = 0.0001$.

we define the following RMS error norms for momentum and pressure, respectively, to measure the differences between the numerical solution and the analytical solution

$$\|e_m\| = \sqrt{\frac{\sum_{i=1}^{\text{ne}} [(u^h - u^e)A_i]^2 + [(v^h - v^e)A_i]^2}{\text{ne}}} \quad \text{and} \quad \|e_p\| = \sqrt{\frac{\sum_{i=1}^{\text{ne}} [(p^h - p^e)A_i]^2}{\text{ne}}} \tag{57}$$

where the superscripts '$h$' and '$e$' represent numerical solution and analytical solution, respectively. 'ne' is the number of elements and $A_i$ is the area of each element. The solutions are those at element centres.

To demonstrate the spatial convergence rate, we subdivide the initial coarse mesh isotropically to obtain a series of finer meshes. To ensure the spatial discretization error will dominate over the temporal error, we use a small timestep, $\Delta t = 0.0001$, and run 100 timesteps on each of these meshes. Tables I and II tabulate the error norms and the corresponding convergence rates on all meshes. The order of convergence is computed according to

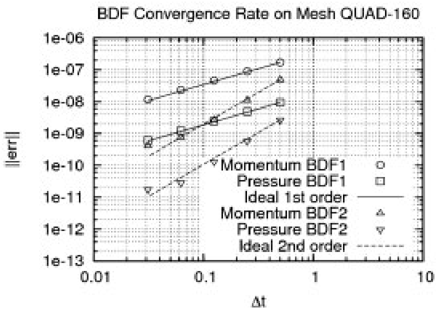
$$r = \frac{\log(\|e\|_{h+1}/\|e\|_h)}{\log 0.5} \tag{58}$$

with the subscript '$h$' denoting the grid refinement level. As can be seen, the current scheme is apparently superconvergent at least for this case, which is quite surprising because the scheme was designed to be only second-order accurate in space. We also observe the similar convergence

Table II. Rate of spatial convergence on triangular meshes.

| Mesh size $h$ | $\|\text{err}_m\|$ | $r$ | $\|\text{err}_p\|$ | $r$ |
|---|---|---|---|---|
| 0.05 | 0.20035E−06 | — | 0.77954E−07 | — |
| 0.025 | 0.10106E−07 | 4.31 | 0.41594E−08 | 4.23 |
| 0.0125 | 0.33023E−09 | 4.94 | 0.23846E−09 | 4.12 |
| 0.00625 | 0.10998E−10 | 4.91 | 0.14516E−10 | 4.04 |

$\Delta t = 0.0001$.



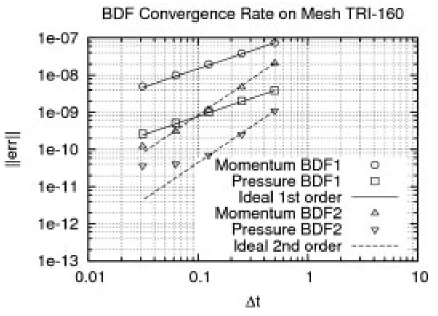


Figure 6. Rate of temporal convergence. Left: quadrilateral mesh and right: triangular mesh.

rates for both velocity and pressure. However, this spatial superconvergence phenomenon needs further investigation.

We also demonstrate the temporal convergence rate of the current hybrid scheme in Figure 6. To ensure the spatial discretization error is small compared with the temporal error, we use very fine meshes for this purpose. The fine meshes are obtained by isotropically subdividing the meshes shown in Figure 5 three times. As can be seen in Figure 6, both BDF1 and BDF2 yield expected rates of convergence for both momentum and pressure. Exceptions occur when the timestep is small and the temporal error becomes comparable with the spatial error.

The correct rates of temporal convergence of pressure justify the need to use Equations (12) and (47) to update the real pressure. We can further justify this by looking at the error graphs shown in Figure 7. Note that the vertical heights in Figure 7 represent the errors between the numerical solution and the exact solution. For the graphs about the auxiliary variable and the pressure, the vertical axis has been scaled by 5000 and for the graph about the velocity divergence, the vertical axis has been scaled by 100. Clearly, both the auxiliary variable and the velocity divergence show an artificial boundary layer. It is this boundary layer that causes the loss of accuracy. The pressure error is nearly ideally smooth without any visible boundary layer. The smoothness of the pressure error is crucial to the correct temporal convergence rate. This, on the other hand, verifies that the method described in Section 5 accurately compute the velocity divergence even at boundaries.

