Dustin Ma
ME 2054 – Parallel Scientific Computing
10/1/2018

Homework 2

In this assignment, we implemented the concept of a flow between a stationary and a moving plate (couette flow) into our code. I program this flow to have a height of 2 units, as well as using a value of 1 for fluid properties such as density and viscosity. These values can be changed to match a realistic fluid. I selected such number to minimize the confusing that might arise while coding.

Beginning to solve the problem, the user enters the desired Reynolds number and the pressure gradient. With the given input, the rest of the variable terms such as velocity of the plate, kinematic viscosity, steady state time, and number of time steps required to reach a steady state are calculated based on this input. I tried to keep all the predefined terms as well as the dynamic memory allocation at the beginning of the file. Constants such as density, viscosity, height of the channel, and number of interval are given a constant float class (Note: I did not pick the int class because division of int gives either 1 or 0). The rest of the variable are defined as float. Memory allocation of array space was done using the command of malloc. Pointers were used in combination with the dynamic memory allocation for concise coding practice.

After the iterations are complete, the program writes the steady state velocity field from y = 0 to y = H along with the user input in a .out file. These results are printed into a text file in the same directory as the executable. Memory allocated at the beginning of the program is then freed and set to NULL.

Similar code was also implemented into MATLAB to produce Figure 1 and Figure 2 on page 2.

To submit the code in a presentable way, I utilized a twostep process to convert the .c file into a .pdf file. The a2ps function in the Linux library can convert any file into a postscript, .ps, file. The code below shows my command.

```
a2ps couette_flow.c -r --columns=2 --chars-per-line=100 --highlight-level=normal --
pro=color -o couette_flow.ps
```

As the man page describes, -r orients the page into landscape, --columns=2 makes two columns on the page, --chars-per-line=100 limits each line to have 100 characters spacing, --highlight-level=normal sets the highlight level, --pro sets the output with color, and -o specify the output. If output isn't specified, a2ps will send the postscript to the default printer.

After the conversion from .c into .ps, the ps2pdf function converts the post script file into a .pdf file. The code below shows my command.

```
ps2pdf couette_flow.ps couette_flow.pdf
```

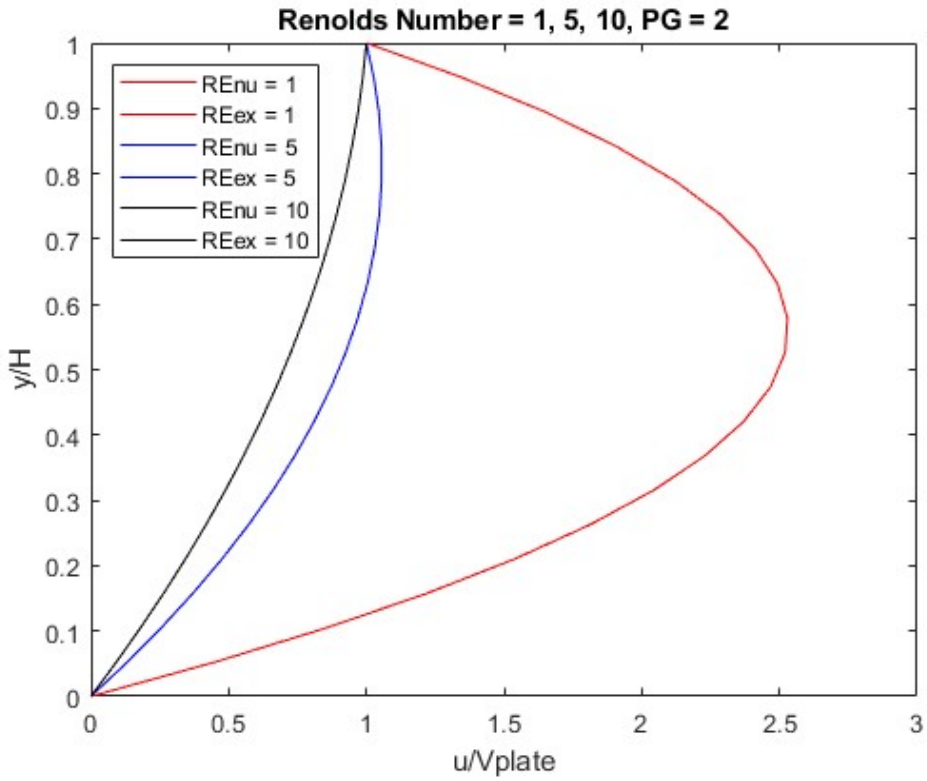

Figure 1. Reynolds Number = 1, 5, 10 with Constant pressure gradient = 2.

