Numerical Solutions for Pressure-Velocity Coupling in Steady Flow Systems using the SIMPLE Algorithm

Aaron Outhwaite
Department of Process Engineering and Applied Sciences, Dalhousie University
Email: aaron.outhwaite@dal.ca

Abstract

A numerical method is described using the Navier-Stokes equations to solve pressure-velocity coupling in incompressible steady flow systems. Both the lid driven cavity flow problem and backward step flow problem are solved using the finite volume method and SIMPLE algorithm with hybrid scheme in MATLAB. For the lid driven cavity problem, Reynolds numbers of 400, 1000, 5000 and 10000 are simulated on grids of 50^2, 100^2 and 150^2, while for the backward step problem, a fixed grid of 300 x 30 is used to test Reynolds numbers of 10, 100, 200, 300 and 500. For both problems, validation with the literature indicates the suitability of the present methods.

Keywords

Computational Fluid Dynamics; 2D Pressure-Velocity Coupling; Finite Volume Method; SIMPLE Algorithm; MATLAB

1. Introduction

In computational fluid dynamics (CFD), two-dimensional (2D) pressure-velocity coupling in steady flow systems utilize the Navier-Stokes equations for continuity and momentum. However, the solution to these equations presents a unique challenge in incompressible steady flow systems, owing to the quasilinear nature of the momentum equations, as well as the need to maintain continuity with an unknown pressure field when the pressure is independent of density (Versteeg + Malalasekera, 2007). To overcome these issues, the pressure field is specified indirectly using the continuity equation, and then solved using an iterative strategy (Patankar, 1980).

Previous numerical work on pressure-velocity coupling in the literature has tended to use the 2D lid driven cavity flow and 2D laminar flow over a backward facing step problems as comparison exercises with experimental work, or as benchmark techniques to test different iterative strategies (Togun et al., 2014; Sahin & Owens, 2003; Barton, 1994). According to Ertuk (2009), the lid driven cavity problem is the most widely studied problem in CFD, owing both to its relatively simple computational implementation, and complicated behaviour, including counter rotating vortices appearing at the corners of the cavity. The backward facing step problem has also been extensively studied, as it demonstrates shear-layer separation and reattachment common to many industrial fluid flow problems, including airfoils, automobiles, combustors, diffusers, reactor design and pipe and duct expansions (Terzi, 2004). For both problems, discretization using the finite volume method (FVM) on a staggered grid, followed by an iterative solution procedure defined by the SIMPLE algorithm have been well established (Versteeg + Malalasekera, 2007; Ferziger & Peric, 2002; Patankar, 1980), and the interested reader is directed to the work of these authors for a complete review of these methods.

The purpose of the present work is to develop a basic generalized CFD method in MATLAB to solve the Navier-Stokes equations for pressure-velocity coupling in steady flow systems using the finite volume method (FVM) with a staggered grid and the SIMPLE algorithm as an iterative technique. This method is applied to both the lid driven cavity and backward facing step benchmark problems, with validation based on numerical and experimental results presented in the literature.

2. Model Description

The governing equations for pressure-velocity coupling for incompressible steady flow systems in the non-dimensional differential form using primitive variables is (Yapici & Uludag, 2013; Kim & Choi, 2000; Williams & Baker, 1997) are:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

$$\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial p}{\partial x} \tag{2}$$

$$\frac{\partial v^2}{\partial y} + \frac{\partial uv}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial p}{\partial y} \tag{3}$$

To solve Eq. 1 – 3, two main difficulties must be overcome. First, the convective terms in Eq. 2 and 3 contain non-linear quantities that must be solved iteratively from a guessed initial velocity field (Patankar, 1980). Second, the pressure field is unknown and must also be solved iteratively (Versteeg + Malalasekera, 2007). However, despite pressure forming part of the source term in Eq. 2 and 3, it does not appear in the continuity equation (Eq. 1). A solution to this problem is to use a staggered mesh approach for velocity components, wherein the velocities are computed at the faces of the control volume (CV), while the pressure is computed at the center of the CV (Recktenweld, 2014; Versteeg + Malalasekera, 2007; Patankar, 1980). The basic arrangement for 2D flow calculations using a staggered grid is described in Fig. 1.

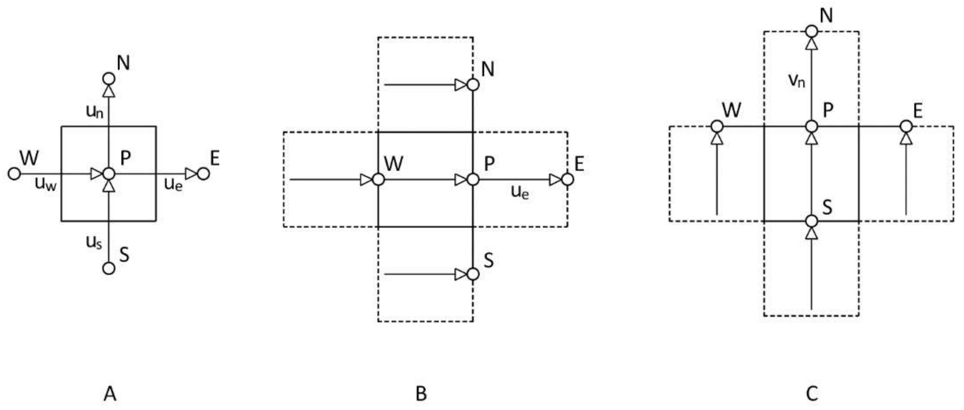

Fig. 1: Forward staggered grid approach with main CV (A), CV and neighbors of u_e (B), and CV and neighbors of v_n (C) (adapted from Recktenwald, 2014).

Using the staggered grid approach described in Fig. 1, both non-linear terms and pressure-velocity coupling can be achieved using an iterative solution strategy. To do this, Eq. 1 – 3 are first be discretized using the finite volume method.

2.1 Model Formulation

Integrating Eq. 2 over the control volume (CV) described in Fig. 1 for u-momentum yields:

$$\iint_V \left[\frac{\partial}{\partial x}(u^2)+\frac{\partial}{\partial y}(uv)\right]dxdy=\frac{1}{Re}\iint_V \left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)dxdy-\iint_V \frac{\partial}{\partial x}(p)dxdy \tag{4}$$

or, through approximation:

$$\frac{\Delta x\Delta y}{\Delta x}(u^2)+\frac{\Delta x\Delta y}{\Delta y}(uv)=\frac{\Delta x\Delta y}{Re}\left[\frac{1}{\Delta x}\left(\frac{\partial u}{\partial x}\right)+\frac{1}{\Delta y}\left(\frac{\partial u}{\partial y}\right)\right]-\left(\frac{\partial p}{\partial x}\right)\Delta x\Delta y \tag{5}$$

Rearranged for the staggered grid described in Fig. 1, Eq. 5 can be written as:

$$[(u^2)\Delta y]_w^e-[(uv)\Delta x]_s^n=\left[\frac{\Delta y}{Re}\left(\frac{\partial u}{\partial x}\right)\right]_w^e+\left[\frac{\Delta x}{Re}\left(\frac{\partial u}{\partial y}\right)\right]_s^n-\left(\frac{\partial p}{\partial x}\right)\Delta x\Delta y \tag{6}$$

The diffusive flux in and out of the CV can be approximated using central differencing for the gradients $\partial u/\partial x$ and $\partial u/\partial y$:

$$\begin{aligned}\left[\frac{\Delta y}{Re}\left(\frac{\partial u}{\partial x}\right)\right]_w^e+\left[\frac{\Delta x}{Re}\left(\frac{\partial u}{\partial y}\right)\right]_s^n \\ =\left(\frac{\Delta y}{Re}\right)_e\left(\frac{u_E-u_P}{\Delta x}\right)-\left(\frac{\Delta y}{Re}\right)_w\left(\frac{u_P-u_W}{\Delta x}\right)+\left(\frac{\Delta x}{Re}\right)_n\left(\frac{u_N-u_P}{\Delta y}\right) \\ -\left(\frac{\Delta x}{Re}\right)_s\left(\frac{u_P-u_S}{\Delta y}\right)\end{aligned} \tag{7}$$

