Implementation of a second order projection method for viscous incompressible flow

Amirreza Hashemi[1] and Júlio Caineta[2]

[1]Computational Modeling and Simulation Program,
Department of Mechanical Engineering and Materials Science, University of Pittsburgh
[2]Computational Modeling and Simulation Program,
Department of Geology and Environmental Science, University of Pittsburgh
E-mail: {amh299, julio.caineta}@pitt.edu

December 15th 2017

.

Summary

In the final project, we plan to solve the two dimensional Navier-Stokes equations based on [1]. The method is a second-order fractional step scheme, where diffusion-convection equations are firstly solved to determine intermediate velocities, which are then projected onto the space of divergence-free vector fields. The discretization of the projection operator introduced in the original paper will be replaced with an approximate projection derived by Lai [2]. Lai's projection method is a modification that accounts for the non-zero divergence of the velocity and has a second order accuracy both in space and time. Finally, convergence and performance of our numerical implementation will be verified by applying the method to vortex spindown in a box, similarly to the test case introduced in the original paper.

1 Introduction and governing equations

In this project, we aim to reproduce the results of [1] and improve it using a projection method that has been developed by Lai [2]. The governing equations are the incompressible Navier-Stokes (1) and the continuity in non-dimensional fashion (2).

$$U_t + (U \cdot \nabla)U = Re\Delta U - \nabla p \tag{1}$$

$$\nabla \cdot U = 0 \tag{2}$$

In these equations, U is the velocity vector field, p is pressure and Re represents the non-dimensional Reynolds number. For illustration and validation, the physical problem is a vortex spindown in a cavity for two Reynolds numbers (100 and 20000). The length of the box is one in each side, and we consider solid boundary conditions for all the side walls, so that $U_d \cdot n = 0$, where d represents the direction index. The results will consist of a contour plot of vortex spindown and a performance test of the implemented numerical method.

2 Numerical methods

The numerical method is a three-step process. The first step is related to an unsplit second-order Godunov method to compute time-centered conservative differences of the nonlinear flux terms $(U\cdot\nabla)U$. The Godunov method is a predictor-corrector method and is used to calculate intermediate velocities. These predicted velocities are then used in a corrector step in which Riemann problems are solved to resolve ambiguities in the upwind direction and the resultant states are used to evaluate centered difference approximations to the advective derivatives. The second step is to solve Equation 2 using a Crank-Nicolson discretization, in which the pressure gradient and the nonlinear term (calculated in the previous time step) are considered as source terms. A discrete Hodge decomposition is the final step of the algorithm, which removes the non-divergence-free component from the solution at next step and updates the pressure.

3 Numerical discretization

Here we provide a short overview of the discretization methods for solving the two dimensional Navier-Stokes equation presented in [3]. More details about the methods of spatial and temporal discretizations are available in [1]. The discretization for the projection method is taken from the third chapter of [2], where we also refer to for more details.

3.1 Spatial discretization

The general discretization of Navier-Stokes (2) is in the form of a projected equation which is mixed with a Crank-Nicolson approximation to yield the second order discretization. The projected relation is given as

$$\frac{U^{n+1} - U^n}{\Delta t} = P\left(\frac{U^* - U^n}{\Delta t} + \nabla p^{n-\frac{1}{2}}\right). \tag{3}$$

The pressure equation is represented in the gradient component of the vector field which is projected in (3) and given in (4).

$$\nabla p^{n+\frac{1}{2}} = (I - P)\left(\frac{U^* - U^n}{\Delta t} + \nabla p^{n-\frac{1}{2}}\right). \tag{4}$$

The right hand side of both equations can be then substituted with the Crank-Nicolson approximation to increase the accuracy to second order.