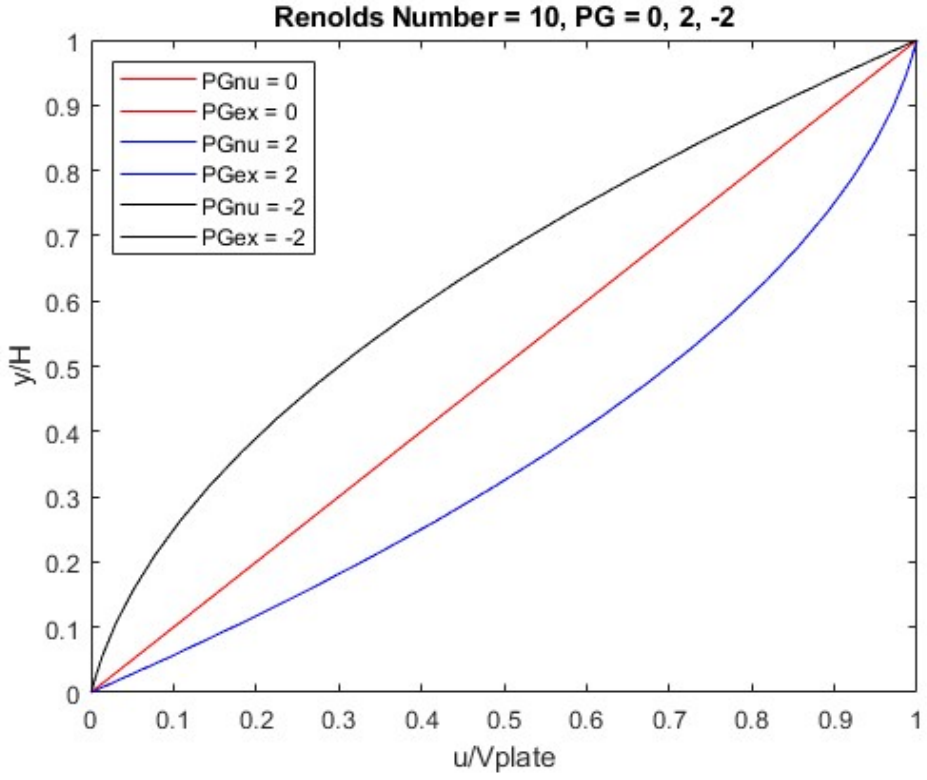

Figure 2. Constant Reynolds Number = 10 with pressure gradient = 0, 2, -2

Below shows the output.txt file when user specifics Re = 1.0 and PG = 2.0

Please enter (1)Reynolds Number and (2)Pressure Gradient:

You've chosen Re = 1.0 and P.G. = 2.0

Below shows the exact and numerical calculation for a couette flow.

You chose Re = 1.00, and PG = 2.0

Exact = 0.00000, Numerical = 0.00000

Exact = 0.45152, Numerical = 0.45152

Exact = 0.85873, Numerical = 0.85873

Exact = 1.22161, Numerical = 1.22161

Exact = 1.54017, Numerical = 1.54016

Exact = 1.81440, Numerical = 1.81440

Exact = 2.04432, Numerical = 2.04432

Exact = 2.22992, Numerical = 2.22992

Exact = 2.37119, Numerical = 2.37119

Exact = 2.46814, Numerical = 2.46814

Exact = 2.52078, Numerical = 2.52077

Exact = 2.52909, Numerical = 2.52908

Exact = 2.49307, Numerical = 2.49307

Exact = 2.41274, Numerical = 2.41274

Exact = 2.28809, Numerical = 2.28809

Exact = 2.11911, Numerical = 2.11911

Exact = 1.90582, Numerical = 1.90582

Exact = 1.64820, Numerical = 1.64820

Exact = 1.34626, Numerical = 1.34626

Exact = 1.00000, Numerical = 1.00000

```c
/*
 * ME 2054 Parallel Scientific Computing
 * Homework #2 - Couette Flow Solver
 * Due: October 1,2018
 *
 * Author: Dustin (Ting-Hsuan) Ma
 */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main(){

// Initializing variables
const float mu = 1.0, rho = 1.0, H = 2.0, numInt = 20.0;        // Arbitrary fluid values
float Re, PG, nu, Uplate, dy, dt, dp, nTimeStep, SteadyT;       // Internally defined values

//Allocating space for arrays
float *v = malloc(20*(sizeof *v));      // exact
float *u = malloc(20*(sizeof *u));      // numerical iteration
float *w = malloc(20*(sizeof *w));      // final numerical

// Asking for user input
printf("Please enter (1)Reynolds Number and (2)Pressure Gradient:\n");
scanf("%f%f", &Re, &PG);         // utilizing pointers
printf("You've chosen Re = %2.1f and P.G. = %1.1f\n",Re,PG);

// Calculation to define terms
dy = H / (numInt - 1);          // step size in y-axis
nu = mu / rho;                  // kinematic viscosity
Uplate = Re * nu / H;           // velocity of plate using user defined reynolds
dp = PG * (-rho);               // term for exact solution term
dt = 0.5 * (pow(dy,2) / nu);    // timestep
SteadyT = pow(H,2) / nu;        // expected steady state time
nTimeStep = SteadyT / dt;       // number of time step until steady state

// Calculating Exact Solution
int i = 0;
for (float y = 0; y < H + dy; y = y + dy){
        v[i] = Uplate * (y/H) + dp/(2*mu) * (pow(y,2) - H * y); // exact solution formula
        i++;    // incrementing index to store in array
}

// Calculating Numerical Solution
for (int i = 0; i < nTimeStep; i++){
        for (int j = 1; j < numInt; j++){
        u[j] = dt * (PG + mu * (w[j+1] - 2 * w[j] + w[j-1]) / pow(dy,2)) + w[j];

        // setting values into another vector for access in next iteration
        w[j] = u[j];

        // applying boundary conditions
        w[0] = 0;
        w[19] = Uplate;
}
}

// Output Final Solution
printf("Below shows the exact and numerical calculation for a couette flow.\n");
printf("You chose Re = %1.2f, and PG = %1.1f\n",Re,PG);
for (int i = 0; i < numInt; i++){
        printf("Exact = %-1.5f, Numerical = %-1.5f\n",v[i]/Uplate,w[i]/Uplate);
}

// Deallocating memory
free(v);free(w);free(u);
v = NULL;u=NULL;w=NULL;

return EXIT_SUCCESS;
}
```