Linearized, the diffusive flux terms become:

$$\left[\frac{\Delta y}{Re}\left(\frac{\partial u}{\partial x}\right)\right]_w^e+\left[\frac{\Delta x}{Re}\left(\frac{\partial u}{\partial y}\right)\right]_s^n=a_e^d u_E+a_w^d u_W+a_n^d u_N+a_s^d u_S-a_p^d u_P \tag{8}$$

where $a_e^d=\Delta y/(\Delta xRe)_e$, $a_w^d=\Delta y/(\Delta xRe)_w$, $a_n^d=\Delta x/(\Delta yRe)_n$, $a_s^d=\Delta x/(\Delta yRe)_s$ and $a_p^d=a_e^d+a_w^d+a_n^d+a_s^d$. The convective flux terms can be approximated in an analogous manner:

$$[(u^2)\Delta y]_w^e-[(uv)\Delta x]_s^n=a_e^c u_E+a_w^c u_W+a_n^c u_N+a_s^c u_S-a_p^c u_P \tag{9}$$

where $a_e^c=(u\Delta y)_e$, $a_w^c=(u\Delta y)_w$, $a_n^c=(v\Delta x)_n$, $a_s^c=(v\Delta x)_s$ and $a_p^d=a_e^c+a_w^c+a_n^c+a_s^c$.
The generalized diffusive-convective flux expression is therefore:

$$a_p u_P=a_e u_E+a_w u_W+a_n u_N+a_s u_S \tag{10}$$

where $a_e=a_e^d+a_e^c$, (etc.) Because the convective flux coefficients a_i^c contain velocity expressions, the diffusive-convective flux coefficient a_i must be estimated. Common approximations are described in Table 1.

Table 1: Common numerical approximations for boundary velocities coefficients (adapted from Versteeg & Malalasekera, 2007).

	Scheme		
	Forward	Central	Hybrid
a_e	$a_e^d + \max[0, -a_e^c]$	$a_e^d - \frac{a_e^c}{2}$	$\max\left[-a_e^c, \left(a_e^d - \frac{a_e^c}{2}\right), 0\right]$
a_w	$a_w^d + \max[0, a_w^c]$	$a_w^d + \frac{a_w^c}{2}$	$\max\left[a_w^c, \left(a_w^d + \frac{a_w^c}{2}\right), 0\right]$
a_n	$a_n^d + \max[0, -a_n^c]$	$a_n^d - \frac{a_n^c}{2}$	$\max\left[-a_n^c, \left(a_n^d - \frac{a_n^c}{2}\right), 0\right]$
a_s	$a_s^d + \max[0, a_s^c]$	$a_s^d + \frac{a_s^c}{2}$	$\max\left[a_s^c, \left(a_s^d + \frac{a_s^c}{2}\right), 0\right]$
a_p	$a_e + a_w + a_n + a_s$ $+(a_e^c - a_w^c + a_n^c - a_s^c)$	$a_e + a_w + a_n + a_s$ $+(a_e^c - a_w^c + a_n^c - a_s^c)$	$a_e + a_w + a_n + a_s$ $+(a_e^c - a_w^c + a_n^c - a_s^c)$

The pressure term in Eq. 6 can be estimated using central differencing as

$$\frac{\partial p}{\partial x}\Delta x \Delta y = (p_P - p_E)\Delta y \tag{11}$$

As such, through substitution and simplification, the discretized u-momentum equation (Eq. 6) becomes:

$$a_p u_P = \sum u_{i,b} a_{i,b}^u + (p_P - p_E)\Delta y + b_u \tag{12a}$$

where b_u represents source-terms arising from the pressure gradient. For v-momentum, Eq. 12a becomes:

$$a_p v_P = \sum u_{i,b} a_{i,b}^v + (p_P - p_N)\Delta x + b_v \tag{12b}$$

The discretized momentum equations can now be solved iteratively. To do this, the SIMPLE algorithm is presented.

2.2 Numerical Methodology

As implemented within an iterative flow solver, the SIMPLE algorithm is (Craft, 2008):

1. Start with initial values
2. Solve discretized momentum equations for u^* and v^* with applied under-relaxation
3. Calculate coefficients and source terms for the pressure correction equation (PCE) and solve for p'
4. Calculate velocity field corrections (u', v') using p' such that $u = u^* + u'$ and $v = v^* + v'$ satisfy continuity
5. Correct velocity field (u, v) and pressure field p

For a forward staggered grid, $u_{i.j} = u_e$ and $v_{i.j} = v_n$. Consequently, Eq. 12 can be written as:

$$a_e^u u_e = \sum a_{i,b}^u u_{i,b} + (p_P - p_E)\Delta y + b_e \tag{13a}$$

and:

$$a_n^v v_n = \sum a_{i,b}^v v_{i,b} + (p_P - p_N)\Delta x + b_n \tag{13b}$$

To satisfy continuity, corrections u', v', and $p'$are applied to the velocity and pressure equations such that the corrected variables are expressed in general as:

$$u = u^* + u' \tag{14a}$$

$$v = v^* + v' \tag{14b}$$

$$p = p^* + p' \tag{14c}$$

Here, estimated velocities u^* and v^* are solved by Eq. 13 as:

$$a_e^u u_e^* = \sum a_{i,b}^u u_{i,b}^* + (p_P^* - p_E^*)\Delta y + b_e \tag{15a}$$

$$a_n^v v_n^* = \sum a_{i,b}^v u_{i,b}^* + (p_P^* - p_N^*)\Delta x + b_n \tag{15b}$$

Using Eq. 14, the velocity corrections $(u'\ v')$ can be estimated by subtracting Eq. 15 from Eq. 13, respectively, yielding:

$$a_e u_e' = \sum a_{i,b}^u u_{i,b}' + (p_P' - p_E')\Delta y \tag{16a}$$

$$a_n v_n' = \sum a_{i,b}^v v_{i,b}' + (p_P' - p_N')\Delta x \tag{16a}$$

For the SIMPLE algorithm, the summation term on the RHS of Eq. 16 are neglected. Consequently, the velocity corrections can be expressed as:

$$u_e' = d_e(p_P' - p_E') \tag{17a}$$

$$v_n' = d_n(p_P' - p_N') \tag{17b}$$

Where $d_e = \Delta y / a_e$ and $d_n = \Delta x / a_n$. The pressure correction p' is solved by discretizing the continuity equation (Eq. 1) as:

$$(u_e - u_w)\Delta y + (u_e - u_w)\Delta x = 0 \tag{18}$$

By substitution of Eq. 17, Eq. 18 becomes:

$$a_P p_P' = a_E p_E' + a_W p_W' + a_N p_N' + a_S p_S' + b_p \tag{19}$$

Where $a_E = d_e \Delta y$, $a_W = d_w \Delta y$, $a_N = d_n \Delta x$, $a_S = d_s \Delta x$, and $a_P = a_E + a_W + a_N + a_S$. The source term b_p is given as:

$$b_p = -u_w^* + u_e^* - v_n^* + v_s^* \tag{20}$$

This simplification requires the application of under-relaxation such that Eq. 14c becomes:

$$p = p^* + \alpha_p p' \tag{21}$$

where α_p is the pressure under-relaxation factor. For steady-flow systems and the SIMPLE algorithm, α_p is defined as being optimal (Ferziger & Peric, 2002) when:

$$\alpha_p = 1.0 - \alpha_u \tag{22}$$

Note here that optimal under-relaxation factors are problem dependent (Patankar, 1980), and may not follow E q. 22 explicitly. For instance, Pantakar (1980) suggests a α_u = 0.5 and α_p = 0.8 for the lid driven cavity problem. In general, small values of $\alpha_u$allow α_p to take any value between 0.1 and 1.0, but with slow convergence, while large values of $\alpha_u$results in faster convergence, but with a restricted range of useful α_p (Ferziger & Peric, 2002). A trial and error method can be used wherein a small under-relaxation factor is applied in the early iterations and increased until convergence is achieved. Here, the velocity under-relaxation factor (α_u) is applied line-by-line by manipulation of Eq. 15 using a pseudo-Jacobi iteration method (Staerdahl, 2016; McDonough, 2007) in MATLAB, as described in Appendix A.