$$\frac{U^{n+1} - U^n}{\Delta t} = P\left(\frac{\epsilon}{2}\Delta(U^n + U^*) - [(U \cdot \nabla)U]^{n+\frac{1}{2}}\right) \tag{5}$$

$$\nabla p^{n+\frac{1}{2}} = (I - P)\left(\frac{\epsilon}{2}\Delta(U^n + U^*) - [(U \cdot \nabla)U]^{n+\frac{1}{2}}\right) \tag{6}$$

The equations (5) and (6) are the numerically broken approximation, where the intermediate calculation will help to obtain a second order approximation throughout the numerical discretization. Here, we are providing the discretization for the terms in the RHS of the equations (5) and (6). First, the divergence term, which can be discretized at the cell centers, is given by

$$\left(D^M \tilde{U}\right)_{i,j} = \frac{1}{\Delta x}\left(\tilde{u}_{i+\frac{1}{2},j} - \tilde{u}_{i-\frac{1}{2},j}\right) + \frac{1}{\Delta y}\left(\tilde{u}_{i,j+\frac{1}{2}} - \tilde{u}_{i,j-\frac{1}{2}}\right). \tag{7}$$

The approximation of $[(U \cdot \nabla)U]^{n+\frac{1}{2}}$ also requires a discretization, which is given as an explicit Godunov method in a finite differences approximation. The discretization is given as follows:

$$\begin{aligned}(uU_x + vU_y)_{i,j} \approx &\frac{1}{2}\left(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}\right)\left(U_{i+\frac{1}{2},j} + U_{i-\frac{1}{2},j}\right) \\ &+\frac{1}{2}\left(v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}}\right)\left(U_{j,i+\frac{1}{2}} + U_{j,i-\frac{1}{2}}\right).\end{aligned} \tag{8}$$

Plugging all the spatial discretization methods into equations (5) and (6), we will get the full approximation for the RHS's. Finally, the definition of velocities in different directions follows the upwind procedure based on the Riemann problem for Burger's equation. The definitions are given as

$$\begin{aligned}u_{i+\frac{1}{2},j} &= \begin{cases} u^L & \text{if } u^L \geq 0, u^L + u^R \geq 0 \\ 0 & \text{if } u^L < 0, u^R > 0 \\ u^R & \text{otherwise} \end{cases} \\ v_{i+\frac{1}{2},j} &= \begin{cases} v^L & \text{if } u_{i+\frac{1}{2},j} > 0 \\ v^R & \text{if } u_{i+\frac{1}{2},j} < 0 \\ \frac{1}{2}(v^R + v^L) & \text{if } u_{i+\frac{1}{2},j} = 0 \end{cases}\end{aligned} \tag{9}$$

3.2 Temporal discretization

The time integration will be approximated with a Runge-Kutta fourth order method, which is a reasonable substitution based on the given advice[1], instead of the Crank-Nicolson approximation. The general form of this method is given as below.

[1]Lecture 14 of CS 294-73, given in 10/10/2017 by Dr. Colella.

$$
\begin{aligned}
y' &= f(t, y) \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\tag{10}
$$

where $k_1 = hf(t_n, y_n)$ and $k_2 = hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)$.

3.3 Discretization of the projection

The projection operator is discretized following a Helmholtz-Hodge decomposition, which decomposes any sufficiently smooth vector field in two orthogonal components, one a divergence-free vector field and the other a curl-free vector field. This means that if V is a vector field, it can be written as

$$
V = V^d + \nabla\phi \tag{11}
$$

where $\nabla \cdot V^d = 0$ (divergence-free), ϕ is a scalar field, and $\nabla \times \nabla\phi = 0$ (curl-free). Given this decomposition, we can define a projection operator $\mathbf{P}$, such that it acts on V and extracts its divergence-free term, thus projecting V onto the space of divergence-free vector fields.

In this study, instead of following the discretization of the projection operator as in [1], we implement an approximate projection operator as presented in [2], where the divergence and gradient are defined using centered differences, and the Laplacian is discretized by a standard five-point representation (in [1] it was not).

