AALTO UNIVERSITY
School of Chemical Technology
CHEM-E7160 Fluid Flow in Process Units

Pressure Correction Scheme for Incompressible Fluid Flow

Ong Chin Kai
620503
Lee De Ming Benedict
620448

Abstract

In this report, pressure-correction schemes for incompressible fluid flows was studied and explored. The report focuses on two schemes, Semi-Implicit Method for Pressure Linked Equation (SIMPLE) method and Pressure-Implicit with Splitting of Operator (PISO) method. It was concluded that the PISO method is more computational-efficient when large discretized time steps were used. This is due to the implicit nature of PISO, hence not needing any iterations. However, it was found that at smaller time steps, SIMPLE methods with forward-differencing techniques are actually more computationally efficient.

Table of Contents

1. Introduction

Computational schemes for incompressible flows generally face a problem of fulfilling the continuity condition due the very properties of incompressible flow. In compressible flows, density is obtainable from the continuity equation and pressure is obtainable from the equations of state (Ferziger & Peric, 2002). However, when in an incompressible case, the continuity equation does not have a dominant variable by which we can determine the velocity field from. The continuity equation acts as a constraint rather than a dynamic equation that can be used to derive information from. One approach to deal with this problem is to utilize a pressure-based formulation to guarantee the satisfaction the continuity condition. A pressure field can be constructed to fulfil the continuity condition, and from thence a velocity field can be derived. Methods that adopt such an approach are categorized as pressure-correction methods (PCM).

A key feature of PCM is the use of a pressure-correction equation (Patankar,1980):

$$\nabla^2 p' = \frac{1}{\Delta t}(\nabla \cdot U)$$

Where p' represents pressure-correction, Δt represents a discretized time step, and U represents the velocity field.

This equation is incorporated into the algorithm, and can be solved numerically to obtain a pressure-field that satisfies the continuity condition.

There are several methods classed under PCM. This report shall mainly cover two methods which are known as Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) and Pressure Implicit with Splitting of Operator (PISO).

2. Semi-Implicit Method for Pressure Linked Equations - SIMPLE

Patankar introduced an iterative method for pressure correction for incompressible fluids **[1]**. In this report, an algorithm for implementing the SIMPLE algorithm in 2 dimensions **[2]** is presented. The momentum and continuity equations are first re-written in their dimensionless forms.

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

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

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (3)$$

where u is the horizontal velocity

v is the vertical velocity

Re is the Reynolds number

p is the pressure

The momentum equations (1) and (2) are then discretized using a staggered grid. In this staggered grid, the positions of u are at the middle of the horizontal interfaces of each cell, the positions of v are at the middle of the vertical interfaces of each cell, and the positions of p are at the center of each cell. Figure 1 below depicts the staggered grid and the positions of u, v, and p.

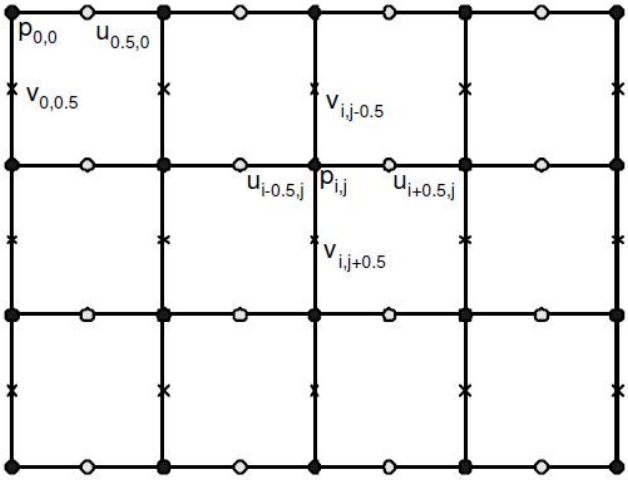

Figure 1: Staggered Grid

The momentum equations are then discretized into the following forms:

$$\frac{u_{i+0.5,j}^{n+1} - u_{i+0.5,j}^{n}}{\Delta t} + \frac{p_{i+1,j}^{n} - p_{i,j}^{n}}{\Delta x} = -a_1 + \frac{1}{Re}(a_3 + a_4) \tag{4}$$

$$a_1 = -\frac{(u^2)_{i+1.5,j}^{n} - (u^2)_{i-0.5,j}^{n}}{2.\Delta x} - \frac{(u\dot{v})_{i+0.5,j+1}^{n} - (u\dot{v})_{i+0.5,j-1}^{n}}{2.\Delta y} \tag{5}$$