## *6.2. Case II: lid-driven cavity problem* ($Re = 1000$)

The lid-driven cavity flow is a well known benchmark problem extensively used to validate an incompressible flow solver. Despite the simple geometry, the lid-driven cavity flow exhibits quite complex physics such as the vortex pattern on the recirculating corners of the cavity depending on

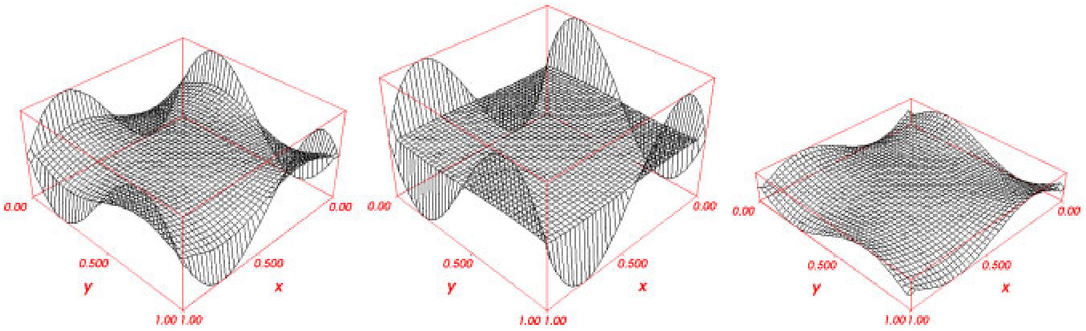


Figure 7. Solution errors of Case I on $40 \times 40$ quadrilateral mesh. Left: auxiliary variable; mid: velocity divergence and right: pressure.

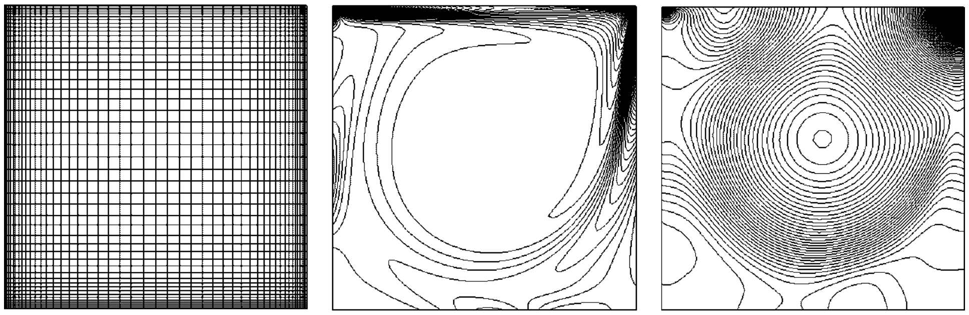


Figure 8. Mesh and solutions for Case II. Left: $48 \times 48$ quadrilateral mesh, $Re = 1000$; mid: vorticity and right: pressure.

the Reynolds number. Steady solution exists for Reynolds number up to 21 000. We use the current hybrid solver to simulate the case with $Re = 1000$ on three consecutively refined quadrilateral meshes on the $[0, 1] \times [0, 1]$ domain. The coarsest mesh consists of $24 \times 24$ quadrilaterals and is shown in Figure 8. The grid lines are clustered near the four corners and walls of the cavity. The top of the cavity is the moving lid with $u = 1.0$ and $v = 0.0$. The remaining boundaries are no-slip boundary with $u = v = 0.0$.

Since we are seeking steady-state solutions, only one non-linear iteration is performed for the momentum equation within each physical timestep. We run 1500 timesteps on the two coarser meshes with $\Delta t = 0.075$ and 2000 timesteps for the finest $96 \times 96$ mesh with $\Delta t = 0.05$. The momentum residual drops about 5.5 orders or magnitude for all three runs. Figure 8 shows the $48 \times 48$ mesh and the computed vorticity and pressure fields. The pressure field is smooth without unphysical wiggles seen. Figure 9 shows the residual convergence history on the $48 \times 48$ mesh. We conclude that the current hybrid scheme converges well for steady-state simulations.