2.3 Case Studies

2.3.1 Lid Driven Cavity

For the lid driven cavity flow problem, an incompressible Newtonian fluid is bound within a 2D cavity defined by stationary walls and bottom boundary, and a top boundary moving with a constant velocity in the positive x direction. Here, the Reynolds number is defined as:

$$Re = \frac{u_{max} L}{\nu} \tag{23}$$

where L (m) is the characteristic length of the cavity, and ν (m^2 s^{-1}) is the kinematic viscosity of the fluid. In the current study both the L and ν are parameterized as 1. The boundary conditions are defined as no-slip for each side wall and the bottom wall, while the top wall velocity is defined as u_{max} (m s^{-1}). These conditions are described in Fig. 2.

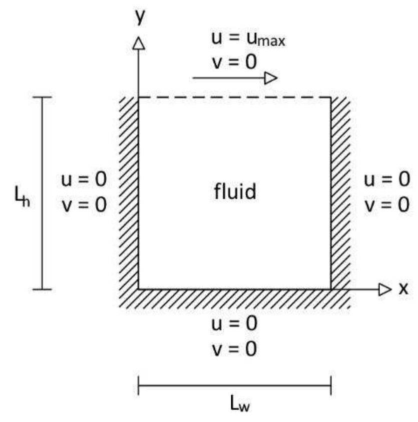

Fig. 2: Lid driven cavity flow problem definition.

For the lid driven cavity problem, convergence is defined as achieved when the difference between the updated u^{n+1} value and the old u value is less than $1(10)^{-5}$. The lid driven cavity problem is implemented in MATLAB using 50^2, 100^2, and 150^2 grid sizes, as described in Appendix A.

2.3.2 Backward Step

For the backward step problem, an incompressible Newtonian fluid flows through a 2D cavity, defined by stationary top and bottom boundary, and a fully developed parabola u-velocity profile at the inlet described by:

$$u = 4u_{max}\frac{y}{H_o}\left(1-\frac{y}{H_o}\right) \tag{24}$$

where H_o (m) is the height of the opening into the cavity, and the maximum velocity u_{max} (m s^{-1}) is defined using the height of the step H_s (m) and the Reynolds number as:

$$Re = \frac{u_{max}H_s}{\nu} \tag{25}$$

where $H_s = H - H_o$, with H the total height of the cavity. The outlet is a total length of $15H_o$ downstream of the step such that flow is considered well developed when exiting the cavity. The boundary conditions are again defined as no-slip for the top and bottom wall, as well as the inlet wall not defined by the parabolic function defined by Eq. 24. The outlet boundary is assumed to satisfy $\partial u/\partial x = \partial v/\partial x = 0$. These conditions are described in Fig. 3.

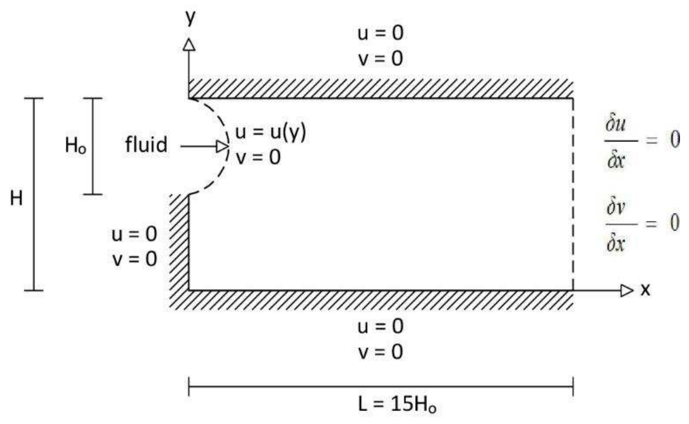
y
u = 0
v = 0
H
H$_o$
fluid
u = u(y)
v = 0
$\frac{\partial u}{\partial x} = 0$
$\frac{\partial v}{\partial x} = 0$
u = 0
v = 0
x
u = 0
v = 0
L = 15H$_o$

Fig. 3: Backward step problem definition.

For the backward step problem, convergence is defined as achieved when the difference between the updated u^{n+1} value and the old u value is less than $1(10)^{-5}$. The backward step problem is implemented in MATLAB using a 300 x 30 grid size, as described in Appendix A.

3. Results and Discussion

3.1 Lid Driven Cavity

The lid driven cavity problem is simulated in MATLAB for Reynolds numbers of 400, 1000, 5000, and 10000, with grid sizes of 50^2, 100^2, and 100^2. The results of these simulations are validated using the cavity midpoint u-velocity as compared to the experimental data of Ghia et al (1982), as shown in Fig. 4 – 7. As is shown in these figures, the pressure-velocity CFD method developed in this work to solve the lid driven cavity problem approximates the experimental data well, with an improvement in overall resolution when the grid size was reduced. However, stability loss is noticeable for $Re \geq 5000$, which is accounted for by Bruneau and Jouron (1990) as resulting from small eddies forming along the walls and in the corners of the cavity. These authors suggest that this indicates a transition to turbulent flow for $Re \geq 5000$, which is supported by the work of Thompson and Ferzinger (1989). The relaxation factors (α_u, α_p), CPU times (s) and convergence iterations required for each lid driven cavity simulation are summarized in Table 2.