$$a_3 = \frac{u_{i+1.5,j}^{n} - 2u_{i+0.5,j}^{n} + u_{i-0.5,j}^{n}}{(\Delta x)^2} \tag{6}$$

$$a_4 = \frac{u_{i+0.5,j+1}^{n} - 2u_{i+0.5,j}^{n} + u_{i+0.5,j-1}^{n}}{(\Delta y)^2} \tag{7}$$

$$\dot{v} = 0.5\left(v_{i,j+0.5} + v_{i+1,j+0.5}\right) \tag{8}$$

$$\dot{\dot{v}} = 0.5\left(v_{i,j-0.5} + v_{i+1,j-0.5}\right) \tag{9}$$

$$\frac{v_{i,j+0.5}^{n+1} - v_{i,j+0.5}^{n}}{\Delta t} + \frac{p_{i,j+1}^{n} - p_{i,j}^{n}}{\Delta y} = -b_1 + \frac{1}{Re}(b_3 + b_4) \tag{10}$$

$$b_1 = -\frac{(v^2)_{i,j+1.5}^{n} - (v^2)_{i,j-0.5}^{n}}{2.\Delta y} - \frac{(v\dot{u})_{i+1,j+0.5}^{n} - \left(v\dot{\dot{u}}\right)_{i-1,j+0.5}^{n}}{2.\Delta x} \tag{11}$$

$$b_3 = \frac{v_{i,j+1.5}^{n} - 2v_{i,j+0.5}^{n} + v_{i,j-0.5}^{n}}{(\Delta y)^2} \tag{12}$$

$$b_4 = \frac{v_{i+1,j+0.5}^{n} - 2v_{i,j+0.5}^{n} + u_{i-1,j+0.5}^{n}}{(\Delta x)^2} \tag{13}$$

$$\dot{u} = 0.5\left(u_{i-0.5,j} + u_{i-0.5,j+1}\right) \tag{14}$$

$$\dot{\dot{u}} = 0.5\left(u_{i+0.5,j} + u_{i+0.5,j+1}\right) \tag{15}$$

where u^n denotes the value of u at the n-th time step

From equations (4) – (15), it is clear that the values of u and v are approximated in a stepwise manner for every time step.

By using a guessed pressure field, p^*, equations (4) and (10) can be solved for an approximation of $(u^*)^{n+1}$ and $(v^*)^{n+1}$. It must be noted here that these approximations do not necessarily fulfil the continuity equation. In order to impose the restriction from the continuity equation the following pressure correction equation is implemented.

$$\nabla^2 p' = \frac{1}{\Delta t}(\nabla . \boldsymbol{U}) \tag{16}$$

where p' is the correction for p^* to obtain the true pressure field, p, ie, $p = p^* + p'$.

Equation (16) is a partial differential equation that belongs to a class of PDEs known as Poisson equations. For a staggered square grid, equation (16) can be solved as a system of linear equations **[3]**.

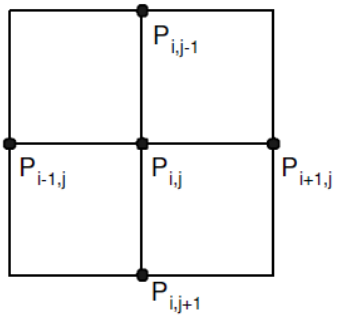

Figure 2: Grid for the Poisson Pressure Equation

$$p'_{i,j} = -\frac{1}{a}\left(b\left(p'_{i+1,j} + p'_{i-1,j}\right) + c\left(p'_{i,j+1} + p'_{i,j-1}\right) + d\right) \tag{17}$$

$$a = 2\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \tag{18}$$

$$b = -\frac{\Delta t}{\Delta x^2} \tag{19}$$

$$c = -\frac{\Delta t}{\Delta y^2} \tag{20}$$

$$d = \frac{1}{\Delta x}\left(u_{i+0.5,j} - u_{i-0.5,j}\right) + \frac{1}{\Delta y}\left(v_{i,j+0.5} - v_{i,j-0.5}\right) \tag{21}$$

Similar to pressure, velocity corrections for u and v are also implemented. The following equations are used to obtain the corrections.

$$\frac{\partial u'}{\partial t} = -\frac{\partial p'}{\partial x} \tag{22}$$

$$\frac{\partial v'}{\partial t} = -\frac{\partial p'}{\partial y} \tag{23}$$

Assuming that velocity corrections at the previous time-step are zero, equations (22) and (23) can then be discretized and rewritten as

$$u' = \frac{1}{\Delta t.\Delta x}\left(p'_{i+1,j} - p'_{i,j}\right) \tag{24}$$

$$v' = \frac{1}{\Delta t.\Delta y}\left(p'_{i,j+1} - p'_{i,j}\right) \tag{25}$$

The velocities that then satisfy the continuity equation can then be computed.

$$u^{n+1} = (u^*)^{n+1} + u' \tag{26}$$

$$v^{n+1} = (v^*)^{n+1} + v' \tag{27}$$

At this stage, convergence will be checked for. If convergence has not been achieved, the correction pressure field will be used as a new guess, equations (4) and (10) will then be solved again. The procedure is then iterated until convergence has been deemed to have been achieved.