We also compare our solutions with the most referenced solutions by Ghia *et al.* [25]. Figure 10 shows the $u$-velocity along the vertical midline and the $v$-velocity along the horizontal midline, compared with the reference values. As can be seen, with increasing refinement of the mesh, the present solution matches the reference solution that was obtained on a mesh finer than all

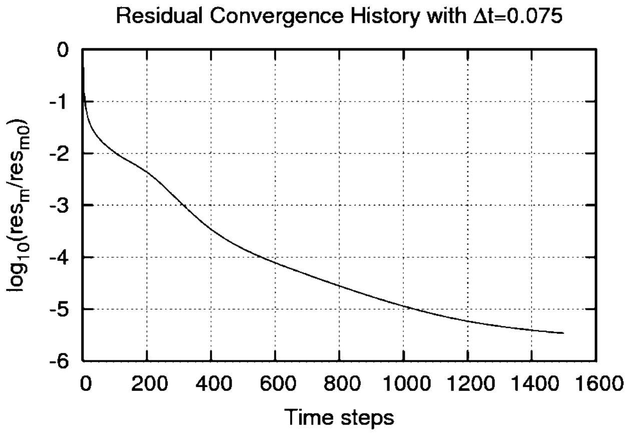


Figure 9. Residual convergence history on $48 \times 48$ quadrilateral mesh.

meshes we are using. The solution obtained on the $96 \times 96$ mesh already agrees very well with the reference solution.

## 6.3. *Case III: flow around circular cylinder* ($Re = 100$)

Now we turn to a non-stationary problem. This case is taken from one of the benchmark problems proposed in [26]. This case is about a laminar incompressible flow around a circular cylinder placed in a channel. Abundant numerical results obtained *via* various numerical solvers are tabulated in [26]. Therefore, this problem has been widely accepted as a standard test case to verify and validate an incompressible solver. The geometry and boundary conditions are illustrated in Figure 11. Since no-slip boundary conditions are applied at the top and the bottom walls of the channel, a special parabolic inflow velocity profile must be given to account for the zero velocity at the inlet tips of both walls. According to [26], the velocity profile is

$$u(0, y, t) = 4u_m y(H - y)/H^2, \quad v = 0$$

with $u_m = 1.5\,\mathrm{m/s}$ and $H = 0.41\,\mathrm{m}$ as the height of the channel. The mean velocity at inflow is $\bar{U} = 2u_m/3 = 1.0\,\mathrm{m/s}$. The Reynolds number based on the mean velocity is 100. At this Reynolds number, the flow behind the cylinder is expected to become non-stationary and periodic Karman vortex shedding should be seen.

As shown in Figure 12, the mesh used here is an unstructured quadrilateral one consisting of 14 568 cells and 14 876 nodes. The total number of unknowns is 29 136 for velocity and 14 803 for pressure. There are 100 points on the cylinder surface and the mesh thickness of the first layer next to the cylinder is 0.001 m. We start from the initial condition given by the inlet velocity profile everywhere. The size of the Krylov space for the momentum equations is 20. The timestep used in this simulation is 0.002 s. At this timestep, at least three order of magnitude drop of momentum residual can be reached with two or three non-linear iterations within each timestep. The Poisson equation is solved with 100 Krylov vectors and considered converged when five orders of magnitude of residual is dropped. We have run 4150 timesteps to reach the final time $t = 8.3\,\mathrm{s}$.