Computing the divergence of (11), we get

$$
\nabla \cdot V = \nabla \cdot V^d \nabla \cdot (\nabla\phi)\,. \tag{12}
$$

Because the first term is divergence-free, (12) can be rewritten as a Poisson equation

$$
\Delta\phi = \nabla \cdot V\,, \tag{13}
$$

which is solved for ϕ in the first step of computing the approximate projection. The divergence-free field is found by the difference

$$
V^d = V - \nabla\phi\,. \tag{14}
$$

Equation 14 corresponds to the application of the projection operator on the vector field V, P(V). The next step consists in the discretization of the operators used in equations (13) and (14). The divergence operator ($\nabla\cdot$) is discretized (D)) by centered differences:

$$
DV = \frac{v^1_{i+1,j} - v^1_{i-1,j}}{2\Delta x} + \frac{v^2_{i,j+1} - v^2_{i,j-1}}{2\Delta y}\,. \tag{15}
$$

The gradient operator (∇) is also discretized (G) using centered differences:

$$
G\phi = \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \right) \tag{16}
$$

The Laplacian (Δ) is approximated in a cell-centered five-point stencil given by

$$
(\Delta^h\phi)_{i,j} = \frac{1}{\Delta x^2}(\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}) + \frac{1}{\Delta y^2}(\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1})\,. \tag{17}
$$

3.4 Boundary conditions

In order to complete the projection, we need to set the boundary conditions for each one of the above operators. For the discrete divergence operator, the boundary conditions are the physical boundary conditions, where there is no flow at the solid-wall boundary. The discrete gradient at the boundary is determined by a second-order extrapolation of the scalar-field, which is equivalent to a first-order extrapolation of the gradients in the interior of the grid. The Poisson equation is solved using homogeneous Neumann boundary conditions, that is, we set the derivative of the field at the boundary to zero.

3.5 Filtering

The centered differences approximation used for the discrete divergence (D) and gradient (G) operators allow a non-physical oscillatory mode to persist. Although the formulation is numerically stable, it just is not physically accurate. As in [2], a filter is applied to remove this effect. This filter consists of a single point-Jacobi iteration, with a different spatial discretization.

For the point-Jacobi iteration, the aforementioned operators are denoted with a "d" (for diagonal) superscript . The diagonal divergence (D^d) is defined at cell nodes, uses variables at cell centers, and is given by

$$D^d V_{i-\frac{1}{2},2-\frac{1}{2}} = \frac{(v^1_{i+1,j} + v^1_{i-1,j}) - (v^1_{i-1,j} + v^1_{i-1,j-1})}{2\Delta x} + \frac{(v^2_{i,j} + v^2_{i-1,j}) - (v^2_{i,j-1} + v_{i-,j-1})}{2\Delta y}\,. \quad (18)$$

The corresponding gradient is also defined at cell centers, using scalars defined at the cell nodes as is given by

$$\begin{aligned}(G^d\phi)^1_{i,j} &= \left[\frac{\left(\phi_{i+1/2,j+1/2} + \phi_{i+1/2,j-1/2}\right) - \left(\phi_{i-1/2,j+1/2} + \phi_{i-1/2,j-1/2}\right)}{2\Delta x}\right] \\ (G^d\phi)^2_{i,j} &= \left[\frac{\left(\phi_{i+1/2,j+1/2} + \phi_{i-1/2,j+1/2}\right) - \left(\phi_{i+1/2,j-1/2} + \phi_{i-1/2,j-1/2}\right)}{2\Delta y}\right]\end{aligned} \quad (19)$$

Given these discretizations, to filter a vector field, first we compute $D^d V$ and initialize ϕ to zero, then relax once on V using

$$\phi := \lambda D^d V\,, \quad (20)$$

where λ is a relaxation chosen to maximize the damping in the point-Jacobi iteration for the diagonal Laplacian, e.g., if $\Delta x = \Delta y = h$, then $\lambda = -\frac{h^2}{4}$. To conclude, V is replaced by

$$V := V - G^d\phi\,. \quad (21)$$

This filter is an additional second-order accurate projection, which is based on the divergence-gradient pair D^d and G^g, and that is capable of suppressing the artifact effect mentioned above.