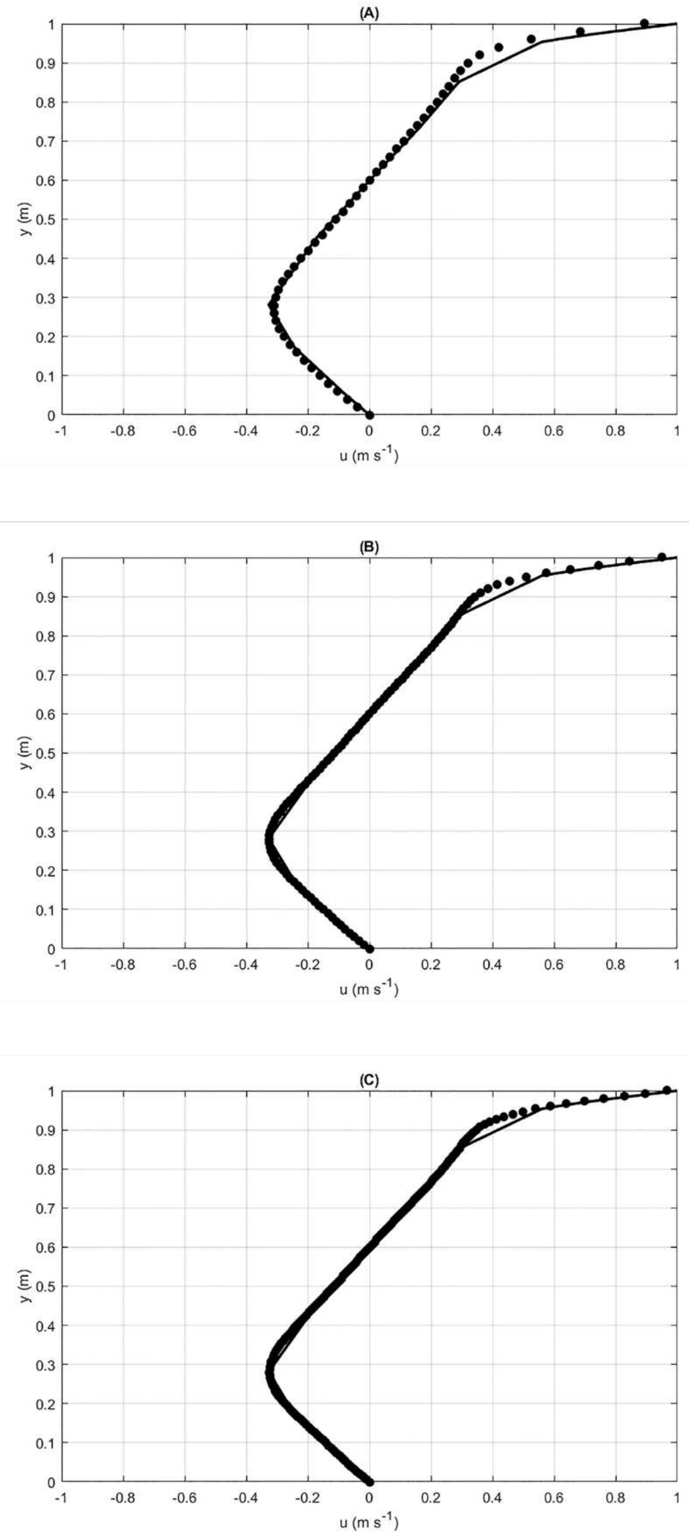

Fig. 4: MATLAB simulated values of midpoint u-velocity (solid dots) versus Ghia et al (1982) experimental data (solid line) for lid-driven cavity flow problem when Re = 400 with grid size (A) 50^2, (B) 100^2, and (C) 150^2.

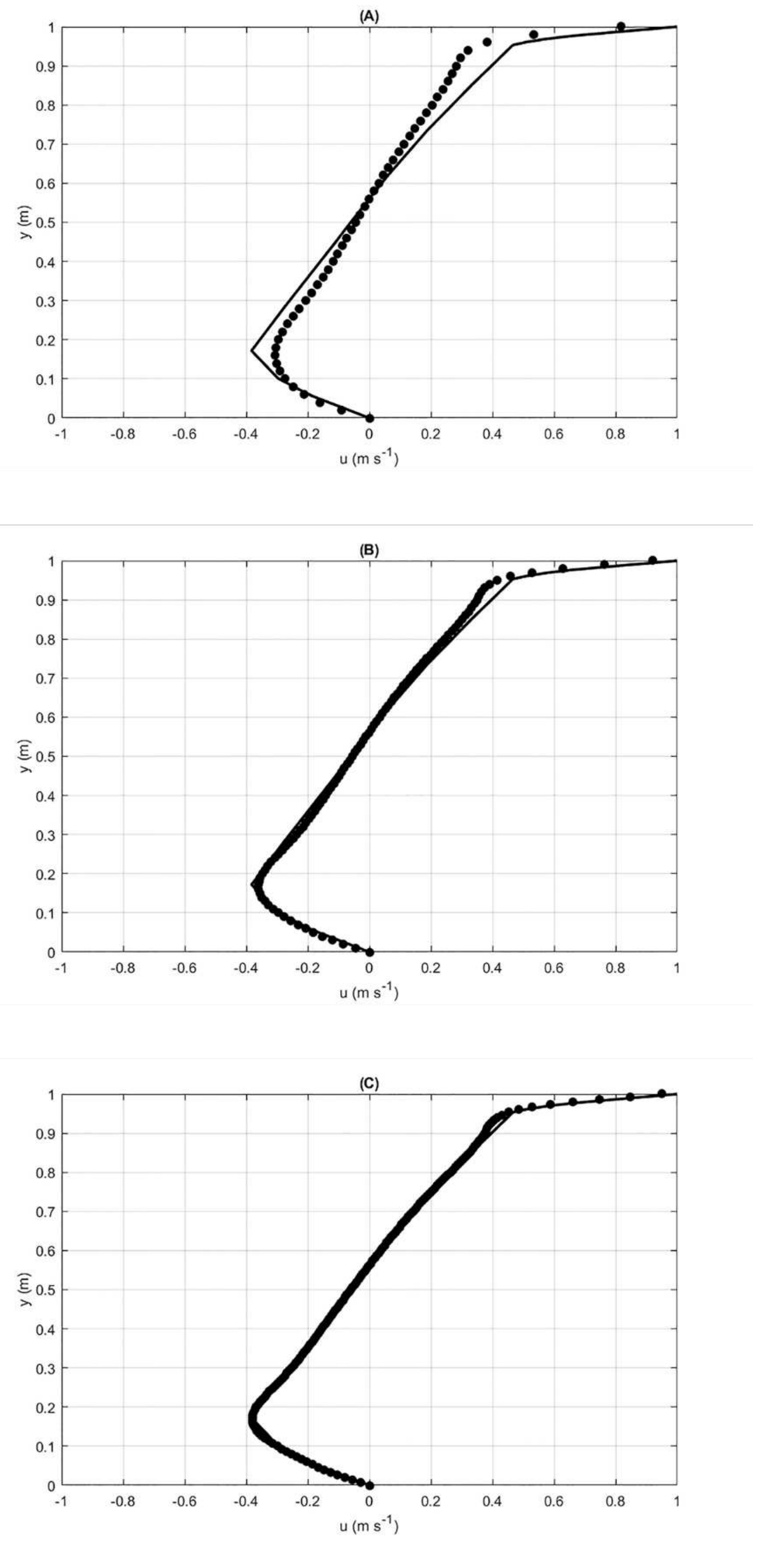

Fig. 5: MATLAB simulated values of midpoint u-velocity (solid dots) versus Ghia et al (1982) experimental data (solid line) for lid-driven cavity flow problem when Re = 1000 with grid size (A) 50^2, (B) 100^2, and (C) 150^2.