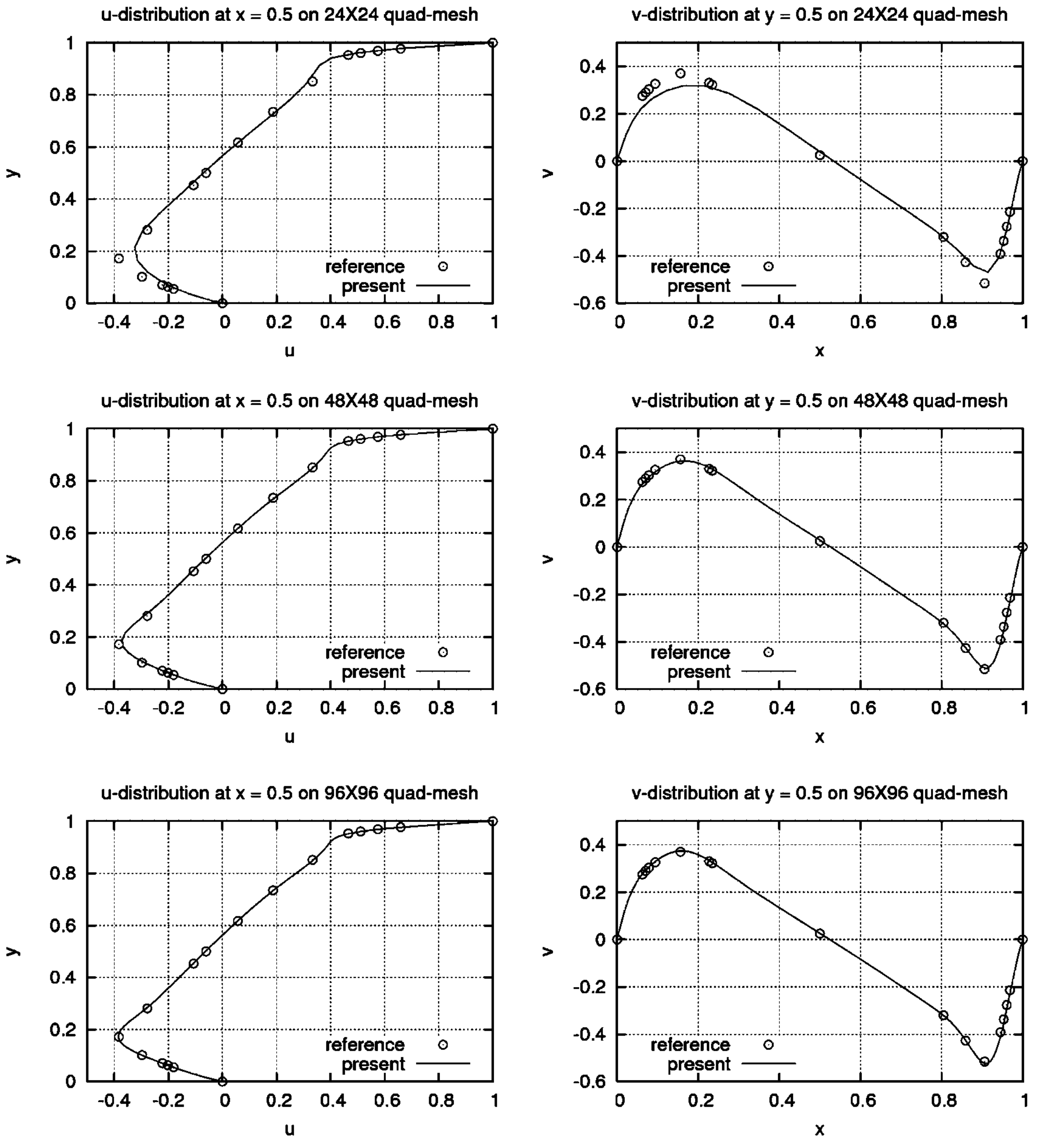


Figure 10. Velocity along the horizontal and vertical mid-line on three meshes of different refinement.

Figure 13 shows the numerical solutions at an instant corresponding to a flow state with maximum lift. The periodic Karman vortex street can be clearly seen from these figures. Figure 14 shows the drag and lift coefficients histories with time. As can be seen, the flow enters the stable periodic state as early as $t \approx 4$ s. Figure 15 provides a close-up view of the drag and lift histories within a lift period starting from $t = 7.936$ s. The period for the lift *vs* time is 0.338 s that is twice the period

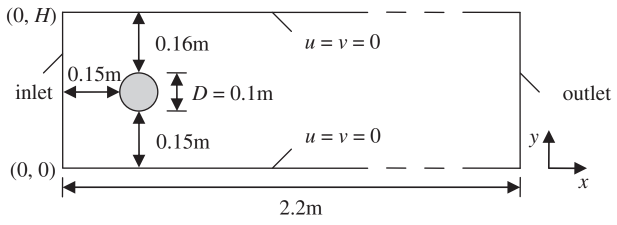


Figure 11. Geometry and boundary conditions for Case III.

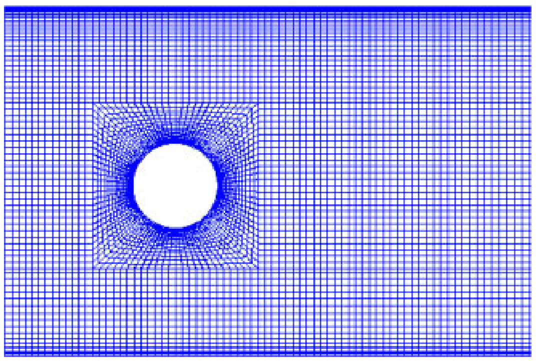


Figure 12. A portion of the unstructured quadrilateral mesh for Case III.