4 Algorithm

The algorithm can be outlined as follows:

1. Compute the advective derivatives centered at time $t^n + \frac{\Delta t}{2}$, $[(U \cdot \nabla)U]^{n+1/2}$ and $[(U \cdot \nabla)p]^{n+1/2}$, using a Godunov method.
2. Solve a discrete form of the continuity equation to obtain p^{n+1}.
3. Solve a discrete form of the momentum equation to find an intermediate velocity field, U^*.
4. Apply the projection operator to U^*, to split it into a divergence-free and a curl-free components. Use the second to update the pressure gradient, and use the first to find U^{n+1}.
5. Apply the filter to the velocity.

5 Software design

The directory structure follows the structure provided on Homework 4. `RectMDArray`, `DBox`, and `FFT` source files are kept in the *src* folder. A new folder *bchlai* has been created to keep the files related to the code developed in the course of this project. The folders which contain the utilities and executables have also been kept to be used later.

5.1 Header and source files

Below is a list of the header files, where the main classes are declared, that delineate the software design.

AdvectionSolver.h addresses the calculation of the relation (8);

DeltaVelocity.h addresses the calculation of the relation (7) (16);

DivergenceOperator.h addresses the calculation of the relation (7) (15);

PoissonSolver.h addresses the calculation of the relation (7) (13);

Projection.h addresses the calculation of the relation (7) (11);

RHSNavierStokes.h addresses the RHS of the Navier-Stokes equations where it combines the RHS terms.

6 Code development

Following the third milestone, where we introduced preliminary outline of our code algorithm, we developed on our code based on the given header files in the order below

FieldData.cpp includes functions to set initial condition, boundary condition (which is Dirichlet in our case of study), and increment as an operational function for field velocity. The implementation is consistent with Dr. Colella's lectures, with changes according to our project/test-cases.

DeltaVelocity.cpp is a calculation of the function for the gradient operator.

DivergenceOperator.cpp is a calculation of the function for the divergence operator.

PoissonSolver.cpp is the calculation of Laplacian operator, which will be used in the calculations of the RHS of velocity and projection calculations, using FFTMD.

Projection.cpp implemented based on the third chapter of Lai's thesis[2].

RHSNavierStokes.cpp handles the right hand side terms of the Navier-Stokes equation.

6.1 Tests

We present four different test routines that have helped us to develop our code.

AdvectionTest.cpp is the test of the advection term. We defined our initial problem based on a spin-down vortex, which discerption will be given in results section. The output is given in the files Advection.vtk and velocity.vtk, which are the results of Advection operator and the initial velocity field, respectively. We have tested this operator for many other initial condition to assure the solution is promising. Our conclusion was that this function works properly, therefore calculation of FieldData.cpp, DeltaVelocity.cpp, and AdvectionOperator.cpp are likely without any error.

BCTest is the test to make sure the boundary conditions are set to be zero, e.g., the Dirichlet boundary conditions are fully imposed. This test creates the file velocity.vtk, where the results show that the Dirichlet boundary conditions have been set correctly for both velocity components.

BCHLaiTest.cpp is the main and final test of our code. We defined our problem based on a spin-down vortex[4]. The outputs are the vorticity and the velocity which are dumped out in each time step.

7 Results and discussion

The results are based on a spin-down vortex problem that has been introduced in [4]. The spin-down is a gaussian distribution function defined as

$$\omega = \frac{1}{\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{22}$$

where $x_0 = 0.4$, $y_0 = 0.5$, $\sigma = 0.04$, and the given Reynolds number is over 40,000. The initial conditions are set in BCHLaiTest.cpp, and they can be changed easily. Based on our understanding, the spin-down vortex problem requires high resolution grid definition, which lead to a challenging problem for us to run on our local computers. The maximum grid size that we were able to run was 256 by 256, where the vortex move towards the corner of the box as it breaks apart and collapses. At the time that the collapse of two separated vorticies merge, a large gradient of velocity impose a diverging error to our calculation.