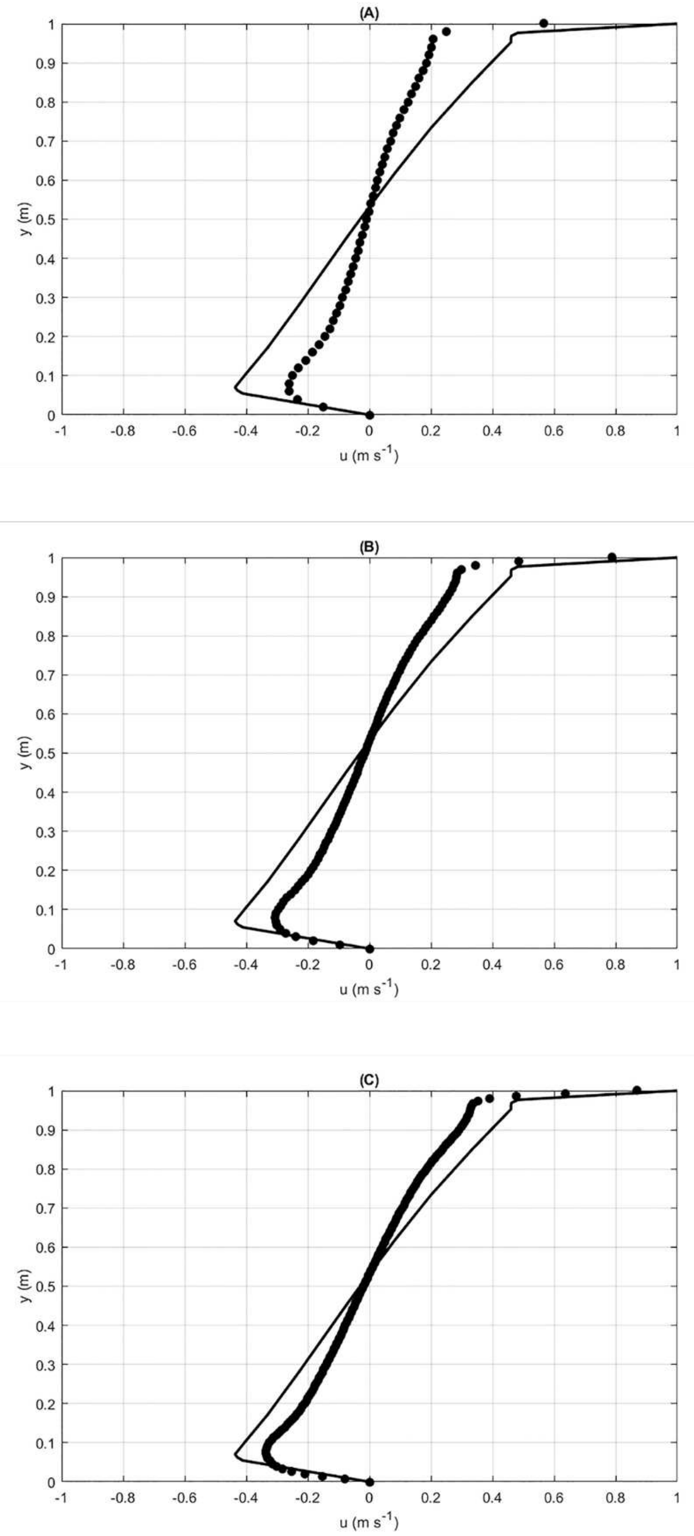

Fig. 6: MATLAB simulated values of midpoint u-velocity (solid dots) versus Ghia et al (1982) experimental data (solid line) for lid-driven cavity flow problem when Re = 5000 with grid size (A) 50^2, (B) 100^2, and (C) 150^2.

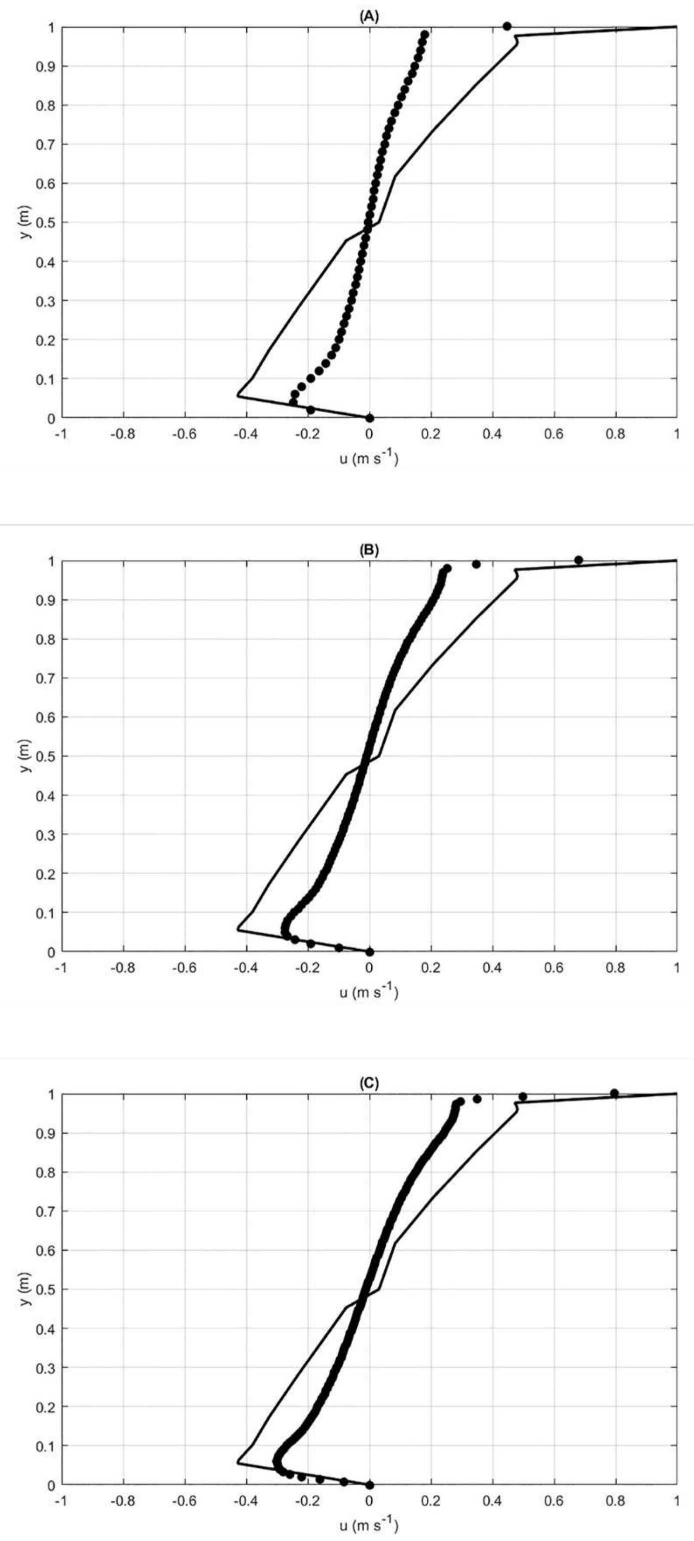

Fig. 7: MATLAB simulated values of midpoint u-velocity (solid dots) versus Ghia et al (1982) experimental data (solid line) for lid-driven cavity flow problem when Re = 10000 with grid size (A) 50^2, (B) 100^2, and (C) 150^2

Table 2: CPU time and convergence iterations for lid driven cavity problem at various Reynolds numbers, grid sizes, and velocity and pressure relaxation factors.

Reynolds Number (Re)	Grid Size	Velocity Relaxation (α_u)	Pressure Relaxation (α_p)	CPU Time (s)	Convergence Iterations (err = 10^{-5})
400	50^2	0.7	0.3	33.263	1053
	100^2			298.019	2831
	150^2			1134.856	4919
1000	50^2	0.7	0.3	56.510	1854
	100^2			542.452	3658
	150^2			1879.729	6919
5000	50^2	0.3	0.7	46.268	1552
	100^2			478.696	4594
	150^2			1978.661	7246
10000	50^2	0.3	0.7	129.491	1708
	100^2			821.138	5548
	150^2			2404.694	8893

3.2 Backward Step

The lid driven cavity problem is simulated in MATLAB for Reynolds numbers of 10, 100, 200, 300, and 500, with a grid size of 300 x 30. Streamline plots of these simulations are shown in Fig. 8, which are validated in Fig. 9 based on the work of Iwai et al (2000). The reattachment points resulting from the present MATLAB simulation correlate well with those presented in Fig. 9, although upper eddies are not visible in the present work when $Re = 500$ as would be expected (Biswas et al 2004). This can be resolved through grid refinement, and by manipulating the expansion ratio to more closely approximate those found in the literature. The relaxation factors (α_u, α_p), reattachment points (x/H_s), CPU times (s) and convergence iterations required for each backward step simulation are summarized in Table 3.

