Programming Assignment #3 : Solving Incompressible Energy Equation

Jerin Roberts
December 19, 2016

Supervisor: Dr. Carl Ollivier-Gooch
Locations: University of British Columbia

Contents

List of Figures

List of Tables

1. Introduction

The are many physical phenomenons in physics and engineering that require linear and non-linear partial differential equations to described the true nature of the system. Solving these systems analytically and finding exact solutions for these equations can be difficult and often require simplifications that ultimately don't fully represent the problem being investigated. Numerical methods for solving PDE's provides a means for finding approximations to the exact solutions without having to make sacrificial simplifications. With recent advancements in computational technology numerical methods can now be easily applied to large and difficult problems that would otherwise be impossible to solve.

1.1. Problem Overview

Numerical problems are essentially solved by breaking the entire solution domain into small discrete points (mesh) and finding the solution at or around these areas. Each point requires the solving the differential equations that represent physical phenomenon being investigated. Since the exact solution cannot be computed, it is instead approximated using various techniques and methods. In this assignment the 2D in-compressible laminar energy equation is applied to a rectangular channel for a given velocity field. The problem will employ an second ordered centered flux calculation with both implicit and explicit euler timer advance methods. The implicit method will be solved using approximate factorization and Gauss-Jordan elimination via the Thomas algorithm. Boundary conditions will be implement using ghost cells allowing the interior scheme to remain the same during calculation of bordering cells. The ghost cells are calculated such the boundary condition is enforced at $i = 1/2$.

2. Implementation

2.1. Program Overview

The C/C++language was selected for this programming assignment. The scripts were compiled using g++/gcc version 5.4.0 on Ubuntu 16.04.02 and are available in the attached zip or for clone via the link provided: `https://github.com/j16out/cfd510` . The program itself is broken into 3 pieces and two levels to produce a modular set that makes it easier to apply to different problems. The highest level contains the "macro" or the main function which can be modified for different problems. The numerical directory contains numerical.cpp script and its header file numerical.hpp. This set contains all the functions necessary for solving the problem numerically. The appendix contains the .hpp file which lists all functions with a short description of each. The vroot directory contains the scripts necessary for drawing data. These scripts make use of the ROOT-v6 libraries. ROOT is a popular data analysis framework primarily written in C++ and Phython. For more information on ROOT libraries visit the link provided: `https://root.cern.ch/` . The functions act on a structure called $carray$ which contains the solution domain array and its various parameters. The struct contains the main array, its defined working area (mesh size), data storage vectors, iteration count, and the represented dimension between points. Having these organized in a struct provides a compact way of passing and modifying the array and all its pertinent parameters. The outline of the struct used for the energy equation problem is shown below.

```
struct carray{
//arrays
double f1 [maxx][maxy];//flux or temp space
double T1 [maxx][maxy];//Temperature
double v1 [maxx][maxy];//x velocity
double u1 [maxx][maxy];//y velocity
//array attributes
int sizex = maxx;
int sizey = maxy;
double DIMx = 0.0;
double DIMy = 0.0;
//time
double ctime = 0;
};
```

2.2. Solution Method

There are three general routes that can be taken to solve the Navier-Stokes equations each of which is characterized by the way it satisfies the continuity equation; Stream function-vorticity methods take and express the velocity field in terms of derivatives of the stream function. Because of this the momentum equations are replaced by a single vorticity transport equation, while a Poisson equation relates the stream function and vorticity. This method works fine for 2D flows, but can be difficult to use in three dimensions as stream function is a 2D entity. The 2nd type is Pressure Poisson methods, which take the divergence of the momentum equations and manipulate this to get a Poisson equation for the pressure. This equation can be simplified by using the continuity equation to eliminate terms. These methods are well-known in the field, but have a reputation for being difficult to program. The method selected for this project is the artificial compressibility method. This method adds a physical time derivative of pressure to the continuity equation. This derivative is scaled by a parameter β that effectively sets the pseudo-compressibility of the fluid.The addition of a time derivative of pressure enables the coupling of the continuity equation with the momentum equations which ideally allows us to advance pressure and velocity in time together.