The break down in our simulation happens in a relatively short period of time, and we believe that by increasing the grid size, the solution would be properly handled. Another possible way of solving this problem, could be to use a different FFT algorithm, to bring the computational cost down in such a way that we would be able to run a larger test case.

Here are the results of the spin-down vortex, at the initial point and just before break down presented.

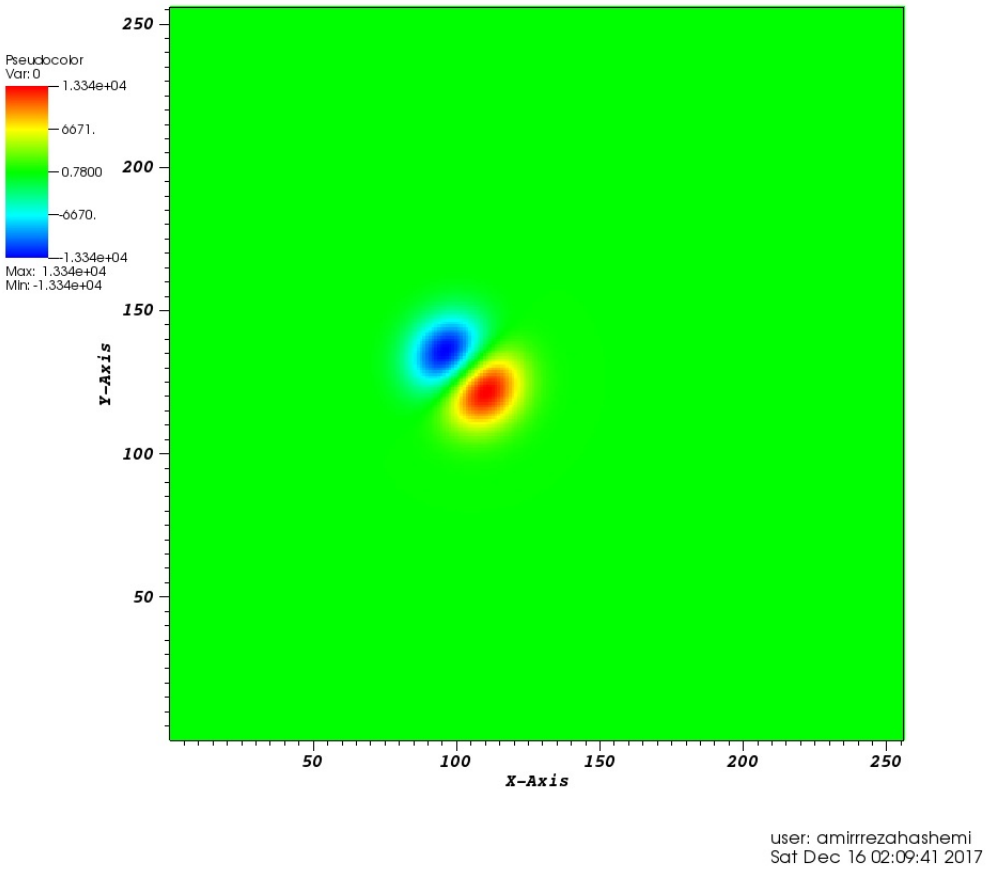

Figure 1: Vorticity contour at the initial condition.