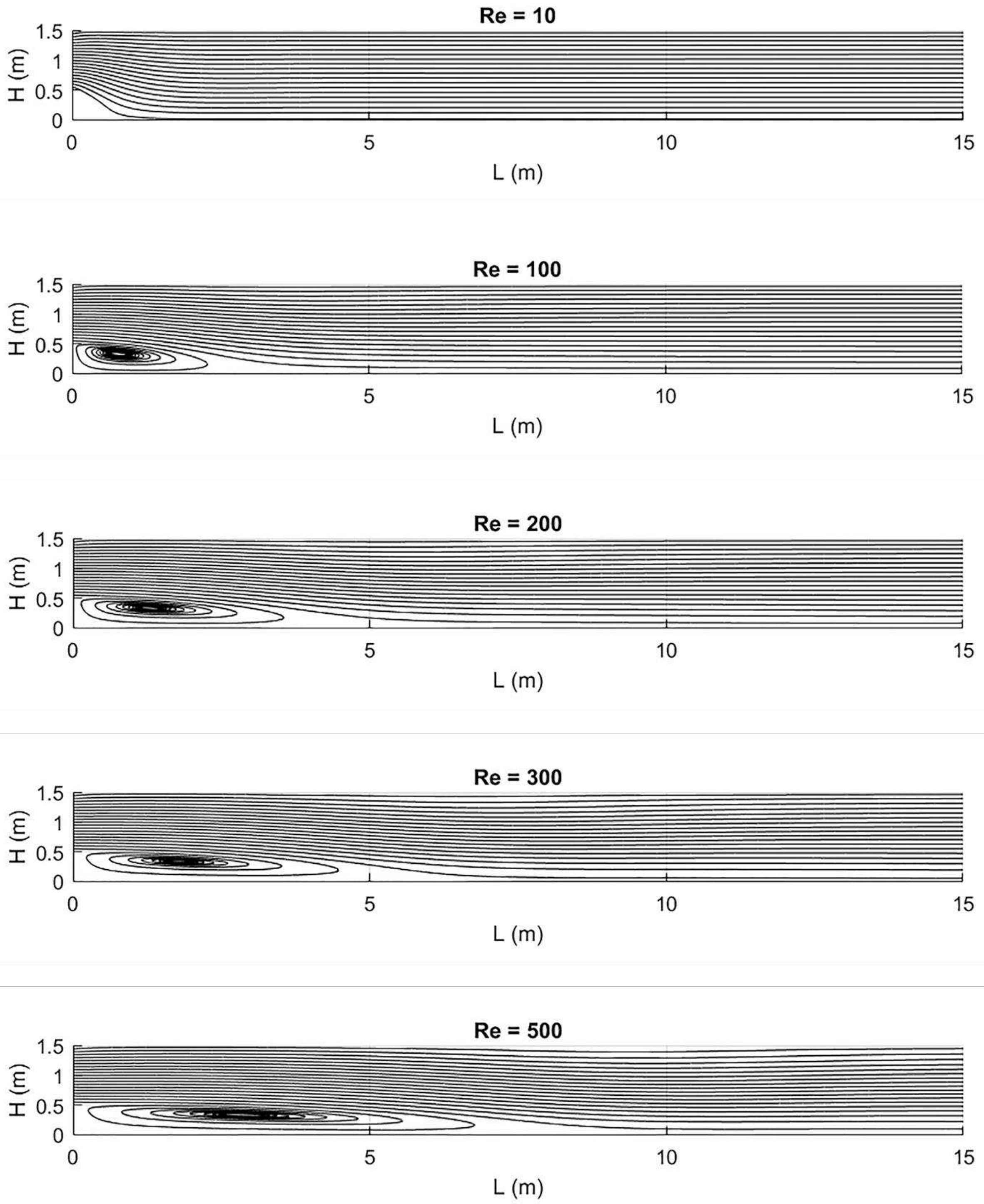

Fig. 8: MATLAB streamline plots for backward step problem using 300 x 30 grid and Re = 10, 100, 200, 300 and 500.

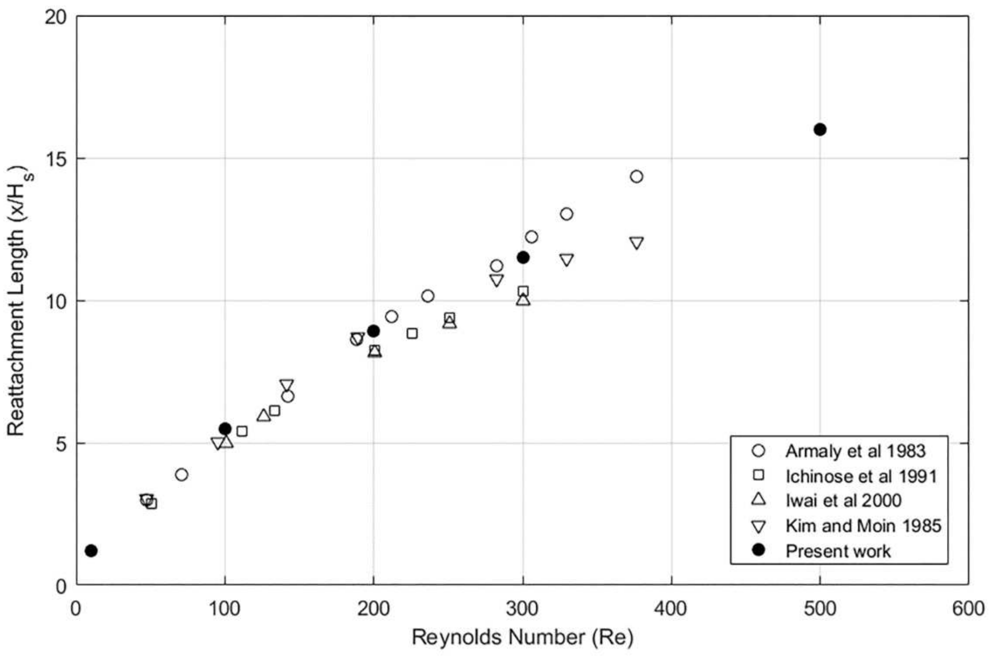

Fig. 9: Reattachment points (adapted from Iwai et al 2000).

Table 3: Reattachment points, CPU time, and convergence iterations in backward step problem at various Reynolds numbers using 300 x 30 grid.

Reynolds Number (Re)	Velocity Relaxation (α_u)	Pressure Relaxation (α_p)	Reattachment Length (x/H_s)	CPU Time (s)	Convergence Iterations (err = 10^{-5})
10	0.3	0.7	1.2	284.506	1799
100	0.3	0.7	5.5	289.498	3320
200	0.3	0.7	8.9	649.648	7330
300	0.7	0.3	11.5	316.969	3717
500	0.8	0.2	16.0	2118.869	22199

4. Conclusion

In this work, a numerical method has been developed using the Navier-Stokes equations for pressure-velocity coupling in incompressible steady-flow systems. The SIMPLE algorithm has been implemented in MATLAB and tested against two classical CFD problems, including the lid driven cavity flow problem, and the backward step flow problem. The simulations developed here show a good correlation to with those presented in the literature, indicating that the current work is a suitable method for solving PV coupled systems using CFD. However, as the SIMPLE algorithm is now over 40 years old, more current methods show far superior results than those presented here (see, for instance, Seibold, 2008). The work here is therefore best suited as an introduction to CFD using the FVM and the SIMPLE algorithm, or as a resource for coding similar problems in novel languages such as Python (Barba, 2017).

References

Barba, L. (2017). CFD Python: 12 steps to Navier-Stokes. Retrieved May 11, 2017 from http://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

Barton, I. (1994). Laminar flow past an enclosed and open backward-facing step. *Physics of Fluids, 6*(12), 4054-4056.

Bruneau, C. H., and Jouron, C. (1990). An efficient scheme for solving steady incompressible Navier-Stokes equations. *Journal of Computational Physics*, *89*(2), 389-413.

Biswas, G., Breuer, M., and Durst, F. (2004). Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers. *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF FLUIDS ENGINEERING*, *126*, 362-374.

Craft, T. (2011). Pressure-Velocity Coupling. *Lecture Notes*.

Erturk, E. (2009). Discussions on driven cavity flow. *International Journal for Numerical Methods in Fluids, 60*(3), 275-294.

Ferziger, J., and Perić, M. (2002). *Computational methods for fluid dynamics* (3rd, rev. ed.). Berlin: Springer.

Ghia, U. K. N. G., Ghia, K. N., and Shin, C. T. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics*, *48*(3), 387-411.

Iwai, H., Nakabe, K., & Suzuki, K. (2000). Flow and heat transfer characteristics of backward-facing step laminar flow in a rectangular duct. *International Journal of Heat and Mass Transfer*, *43*(3), 457-471.

Kim, D., and Choi, H. (2000). A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids. *Journal of computational physics*, *162*(2), 411-428.

McDonough, J. M. (2007). Lectures in computational fluid dynamics of incompressible flow: Mathematics, algorithms and implementations. *Departments of mechanical engineering and mathematics, University of Kentucky*.