The non-dimensional Navier-Stokes equations in artificial compressibility form can be written as equations 2.1 to 2.3

$$\frac{\partial P}{\partial t}+\frac{1}{\beta}\frac{\partial u}{\partial x}+\frac{1}{\beta}\frac{\partial v}{\partial y}=0 \tag{2.1}$$

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial vu}{\partial y}=-\frac{\partial P}{\partial x}+\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{2.2}$$

$$\frac{\partial v}{\partial t}+\frac{\partial vu}{\partial x}+\frac{\partial v^2}{\partial y}=-\frac{\partial P}{\partial y}+\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \tag{2.3}$$

We can take this and write it in our standard form (eq 2.4 which enables to integrate over the computational cell by applying gauss's Theorem and dividing over the size of the finite volume.

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=0 \tag{2.4}$$

where

$$U=\begin{bmatrix}P\\u\\v\end{bmatrix} \tag{2.5}$$

$$F=\begin{bmatrix}\frac{u}{\beta}\\u^2+P-\frac{1}{Re}\frac{\partial u}{\partial x}\\uv-\frac{1}{Re}\frac{\partial v}{\partial x}\end{bmatrix} \tag{2.6}$$

$$G=\begin{bmatrix}\frac{v}{\beta}\\uv-\frac{1}{Re}\frac{\partial u}{\partial y}\\v^2+P-\frac{1}{Re}\frac{\partial v}{\partial y}\end{bmatrix} \tag{2.7}$$

Which leads to navier stokes in finite volume form as shown in equation 2.8

$$\frac{dU_{ij}}{dt}=-\frac{F_{i+\frac{1}{2},j}-F_{i-\frac{1}{2},j}}{\triangle x}-\frac{G_{i,j+\frac{1}{2}}-G_{i,j-\frac{1}{2}}}{\triangle y} \tag{2.8}$$

Where the right hand side (RHS) represents the 2nd order approximation to the flux integral, where U, G, and F as before are represented as 3x1 matrices of Pressure and velocity components.

3. Validation

3.1. Flux Computation

The program was built and tested in small components to ensure its correctness. The flux integral was run and compared to the exact solution for the flux integral. The problem will employ a second ordered centered flux calculation which is found on the RHS of the discretized energy equation.

The flux was calculated using the function $update\_flux()$, which is shown below. The function calls two other functions which grab the necessary data using the $surr$ struct from the current solution. The flux calculated using $calc\_newcell()$ is then stored on the $carray$ struct under the $f1$ array.

```
void update_flux(carray & myarray){
surr mysurr;
vec ftemp;
for(int j = 1; j < myarray.sizey-1; ++j)
{
    for(int i = 1; i < myarray.sizex-1; ++i)
    {
    //----get surrounding cells and compute new cell----//
    get_nsurcells(myarray, i, j, mysurr);
    calc_flux(myarray, mysurr, ftemp);

    //-----update current cell----//
    myarray.f1[i][j] = ftemp;
    }
}
}
```

The correctness of the flux integral was validated using the exact computed flux. The L_2 norms for the flux calculation are displayed in table 3.2. The order of accuracy for the scheme was determined using the log log method as displayed in figure **??**. The order of accuracy for this scheme was estimated to be 2nd order accurate based on the L_2 data from 8 different meshes.

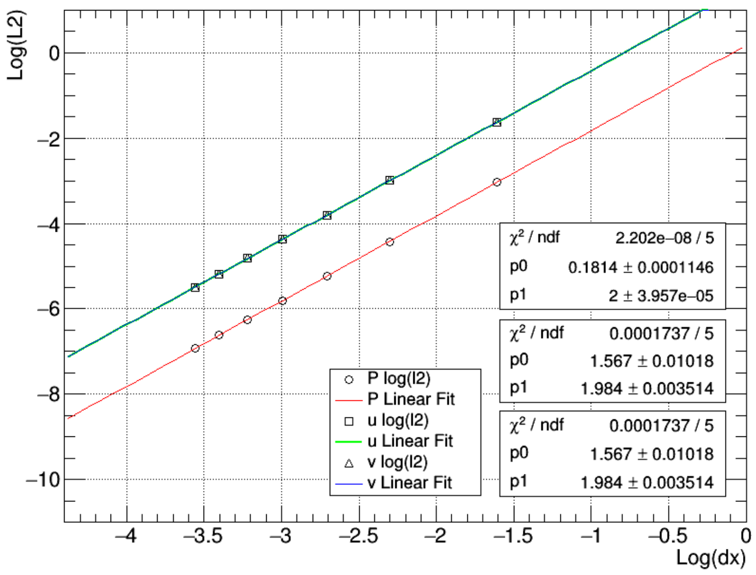


Figure 3.1: Order of Accuracy of Flux Integral for Pressure and Velocity components