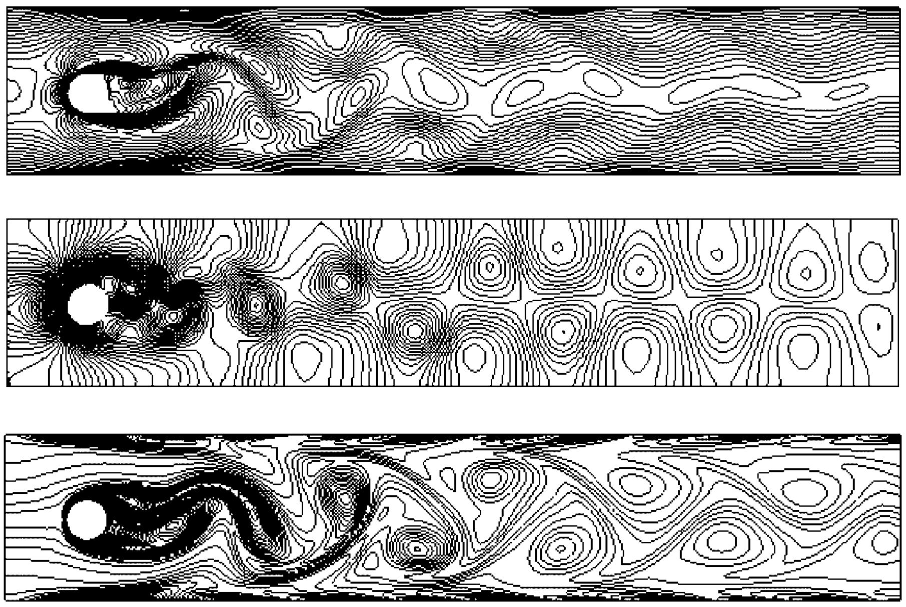
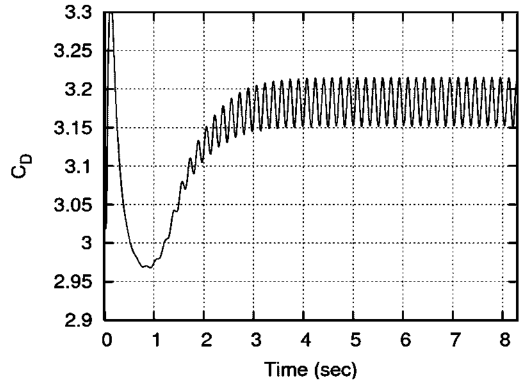


Figure 13. Solutions of Case III. Top: velocity magnitude; mid: pressure and bottom: vorticity. The instant corresponds to the flow state with $c_{L\,max}$.



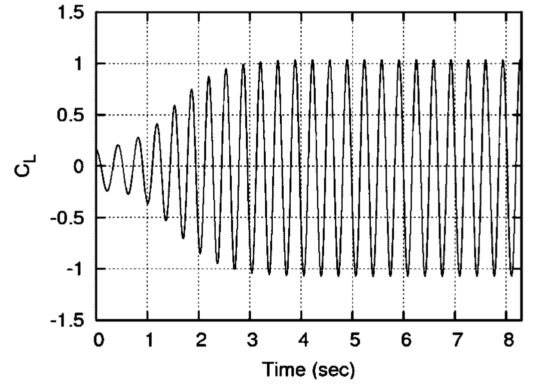
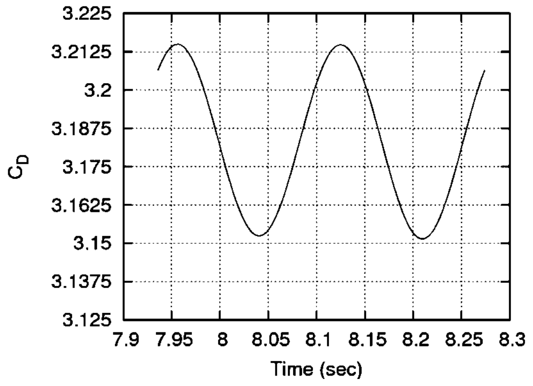


Figure 14. Drag and lift histories for Case III.