Patankar, S. (1980). *Numerical heat transfer and fluid flow* (Series in computational methods in mechanics and thermal sciences). Washington: New York: Hemisphere Pub.; McGraw-Hill.

Recktenwald, G. (2010). The SIMPLE Algorithm for Pressure-Velocity Coupling. *Portland State University, Department of Mechanical Engineering*.

Sahin, M., and Owens, R. (2003). A novel fully implicit finite volume method applied to the lid-driven cavity problem—Part I: High Reynolds number flow calculations. *International Journal for Numerical Methods in Fluids, 42*(1), 57-77.

Seibold, B. (2008). A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains mit18086 navierstokes. m. *Massachusetts Institute of Technology*.

Thompson, M. C., and Ferziger, J. H. (1989). An adaptive multigrid technique for the incompressible Navier-Stokes equations. *Journal of computational Physics*, *82*(1), 94-121.

Togun, Safaei, Sadri, Kazi, Badarudin, Hooman, and Sadeghinezhad. (2014). Numerical simulation of laminar to turbulent nanofluid flow and heat transfer over a backward-facing step. *Applied Mathematics and Computation, 239*, 153-170.

Von Terzi, D., and Fasel, H. F. (2004). *Numerical Investigation of Transitional and Turbulent Backward -facing Step Flows.* ProQuest Dissertations and Theses.

Versteeg, H., and Malalasekera, W. (2007). *An introduction to computational fluid dynamics: The finite volume method* (2nd ed.). Harlow, England; New York: Pearson Education

Williams, P. T., and Baker, A. J. (1997). Numerical simulations of laminar flow over a 3D backward-facing step. *International Journal for Numerical Methods in Fluids*, *24*(11), 1159-1183.

Yapici, K., and Uludag, Y. (2013). Finite volume simulation of 2-D steady square lid driven cavity flow at high reynolds numbers. *Brazilian Journal of Chemical Engineering*, *30*(4), 923-937.

Nomenclature

Symbols

a	Coefficient of FVM (W K^{-1})
A	CV boundary Area (m^2)
b	Source term
e	East CV boundary (-)
E	East FVM node (-)
H	Height (m)
L	Length (m)
n	North CV boundary (-)
N	North FVM node (-)
p	Pressure (kPa)
P	Central CV node (-)
Re	Reynolds Number (-)
s	South CV boundary (-)
S	South FVM node (-)
T	Temperature (^{o}C)
u	Velocity in x-direction (m s^{-1})
v	Velocity in y-direction (m s^{-1})
V	Volume (m^3)
w	West CV boundary (-)
W	West FVM node (-)

Subscripts

h	Refers to cavity height
max	Refers to maximum velocity
o	Refers to opening height in backward step problem
s	Refers to step height in backward step problem
w	Refers to cavity width in Fig, 2

Superscripts

c	Refers to convective flux
d	Refers to diffusive flux

Greek Symbols

α	Relaxation factor (-)
ν	Kinematic viscosity ($m^2\ s^{-1}$)

Abbreviations

1D	One-dimensional
CFD	Computational fluid dynamics
CV	Control volume
FVM	Finite volume method