Mesh Size	**Pressure L^2norm**	**Velocity L^2norm**	$\triangle x$	**Order**
10 x 10	$4.7946 * 10^{-2}$	$1.9505 * 10^{-1}$	0.1	-
20 x 20	$1.1983 * 10^{-2}$	$5.0025 * 10^{-2}$	0.05	1.952
40 x 40	$2.9957 * 10^{-3}$	$1.2585 * 10^{-2}$	0.025	1.980
80 x 80	$7.4892 * 10^{-4}$	$3.1514 * 10^{-3}$	0.0125	2.002

Table 3.1: Table of L^2 norms for increasing mesh size of flux integral

This matches what was expect by implementing a 2nd order interior flux evaluation and boundary schemes. It should be noted as seen in figure **??** the relative error decreases when refining the mesh. This is expect as the numerical solution is a discrete representation of the continuous exact solution.

3.2. Pressure Oscillations

It was found that steady-state pressure distributions have some oscillations. These were present because of decoupling between pressure in alternate lines of the mesh. To fix this problem a term was added to the right-hand side of the pressure equation which is displayed as 3.1. This essentially introduces a artificial viscosity component which for a properly selected A can smooth out the pressure fluctuations while maintaining 2nd order accuracy.

$$A\left(\frac{\overline{P}_{i+1,j} - 2\overline{P}_{i,j} + \overline{P}_{i-1,j}}{\triangle x^2} + \frac{\overline{P}_{i,j+1} - 2\overline{P}_{i,j} + \overline{P}_{i,j-1}}{\triangle y^2}\right)\triangle x \triangle y \tag{3.1}$$

3.3. Flux Jacobian

For the time discetization of the navier stokes equations a implicit euler method was selected. The fully implicit time discretization of NS equations can be written as

$$\begin{aligned} \delta U_{ij} + \triangle t A_x \delta U_{i-1,j} + \triangle t B_x \delta U_{i,j} + \triangle t C_x \delta U_{i+1,j} \\ + \triangle t A_y \delta U_{i,j-1} + \triangle t B_y \delta U_{i,j} + \triangle t A_y \delta U_{i,j+1} \\ = -\triangle t \frac{F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}}{\triangle x} - \triangle t \frac{G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}}}{\triangle y} \end{aligned} \tag{3.2}$$

The implementation of the left hand side (LHS) can be validated by implementing a very small known change in solution for one cell between time steps. The difference between the flux calculated on the RHS from time level n to n+1 should be roughly equivalent to the LHS of the implementation (minus δU_{ij}).

$$\begin{aligned}&\left(\frac{F_{i+\frac{1}{2},j}-F_{i-\frac{1}{2},j}}{\triangle x}-\frac{G_{i,j+\frac{1}{2}}-G_{i,j-\frac{1}{2}}}{\triangle y}\right)^{n+1}\\ -&\left(\frac{F_{i+\frac{1}{2},j}-F_{i-\frac{1}{2},j}}{\triangle x}-\frac{G_{i,j+\frac{1}{2}}-G_{i,j-\frac{1}{2}}}{\triangle y}\right)^{n}\\ &=A_x\delta U_{i-1,j}+B_x\delta U_{i,j}+C_x\delta U_{i+1,j}\\ &+A_y\delta U_{i,j-1}+B_y\delta U_{i,j}+A_y\delta U_{i,j+1}\end{aligned}\tag{3.3}$$

Since we've only selected one cell to change only 5 cells in total will see a calculated change which corresponds each to one term on the LHS being non-zero. This provides a great method for narrowing down which term was incorrect. If the error was found to be greater then 10^{-10} then the term was considered to be incorrect. The errors for each corresponding component of change are tabulated in table **??**.