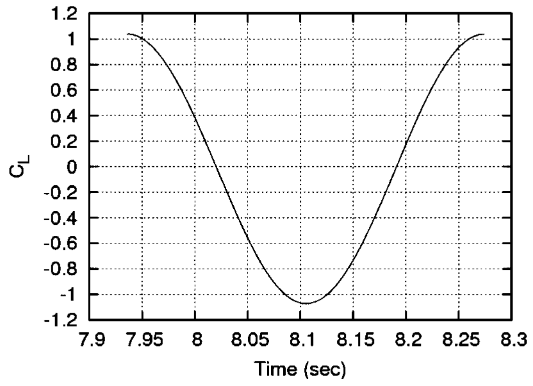


Figure 15. Drag and lift histories within a lift period for Case III.

Table III. Comparison between the present solutions and the reference solutions [26] for Case III.

| | | $c_{\mathrm{D\,max}}$ | $c_{\mathrm{L\,max}}$ | $St$ | $\Delta p$ |
|---|---|---|---|---|---|
| Reference | Lower bound | 3.22 | 0.99 | 0.295 | 2.46 |
| | Upper bound | 3.24 | 1.01 | 0.305 | 2.5 |
| Present | | 3.215 | 1.0374 | 0.2959 | 2.4687 |

for the drag *vs* time. From Figure 15, we can also observe that the phase of the drag coefficient is a little away from that of the lift coefficient. To quantitatively compare the present solution with those available in [26], we compute the maximum drag coefficient $c_{\mathrm{D\,max}}$, maximum lift coefficient $c_{\mathrm{L\,max}}$, the Strouhal number $St = D/(T\bar{U})$ and the pressure difference $\Delta p$ between the front and rear stagnation points on the cylinder surface at $t = t_0 + 0.5T$ where $t_0$ corresponds to the flow state with $c_{\mathrm{L\,max}}$ and $T$ is the period of the lift *vs* time curve. $c_{\mathrm{D\,max}}$ and $c_{\mathrm{L\,max}}$ are computed as the average of the last 20 and 10 periods, respectively. Table III lists the results. The comparison shows that the present Strouhal number and pressure difference are within the lower bound and the upper bound provided in the reference, and the drag and lift coefficients also agree favourably with the reference values.

## 7. CONCLUSIONS

In this paper, we present a detailed description of the development of a hybrid FV/element solver for laminar incompressible NSEs on unstructured meshes. The current solver belongs to a pressure-correction or projection method. A segregated approach is used to decouple the pressure from the velocity. A fractional step method is adopted to compute the velocity field. An intermediate velocity is first obtained by solving the original momentum equations with the cell-centred FVM. The FV solver is fully implicit and truly matrix-free using the Jacobian-free GMRES solver together with the matrix-free LU-SGS preconditioner. The fractional step leads to the Hodge decomposition of the intermediate velocity into the sum of a divergence-free velocity field and a curl-free vector field that is the gradient of an auxiliary variable. The auxiliary variable is closely related to the real pressure. A Poisson equation is derived from the Hodge decomposition and is solved by the node-based Galerkin FEM for the auxiliary variable. The auxiliary variable is used to update the velocity field and the pressure field as well. To annihilate the artificial pressure boundary layer, the update of the pressure must take the discrete velocity divergence into account. Since the velocity and the auxiliary variable are stored at different locations on the mesh, the current scheme is considered as a staggered-mesh scheme. However, it is unlike the conventional staggered grid scheme because it uses a different storage scheme.

A test case with analytical solutions demonstrates the superconvergence feature of the current scheme in space for both velocity and pressure. However, this superconvergence phenomenon needs further investigation and analysis. More importantly, the temporal convergence exhibits the correct rates (first order for BDF1 and second order for BDF2) for both velocity and pressure. The test on the classic lid-driven cavity problem shows that the current hybrid solver generates accurate velocity profiles inside the cavity. The test also shows the solver converges well for steady-state simulations. A more practical test case of the flow passing around a circular cylinder placed in a channel further verify that the current solver is accurate in predicting the force coefficients and the Strouhal number.

All test cases shown in this paper are of relatively low Reynolds number flows in which it is unnecessary to employ any types of slope limiter for stability purpose. However, in high Reynolds number flows, the boundary layer is very thin and velocity gradients are so high that it may become necessary to employ a slope limiter or other stabilization techniques to ensure the stability of the solver. We are currently investigating the performance of the current hybrid solver for high Reynolds number flows and will report the results in a future paper.