Appendix A: MATLAB code

```
% computational fluid dynamics
% finite volume method with SIMPLE algorithm
% aaron outhwaite 2017
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BACKWARD STEP PROBLEM
% variable declaration
nu = 2e-3;              % Kinematic viscosity (m^2/s)
H = 1.5;                % Height of cavity (m)
Ho = 1.0;               % Height of opening (m)
Hs = H-Ho;              % Height of step (m)
L = 15*Ho;              % Length of cavity (m)
Re = 500;               % Reynolds number; test Re = 10, 100, 200, 300, 500
uMax = Re*nu/Hs;        % Maximum velocity(m/s)
alphaU = 0.8;           % Velocity relaxation (under)
alphaP = 0.2;           % Pressure relaxation (under)
maxIter = 50000;        % Iteration max for SIMPLE algorithm (-)
maxNMiter = 1;          % Iteration max for numerical method (-)
err = 1e-5;             % Convergence criteria (-)
% initialize grid matrix
Nx = 150;
dx = L/Nx;
x = 0:dx:L;
Ny = 60;
dy = H/Ny;
y = 0:dy:H;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% LID DRIVEN CAVITY PROBLEM
% variable declaration
%{
nu = 1;                 % Kinematic viscosity (m^2/s)
Lw = 1;                 % Width of cavity (m)
Lh = Lw;                % Height of cavity (m)
uLid = 1.0;             % Lid velocity (m/s)
Re = 10000;             % Re = uLid*Lw/nu; test Re = 400,1000,5000,10000
alphaU = 0.3;           % Velocity relaxation (under)
alphaP = 0.7;           % Pressure relaxation (under); aP = 1 - aU
maxIter = 25000;        % Iteration max for SIMPLE algorithm (-)
maxNMiter = 1;          % Iteration max for numerical method (-)
err = 1e-5;             % Convergence criteria (-)
refSoln = xlsread('Ghia_10000.xlsx');
% initialize grid matrix
Nx = 150;
dx = Lw/Nx;
x = 0:dx:Lw;
Ny = Nx;
dy = Lh/Ny;
y = 0:dy:Lh;
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% preallocate memory for u, v, p matrices
u =         zeros(Nx+1,Ny+2);
uStar =     zeros(Nx+1,Ny+2);
uPrime =    zeros(Nx+1,Ny+2);
v =         zeros(Nx+2,Ny+1);
vStar =     zeros(Nx+2,Ny+1);
vPrime =    zeros(Nx+2,Ny+1);
p =         zeros(Nx+2,Ny+2);
pStar =     zeros(Nx+2,Ny+2);
pPrime =    zeros(Nx+2,Ny+2);
dU =        zeros(Nx+1,Ny+2);
dV =        zeros(Nx+2,Ny+1);
uLeft =     zeros(1,Ny+2);
% initial conditions
u(:,:) = 0.0;
v(:,:) = 0.0;
p(:,:) = 0.0;
uOld = u;
vOld = v;
pStar = p;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIMPLE algorithm
for n = 1:maxIter
    % STEP 1a: solve x-momentum as uStar
    % setup coefficients
    for i = 2:Nx
            for j = 2:Ny+1
                % convective flux using central differencing
                uCe = dy*0.5*(uOld(i,j)+uOld(i+1,j));
                uCw = dy*0.5*(uOld(i,j)+uOld(i-1,j));
                uCn = dx*0.5*(vOld(i,j)+vOld(i+1,j));
                uCs = dx*0.5*(vOld(i,j-1)+vOld(i+1,j-1));
                % boundary coefficients using hybrid schemes
                uDx = dy/(dx*Re);
                uDy = dx/(dy*Re);
                auE_h = [-uCe,(uDx-0.5*uCe),0];
                auW_h = [uCw,(uDx+0.5*uCw),0];
                auN_h = [-uCn,(uDy-0.5*uCn),0];
                auS_h = [uCs,(uDy+0.5*uCs),0];
                auE(i,j) = max(auE_h);
                auW(i,j) = max(auW_h);
                auN(i,j) = max(auN_h);
                auS(i,j) = max(auS_h);
                % central coefficient using applied under-relaxation
                auP(i,j) = (auE(i,j)+auW(i,j)+auN(i,j)+auS(i,j))/alphaU;
                % set u velocity component of PCE
                dU(i,j) = dy/auP(i,j);
            end
    end
    % use previous calculation as initial guess in numerical method
    uStar = uOld;
    for iter = maxNMiter
        for i = 2:Nx
            for j = 2:Ny+1
                uStar(i,j) = ...
                    (auE(i,j)*uStar(i+1,j)+ auW(i,j)*uStar(i-1,j)...
                    + auN(i,j)*uStar(i,j+1)+auS(i,j)*uStar(i,j-1)...
                    - dy*(pStar(i+1,j)-pStar(i,j))...
                    + (1-alphaU)*auP(i,j)*uOld(i,j))...
                    /auP(i,j);
            end
        end
    end

    % STEP 1b: solve y-momentum as vStar
    % setup coefficients
    for i = 2:Nx+1
            for j = 2:Ny
                % convective flux using central differencing
                vCe = dy*0.5*(uOld(i,j)+uOld(i,j+1));
                vCw = dy*0.5*(uOld(i-1,j)+uOld(i-1,j+1));
                vCn = dx*0.5*(vOld(i,j)+vOld(i,j+1));
                vCs = dx*0.5*(vOld(i,j)+vOld(i,j-1));
                % boundary coefficients using hybrid scheme
                vDx = dy/(dx*Re);
                vDy = dx/(dy*Re);
                avE_h = [-vCe,(vDx-0.5*vCe),0];
                avW_h = [vCw,(vDx+0.5*vCw),0];
                avN_h = [-vCn,(vDy-0.5*vCn),0];
                avS_h = [vCs,(vDy+0.5*vCs),0];
                avE(i,j) = max(avE_h);
                avW(i,j) = max(avW_h);
                avN(i,j) = max(avN_h);
                avS(i,j) = max(avS_h);
                % central coefficient using applied under-relaxation
                avP(i,j) = (avE(i,j)+avW(i,j)+avN(i,j)+avS(i,j))/alphaU;
                % set u velocity component of PCE
                dV(i,j) = dx/avP(i,j);
            end
    end
    % use previous calculation as initial guess in numerical method
    vStar = vOld;
    for iter = maxNMiter
        for i = 2:Nx+1
            for j = 2:Ny
                vStar(i,j) = ...
                    (avE(i,j)*vStar(i+1,j)+ avW(i,j)*vStar(i-1,j)...
                    + avN(i,j)*vStar(i,j+1)+avS(i,j)*vStar(i,j-1)...
                    - dy*(pStar(i,j+1)-pStar(i,j))...
                    + (1-alphaU)*avP(i,j)*vOld(i,j))...
                    /avP(i,j);
            end
        end
    end

    % STEP 2: solve pressure correction equation (PCE)
    % setup coefficients
    for i = 2:Nx+1
        for j = 2:Ny+1
            % boundary coefficients
            aPE(i,j) = dU(i,j)*dy;
            aPW(i,j) = dU(i-1,j)*dy;
            aPN(i,j) = dV(i,j)*dx;
            aPS(i,j) = dV(i,j-1)*dx;
            % central coefficient
            aPP(i,j) = aPE(i,j)+aPW(i,j)+aPN(i,j)+aPS(i,j);
            % RHS value
            bPP(i,j)= (uStar(i-1,j)-uStar(i,j))*dy+(vStar(i,j-1)...
            -vStar(i,j))*dx;
        end
    end
    % fix pressure to zero at bottom left cell
    if n == 1
    i = 2;
    j = 2;
    aPP(i,j) = 1.0;
    aPE(i,j) = 0.0;
    aPW(i,j) = 0.0;
    aPN(i,j) = 0.0;
    aPS(i,j) = 0.0;
    bPP(i,j) = 0.0;
    end
    pPrime(:,:) = 0;
    for iter = maxNMiter
        for i = 2:Nx+1
            for j = 2:Ny+1
                pPrime(i,j) = ...
                    (aPE(i,j)*pPrime(i+1,j)+ aPW(i,j)*pPrime(i-1,j)...
                    +aPN(i,j)*pPrime(i,j+1)+aPS(i,j)*pPrime(i,j-1)...
                    +bPP(i,j))/aPP(i,j);
            end
        end
    end

    % STEP 3: calculate velocity corrections
    % u corrections
    for i = 2:Nx
        for j = 2:Ny+1
            uPrime(i,j)=dU(i,j)*(pPrime(i,j)-pPrime(i+1,j));
        end
    end
    % v corrections
    for i = 2:Nx+1
        for j = 2:Ny
            vPrime(i,j)=dV(i,j)*(pPrime(i,j)-pPrime(i,j+1));
        end
    end

    % STEP 4: correct u, v and p
    % p corrections with under-relaxation
    for i = 2:Nx+1
        for j = 2:Ny+1
            p(i,j)=pStar(i,j)+pPrime(i,j)*alphaP;
        end
    end
    % u corrections
    for i = 2:Nx
        for j = 2:Ny+1
            u(i,j)=uStar(i,j)+uPrime(i,j);
        end
    end
    % v corrections
    for i = 2:Nx+1
        for j = 2:Ny
            v(i,j)=vStar(i,j)+vPrime(i,j);
        end
    end
    % check for convergence
    if n>10
        [cTest] = Converge_Step(u,uOld,v,vOld,Nx,Ny);
        if cTest < err
            break;
        end
    end
    Num_Iteration = n;

    % STEP 5: update cell velocities
    % update old values
    uOld = u;
    vOld = v;
    pStar = p;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % BACKWARD STEP PROBLEM
    % apply inlet velocity function
    yy = H:-dy:0;
    for j = Ny+1:-1:2
        if yy(1,j)< Ho
            uLeft(1,j) = 4*uMax*(yy(1,j)/Ho)*(1-(yy(1,j)/Ho));
            vLeft(1,j) = 0.0;
        elseif yy(1,j)>= Ho
            uLeft(1,j) = 0.0;
            vLeft(1,j) = 0.0;
        end
    end
    % apply boundary conditions
    uOld(:,1) = 0.0;                   % bottom boundary
    vOld(:,1) = 0.0;
    uOld(:,Ny+2) = 0.0;                % top boundary
    vOld(:,Ny+1) = 0.0;
    uOld(1,:) = uLeft(1,:);            % left boundary
    vOld(1,:) = vLeft(1,:);
    uOld(Nx+1,:) = uOld(Nx,:);         % right boundary
    vOld(Nx+2,:) = vOld(Nx+1,:);
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % LID DRIVEN CAVITY PROBLEM
    % apply boundary conditions
    %{
    uOld(:,1) = 0.0;                           % bottom boundary
    uOld(2:Nx,Ny+2) = 2*uLid-uOld(2:Nx,Ny+1);  % top boundary
    vOld(1,:) =  0.0;                          % left boundary
    vOld(Nx+2,:) = 0.0;                        % right boundary
    plot midpoint velocity
    if mod(n,10) == 0
       uMid = uOld(Nx/2+1,1:Ny+1);
       yMid = 0:dy:Lh;
       plot(refSoln(:,2),refSoln(:,1),uMid,yMid,'ko','MarkerFaceColor','k')
       axis ([-1 1 0 1]);
       drawnow;
    end
    %}
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BACKWARD STEP PROBLEM
% streamline plot
[xM,yM] = meshgrid(x,y);
U = uOld(1:Nx+1,1:Ny+1);
V = vOld(1:Nx+1,1:Ny+1);
sy = 0:dy:H;
sx = sy*0;
figure
axis equal;
grid on
xlim([0 L]);
ylim([0 H]);
h = streamline(xM,yM,U',V',sx,sy);
set(h,'Color','Black')
% centerline u-velocity
A = zeros(Nx+1,Ny+1);
for a = 1:10:Nx+1
    A(a,:) = U(a,:);
end
Ap = A';
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function[cmax]=Converge(u,uOld,v,vOld,Nx,Ny)

uX = u(1:Nx+1,1:Ny+1);
uXO = uOld(1:Nx+1,1:Ny+1);
vX = v(1:Nx+1,1:Ny+1);
vXO = vOld(1:Nx+1,1:Ny+1);

cmax1=max(max(uX-uXO));
cmax2=max(max(vX-vXO));
cmax = max(cmax1, cmax2);
return
end

function[cmax]=Converge(u,uOld,v,vOld)

cmax1=max(max(u-uOld));
cmax2=max(max(v-vOld));
cmax = max(cmax1, cmax2);
return
end
```