Component (i, j)	**Pressure Error**	**u Error**	**v Error**
Ax (11, 10)	$3.000 * 10^{-15}$	$5.001 * 10^{-12}$	$5.001 * 10^{-12}$
Cx (9, 10)	$1.000 * 10^{-15}$	$5.002 * 10^{-12}$	$5.001 * 10^{-12}$
Bx By (10, 10)	$1.000 * 10^{-15}$	$2.500 * 10^{-13}$	$2.500 * 10^{-13}$
Cy (10, 9)	$1.000 * 10^{-15}$	$5.001 * 10^{-12}$	$5.000 * 10^{-12}$
Ay (10, 11)	$3.000 * 10^{-15}$	$4.995 * 10^{-12}$	$4.995 * 10^{-12}$

Table 3.2: Table of L^2 norms for increasing mesh size of flux integral

3.4. Block Thomas

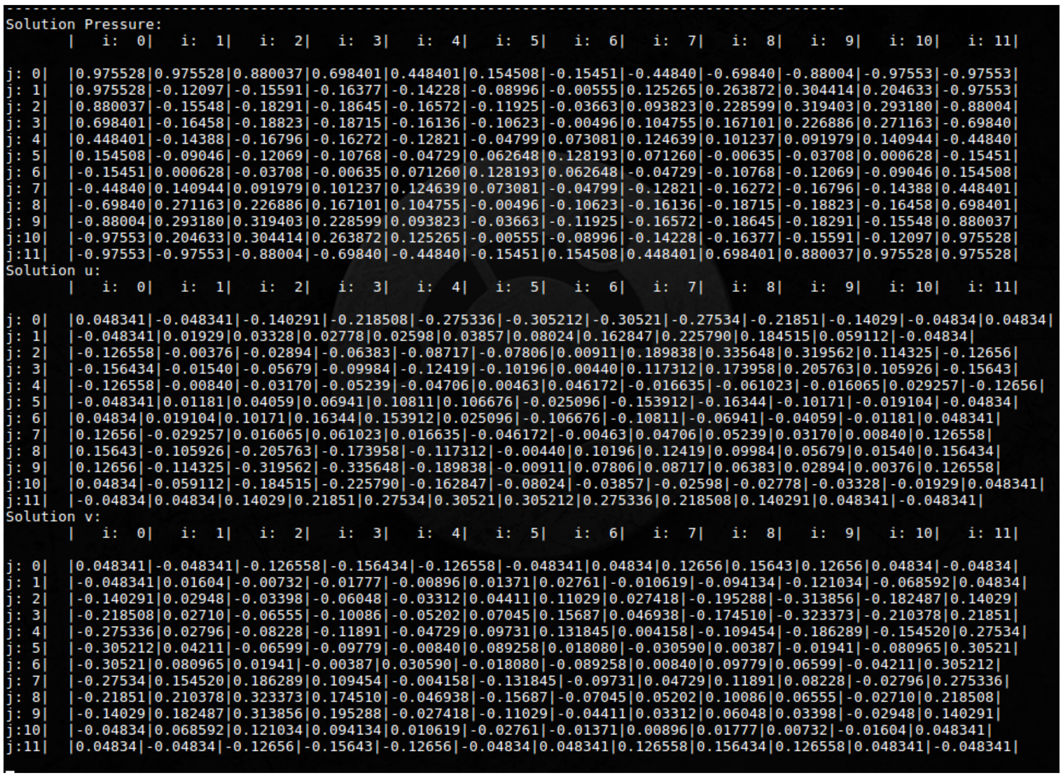
```
void solve_LinSys(carray & myarray, double tstep, double & mdiff)
{
//--linear system num 1---//
crow myrow;
for(int j = 1; j < myarray.sizey-1; ++j)
{
load_row(myarray, myrow, j, tstep);
solve_block_thomas(myarray, myrow, myarray.sizex, j);
}
ccol mycol;
for(int i = 1; i < myarray.sizex-1; ++i)
{
load_col(myarray, mycol, i, tstep);
solve_block_thomas(myarray, mycol, myarray.sizey, i);
}
}
```

4. Conclusion

This project provides great insight into the internal algorithms used for calculating numerical solutions for time varying problems. The Energy Equation applied to a steady state channel provided a great platform for developing and testing the program. Having the analytic solution really help fine tune the program and helped given an idea on how accurate the solutions could be. Investigating the error and how it changes relative to boundaries and mesh sizes will provides insight on how the methods should be applied to bigger problems to minimize time and error. Stability analysis provided a great means for understanding the computational limits of the program in terms of speed. Numerical methods provide a great way of solving difficult problems and until new methods are discovered for finding exact solution will remain the main method for finding solutions to difficult problems.