The SIMPLE algorithm is a very mature technique that is commonly used in computational fluid dynamics. However, an obvious drawback for the method is the fact that iterations are required in the technique. It has been noted that the SIMPLE

algorithm can sometimes be computationally intensive. There have been several attempts to improve on the method such as SIMPLER, SIMPLEC.

3. Pressure-Implicit with Splitting of Operator – PISO

Another method that can be used to implement a pressure correction scheme is known as the PISO, Pressure-implicit Splitting of Operator, method **[4]**. In this method, no iterations are required. This has the benefit of being considerably less computationally intensive than the SIMPLE algorithm. Similar to the SIMPLE method, the PISO method begins with the dimensionless momentum and continuity equations.

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

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

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{3}$$

In addition, the PISO method also uses the same staggered grid that the SIMPLE method uses. Assuming that the velocity field at the n-th time step has been computed and satisfies the continuity equation, the solution at the next time step can be computed in the following steps. u^* and v^* are first calculated using equations (28) and (29). The equations are discretized first before solving for u^*.

$$\frac{u^*-u^n}{\Delta t}=-\frac{\partial (u^n)^2}{\partial x}-\frac{\partial (u^n v^n)}{\partial y} \tag{28}$$

$$\frac{v^*-v^n}{\Delta t}=-\frac{\partial (v^n)^2}{\partial y}-\frac{\partial (u^n v^n)}{\partial x} \tag{29}$$

u^{**} and v^{**} are then computed by solving the equations (30) and (31). Upon discretizing, these equations become systems of linear equations.

(30)

$$\frac{u^{**} - u^{*}}{\Delta t} = \frac{1}{Re}\left(\frac{\partial^2 u^{**}}{\partial x^2} + \frac{\partial^2 u^{**}}{\partial y^2}\right)$$

$$\frac{v^{**} - v^{*}}{\Delta t} = \frac{1}{Re}\left(\frac{\partial^2 v^{**}}{\partial x^2} + \frac{\partial^2 v^{**}}{\partial y^2}\right) \quad (31)$$

Finally, pressure correction is applied to obtain the velocity fields at the next time step.

$$\frac{u^{n+1} - u^{**}}{\Delta t} = -\frac{\partial p^{n+1}}{\partial x} \quad (32)$$

$$\frac{v^{n+1} - v^{**}}{\Delta t} = -\frac{\partial p^{n+1}}{\partial y} \quad (33)$$

In order to obtain the pressure gradients required in equations (32) and (33), another Poisson equation must be solved.

$$-\nabla^2 p^{n+1} = -\frac{1}{\Delta t}\nabla . \boldsymbol{U}^{n} \quad (34)$$

Finally, the new velocity field is obtained using equations (32) and (33). These equations can be written using vector notation as

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^{n} - \Delta t \nabla P^{n+1} \quad (34)$$

4. Lid-Driven Cavity Flow – Test Case

In order to test the effectiveness of these methods, a test case is required in order to compare the results obtained. One of the most commonly used test cases is a lid-driven cavity. As suggested in the name of the test case, the system consists of a cavity with a moving lid. The no-slip boundary conditions are then applied onto the test case. The fluid velocities in the left, bottom and right walls are all set to zero. The vertical velocity along the top wall is set to zero while the horizontal velocity is set to match the velocity of the lid. This is depicted in Figure 3.