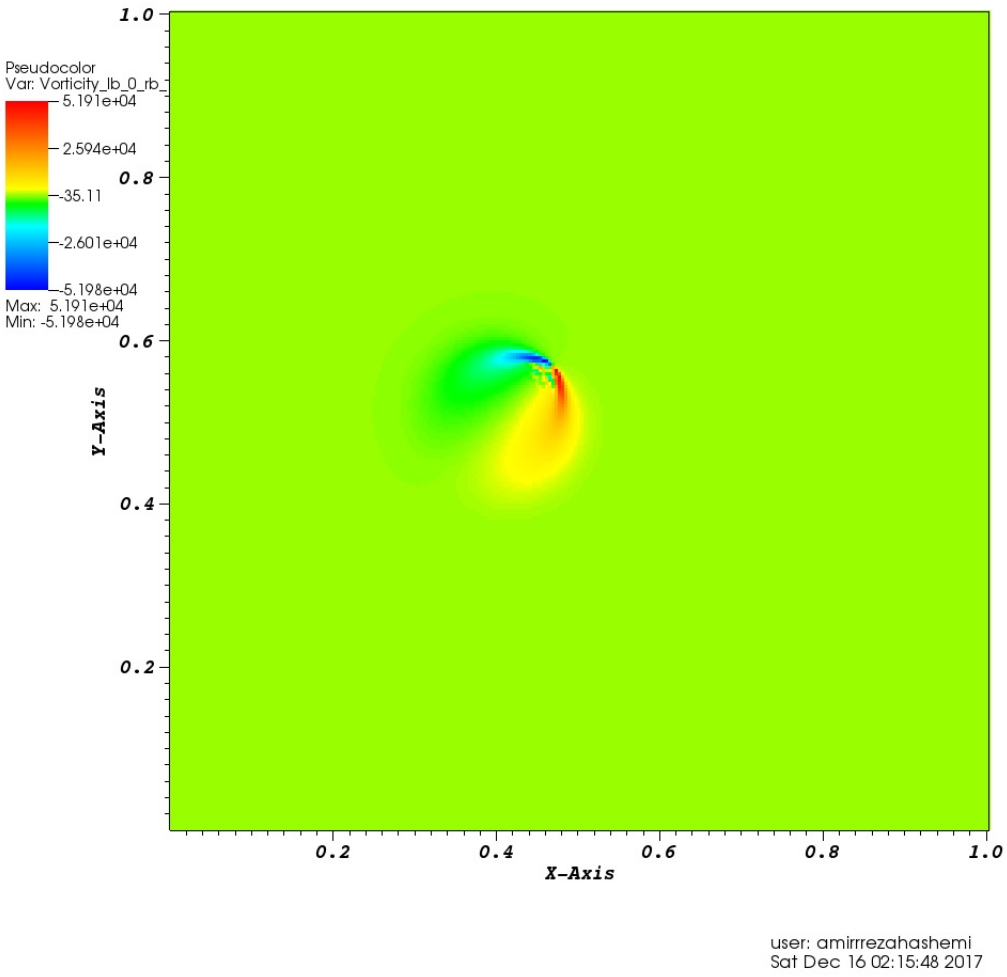

Figure 2: Vorticity contour before the break down.

8 Instruction to run

The executive files for test cases and makefile are located in /Code/exec/. The following commands will produce the outcome for each test

AdvectionTest.cpp ./AdvectionTest.exe

BCTest ./BCTest.exe

BCHLaiTest.cpp ./BCHLaiTest.exe

The outcome will be in the form of .vtk files as they have been explained earlier, and can be viewed with VisIt.

References

[1] J. Bell, L. Howell, and P. Colella, An efficient second-order projection method for viscous incompressible flow, in *10th Computational Fluid Dynamics Conference*, p. 24, Reston, Virginia, 1991, American Institute of Aeronautics and Astronautics.

[2] M. F. Lai, *A Projection Method for Reacting Flow in the Zero Mach Number Limit*, Ph.d. thesis, University of California at Berkeley, 1993.

[3] J. B. Bell, P. Colella, and H. M. Glaz, *Journal of Computational Physics* **85** (1989).

[4] L. Howell and J. Bell, An adaptive mesh projection method for viscous incompressible flow, in *SIAM Journal on Scientific Computing*, p. 18, Reston, Virginia, 1997, American Institute of Aeronautics and Astronautics.