A. Appendix

A.1. Thomas Block Data Last Pass



```
------------------------------------------------------------------------------------------------------
Solution Pressure:
        |   i:  0|   i:  1|   i:  2|   i:  3|   i:  4|   i:  5|   i:  6|   i:  7|   i:  8|   i:  9|   i: 10|   i: 11|

j: 0|   |0.975528|0.975528|0.880037|0.698401|0.448401|0.154508|-0.15451|-0.44840|-0.69840|-0.88004|-0.97553|-0.97553|
j: 1|   |0.975528|-0.12097|-0.15591|-0.16377|-0.14228|-0.08996|-0.00555|0.125265|0.263872|0.304414|0.204633|-0.97553|
j: 2|   |0.880037|-0.15548|-0.18291|-0.18645|-0.16572|-0.11925|-0.03663|0.093823|0.228599|0.319403|0.293180|-0.88004|
j: 3|   |0.698401|-0.16458|-0.18823|-0.18715|-0.16136|-0.10623|-0.00496|0.104755|0.167101|0.226886|0.271163|-0.69840|
j: 4|   |0.448401|-0.14388|-0.16796|-0.16272|-0.12821|-0.04799|0.073081|0.124639|0.101237|0.091979|0.140944|-0.44840|
j: 5|   |0.154508|-0.09046|-0.12069|-0.10768|-0.04729|0.062648|0.128193|0.071260|-0.00635|-0.03708|0.000628|-0.15451|
j: 6|   |-0.15451|0.000628|-0.03708|-0.00635|0.071260|0.128193|0.062648|-0.04729|-0.10768|-0.12069|-0.09046|0.154508|
j: 7|   |-0.44840|0.140944|0.091979|0.101237|0.124639|0.073081|-0.04799|-0.12821|-0.16272|-0.16796|-0.14388|0.448401|
j: 8|   |-0.69840|0.271163|0.226886|0.167101|0.104755|-0.00496|-0.10623|-0.16136|-0.18715|-0.18823|-0.16458|0.698401|
j: 9|   |-0.88004|0.293180|0.319403|0.228599|0.093823|-0.03663|-0.11925|-0.16572|-0.18645|-0.18291|-0.15548|0.880037|
j:10|   |-0.97553|0.204633|0.304414|0.263872|0.125265|-0.00555|-0.08996|-0.14228|-0.16377|-0.15591|-0.12097|0.975528|
j:11|   |-0.97553|-0.97553|-0.88004|-0.69840|-0.44840|-0.15451|0.154508|0.448401|0.698401|0.880037|0.975528|0.975528|
Solution u:
        |   i:  0|   i:  1|   i:  2|   i:  3|   i:  4|   i:  5|   i:  6|   i:  7|   i:  8|   i:  9|   i: 10|   i: 11|

j: 0|   |0.048341|-0.048341|-0.140291|-0.218508|-0.275336|-0.305212|-0.30521|-0.27534|-0.21851|-0.14029|-0.04834|0.04834|
j: 1|   |-0.048341|0.01929|0.03328|0.02778|0.02598|0.03857|0.08024|0.162847|0.225790|0.184515|0.059112|-0.04834|
j: 2|   |-0.126558|-0.00376|-0.02894|-0.06383|-0.08717|-0.07806|0.00911|0.189838|0.335648|0.319562|0.114325|-0.12656|
j: 3|   |-0.156434|-0.01540|-0.05679|-0.09984|-0.12419|-0.10196|0.00440|0.117312|0.173958|0.205763|0.105926|-0.15643|
j: 4|   |-0.126558|-0.00840|-0.03170|-0.05239|-0.04706|0.00463|0.046172|-0.016635|-0.061023|-0.016065|0.029257|-0.12656|
j: 5|   |-0.048341|0.01181|0.04059|0.06941|0.10811|0.106676|-0.025096|-0.153912|-0.16344|-0.10171|-0.019104|-0.04834|