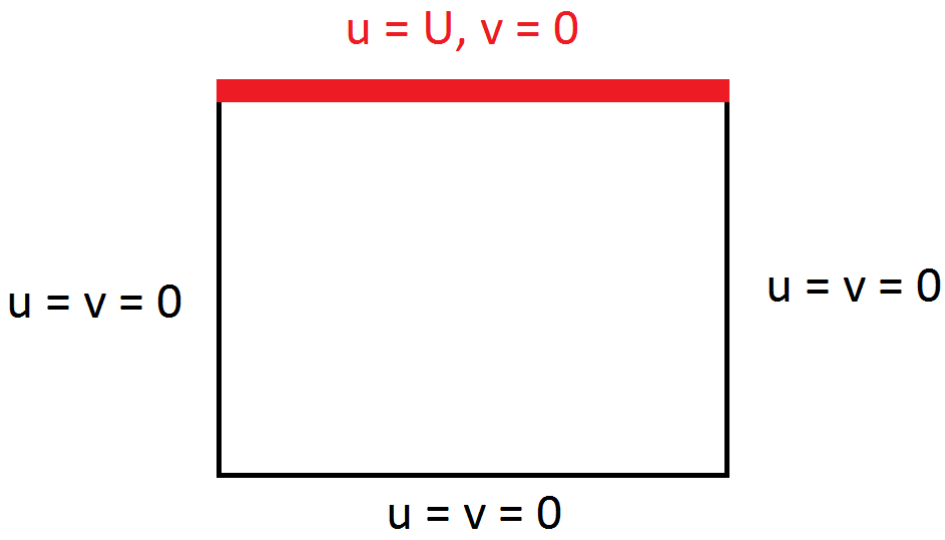

Figure 3: Diagram of the lid-driven cavity

The lid-driven cavity is so commonly used as it is a well-documented system and a lot of reference material is available to compare the results of any simulation with.

5. Implementation of pressure correction schemes in MATLAB

In this task, we attempted to implement both the SIMPLE and PISO methods in MATLAB in order to simulate the lid-driven cavity test case. Unfortunately, there were numerous hiccups along the coding process and we were unable to complete the programs for either methods before the deadline.

Instead, we present here the simulation results from a program written by Seibold **[5]** in the implementation of the PISO method, as well as an simulation from an implementation of the SIMPLE method, which was found from literature. In the program, the cavity is a 1×1 square, 90 grid points are used for both the horizontal and vertical directions, the total time length is 4 with a time step of 0.01. The Reynolds number is set to 100.

In the figures, the color represents the magnitude of the pressure field at each position, the arrows depict the velocity and the contour lines depict the stream function.