ACKNOWLEDGEMENTS

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 DAAD19-03-D-0001. We are also grateful for the funding support from the Northrop Grumman Ship Systems.

## REFERENCES

1. Gresho PM, Sani RL. *Incompressible Flow and the Finite Element Method*, *Volume Two*: *Isothermal Laminar Flow*. Wiley: New York, 2000.
2. Chorin AJ. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 1967; **2**:12–26.
3. Turkel E. Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics* 1987; **72**:277–298.
4. Brooks AN, Hughes TJR. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**:199–259.
5. Tezduyar TE, Mittal S, Ray SE, Shih R. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity–pressure elements. *Computer Methods in Applied Mechanics and Engineering* 1992; **95**:221–242.
6. Hughes TJR, Franca LP, Balestra M. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska–Brezzi condition: a stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**:85–99.
7. Hughes TJR, Franca LP, Hulbert GM. A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/Least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering* 1989; **73**:173–189.
8. Guermond JL, Minev P, Shen J. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:6011–6045.
9. Caretto LS, Gosman AD, Patankar SV, Spalding DB. Two calculation procedures for steady, three-dimensional flows with recirculation. *Proceedings of the Third International Conference on Numerical Methods*, vol. 19. Springer-Verlag: New York, 1972; 60–68.
10. Issa RI. Solution of the implicitly discretized fluid flow equations by operator-splitting. *Journal of Computational Physics* 1985; **62**:40–65.
11. Aliabadi S, Johnson A, Abedi J. Stabilized-finite-element/interface-capturing technique for parallel computation of unsteady flows with interfaces. *Computer and Fluids* 2003; **32**:535–545.
12. Aliabadi S, Johnson A, Abedi J, Tu S, Tate A. Simulation of contaminant dispersion on the Cray X1: Verification and implementation. *Journal of Aerospace Computing*, *Information and Communication* 2004; **1**:341–361.
13. Tu S, Watts M, Fuller A, Patel R, Aliabadi S. Development and performance of *CaMEL_Aero*, a truly matrix-free, parallel and vectorized unstructured finite volume solver for compressible flows. *Proceedings of the 25th Army Science Conference*, Orlando, FL, November 2006.
14. Timmermans LJP, Minev PD, Vosse FNVD. An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids* 1996; **22**:673–688.
15. Guermond JL, Shen J. On the error estimates for the rotational pressure-correction projection methods. *Mathematics of Computation* 2004; **73**:1719–1737.
16. Mathur SR, Murphy JY. A pressure-based method for unstructured meshes. *Numerical Heat Transfer*, *Part B* 1997; **31**:195–215.
17. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, PA, 2003.
18. Knoll D, Keyes D. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics* 2004; **193**:357–397.
19. Luo H, Baum J, Lohner R. A fast matrix-free implicit method for compressible flows on unstructured grids. *Journal of Computational Physics* 1998; **146**:664–690.
20. Aliabadi S. Parallel finite element computations in aerospace applications. *Ph.D. Thesis*, University of Minnesota, 1994.
21. Tu S, Aliabadi S, Johnson A, Watts M. A robust parallel implicit finite volume solver for high-speed compressible flows. *43rd AIAA Aerospace Sciences Meeting and Exhibit*, *AIAA paper 2005-1396*, Reno, Nevada, 2005.
22. Zhu JZ, Zienkiewicz OC. Adaptive techniques in the finite element method. *Communication in Applied Numerical Methods* 1988; **4**:197–204.
23. Valli AMP. Control strategies for timestep selection in finite simulation of incompressible flows and coupled heat and mass transfer. *Programa de Engenharia Civil*, *COPPE*, 2001.
24. Carey GF, Krishnan R. Penalty approximation of Stokes flow. *Computer Methods in Applied Mechanics and Engineering* 1982; **35**:169–206.

25. Ghia U, Ghia KN, Shin CT. High-*Re* solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics* 1982; **48**:387–411.
26. Turek S, Schafer M. Benchmark computations of laminar flows around a cylinder. In *Flow Simulation with High-performance Computers II*, Hirschel EH (ed.), Notes on Numerical Fluid Mechanics, vol. 52. Vieweg, 1996; 547–566.