j: 6|   |0.04834|0.019104|0.10171|0.16344|0.153912|0.025096|-0.106676|-0.10811|-0.06941|-0.04059|-0.01181|0.048341|
j: 7|   |0.12656|-0.029257|0.016065|0.061023|0.016635|-0.046172|-0.00463|0.04706|0.05239|0.03170|0.00840|0.126558|
j: 8|   |0.15643|-0.105926|-0.205763|-0.173958|-0.117312|-0.00440|0.10196|0.12419|0.09984|0.05679|0.01540|0.156434|
j: 9|   |0.12656|-0.114325|-0.319562|-0.335648|-0.189838|-0.00911|0.07806|0.08717|0.06383|0.02894|0.00376|0.126558|
j:10|   |0.04834|-0.059112|-0.184515|-0.225790|-0.162847|-0.08024|-0.03857|-0.02598|-0.02778|-0.03328|-0.01929|0.048341|
j:11|   |-0.04834|0.04834|0.14029|0.21851|0.27534|0.30521|0.305212|0.275336|0.218508|0.140291|0.048341|-0.048341|
Solution v:
        |   i:  0|   i:  1|   i:  2|   i:  3|   i:  4|   i:  5|   i:  6|   i:  7|   i:  8|   i:  9|   i: 10|   i: 11|

j: 0|   |0.048341|-0.048341|-0.126558|-0.156434|-0.126558|-0.048341|0.04834|0.12656|0.15643|0.12656|0.04834|-0.04834|
j: 1|   |-0.048341|0.01604|-0.00732|-0.01777|-0.00896|0.01371|0.02761|-0.010619|-0.094134|-0.121034|-0.068592|0.04834|
j: 2|   |-0.140291|0.02948|-0.03398|-0.06048|-0.03312|0.04411|0.11029|0.027418|-0.195288|-0.313856|-0.182487|0.14029|
j: 3|   |-0.218508|0.02710|-0.06555|-0.10086|-0.05202|0.07045|0.15687|0.046938|-0.174510|-0.323373|-0.210378|0.21851|
j: 4|   |-0.275336|0.02796|-0.08228|-0.11891|-0.04729|0.09731|0.131845|0.004158|-0.109454|-0.186289|-0.154520|0.27534|
j: 5|   |-0.305212|0.04211|-0.06599|-0.09779|-0.00840|0.089258|0.018080|-0.030590|0.00387|-0.01941|-0.080965|0.30521|
j: 6|   |-0.30521|0.080965|0.01941|-0.00387|0.030590|-0.018080|-0.089258|0.00840|0.09779|0.06599|-0.04211|0.305212|
j: 7|   |-0.27534|0.154520|0.186289|0.109454|-0.004158|-0.131845|-0.09731|0.04729|0.11891|0.08228|-0.02796|0.275336|
j: 8|   |-0.21851|0.210378|0.323373|0.174510|-0.046938|-0.15687|-0.07045|0.05202|0.10086|0.06555|-0.02710|0.218508|
j: 9|   |-0.14029|0.182487|0.313856|0.195288|-0.027418|-0.11029|-0.04411|0.03312|0.06048|0.03398|-0.02948|0.140291|
j:10|   |-0.04834|0.068592|0.121034|0.094134|0.010619|-0.02761|-0.01371|0.00896|0.01777|0.00732|-0.01604|0.048341|
j:11|   |0.04834|-0.04834|-0.12656|-0.15643|-0.12656|-0.04834|0.048341|0.126558|0.156434|0.126558|0.048341|-0.048341|
```

A.2. FinalProb.cpp

References

[Celik, 2006] Ismail B. Celik1, Urmila Ghia, Patrick J.Roache and Christopher J. Freitas "Procedure for Esitmation and Reporting Uncertainty Due to Discretization in CFD apllications", West Virginia University, Morgantown WV, USA