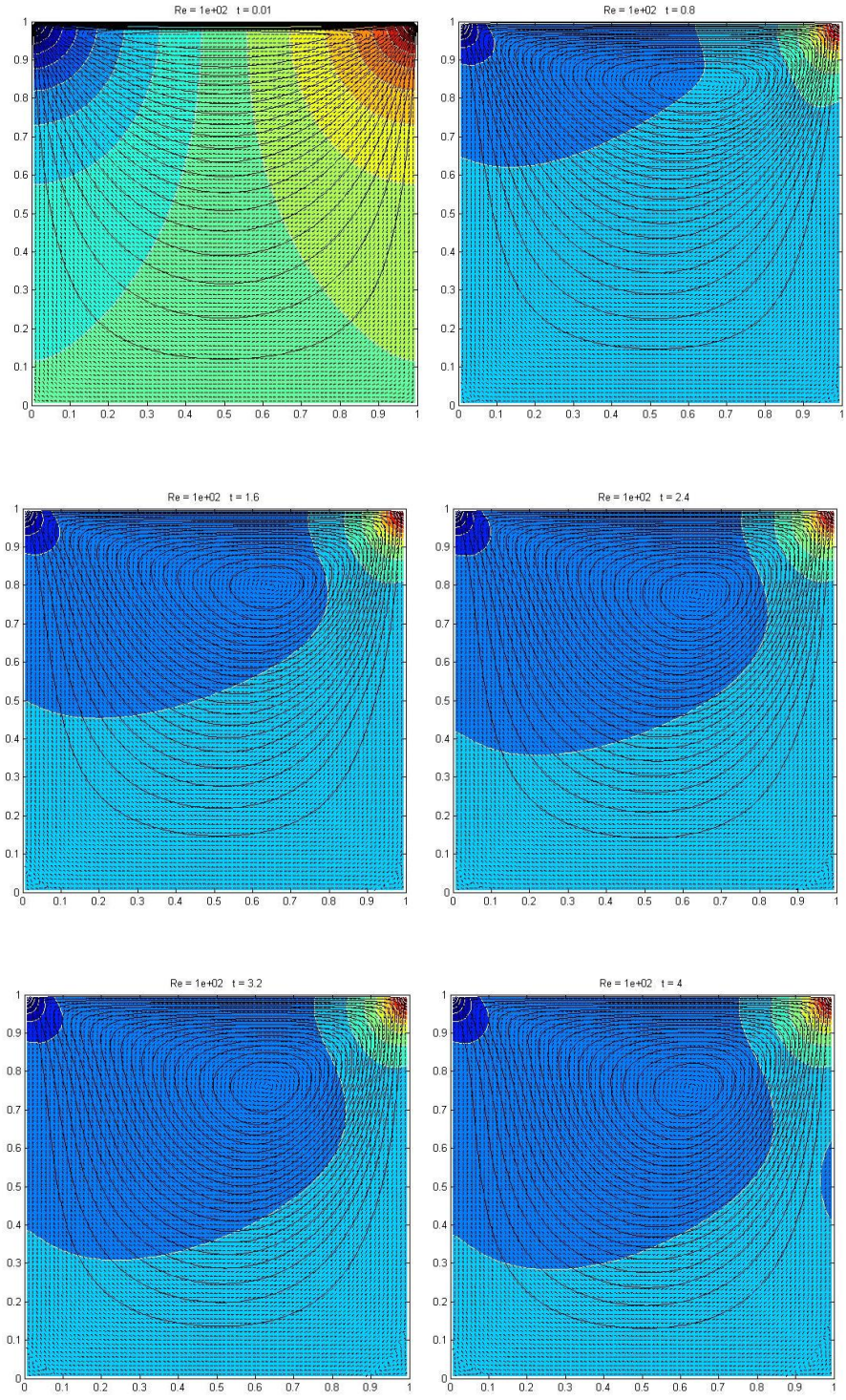

Figure 4: Simulation results from PISO program

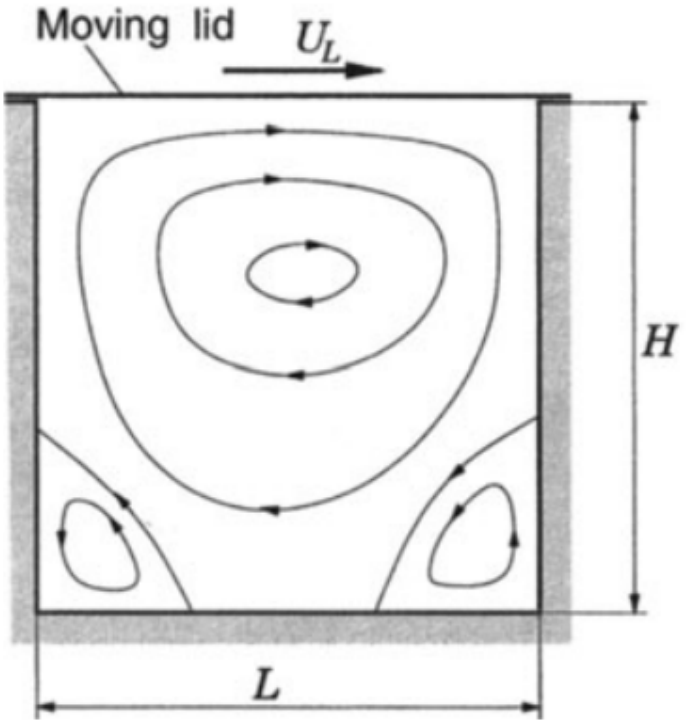

Figure 5. Simulation Results Using SIMPLE Method (Ferziger & Peric,2002)

6. Conclusion and Additional Comments

The SIMPLE method first calculates an approximate velocity field at the next time step, then uses it to calculate the pressure correction field, and from the pressure gradient obtained, iterate to obtain a corrected velocity field, hence fulfilling the continuity equation.

The PISO method calculates approximate velocity fields for the next two time steps, which creates a system of equations which can be solved implicitly. When large time steps are used, the PISO method has an advantage over the SIMPLE method as accurate results can be achieved with significantly less computational demands.

However, when small time steps are used, variants of SIMPLE schemes can perform better than PISO schemes, producing results just as accurate at a lower computational cost. Variant SIMPLE schemes using one-sided forward differencing methods are

found to perform better than the standard PISO scheme, and are easier to code as well (Barton,1998).

7. References

[1] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere, 1980

[2] A. Jafari, A. R. Haghighi, *Communications on Advanced Computational Science with Applications 2015* No.2 (2015) 72-82

[3] John D. Anderson, Jr. *Computational Fluid Dynamics: The Basics with Applications*, McGraw-Hill Inc, 1995.

[4] R. I. Issa, Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting, *Journal of Computational Physics* **62**, 40-65, 1985.

[5] B. Seibold, A Compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains. Retrieved from : http://math.mit.edu/~gs/cse/codes/mit18086_navierstokes.pdf

[6] Ferziger, J. H., & Peric, M. (2002). Computational Methods for Fluid Dynamics. Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-642-56026-2

[7] Barton, I. E. "Comparison of SIMPLE- and PISO-type Algorithms for Transient Flows." *International Journal for Numerical Methods in Fluids*. John Wiley & Sons, Ltd, 04 Dec. 1998.