MECH 510: Final Project

Athena Liu

Dec 18, 2019

1 General Validation

1.1 Correctness of residual

I wrote a subroutine `calculate_flux_integral()` in `navier-stokes.h/cpp` to compute the flux integral (residual) for the Navier-Stokes equation using a 2nd-order centered scheme. I verified my solution against the theoretical solution given by the instruction, setting $P_0 = v_0 = u_0 = 1$, $Re = 10$, $\beta = 1$ on a square domain of length 1. The L_2 error norms are reported in Table 1. As expected, the order of accuracy converges to 2, and the errors are reasonably small for fine meshes.

Table 1: L_2 errors of flux integral (residual) with mesh refinement.

	1st Component			2nd Component			3rd Component		
Mesh	L2 Error	Ratio	Order	L2 Error	Ratio	Order	L2 Error	Ratio	Order
10*10	4.7947E-02			1.9506E-01			1.9506E-01		
20*20	1.1983E-02	4.00	2.00	5.0026E-02	3.90	1.96	5.0026E-02	3.90	1.96
40*40	2.9957E-03	4.00	2.00	1.2586E-02	3.97	1.99	1.2586E-02	3.97	1.99
80*80	7.4892E-04	4.00	2.00	3.1514E-03	3.99	2.00	3.1514E-03	3.99	2.00

1.2 Correctness of flux Jacobian

The flux Jacobians are implemented in the source file `jacobians.h/cpp`, and I wrote a subroutine `assemble_jacobians()` in `navier-stokes.h/cpp` to compute the RHS of Equation (1) in the instruction. Setting $Re = 10$, δU as described in the instruction, and calculate the difference of the LHS and RHS of Equation (1). The non-zero values of (RHS-LHS) are reported in Table 2. As expected, they are all smaller than 10^{-10}, so we can assume that the flux Jacobians are correctly implemented.

Table 2: Absolute errors (RHS - LHS) of the approximation to Equation (1).

	1st Component			2nd Component			3rd Component		
	j=9	j=10	j=11	j=9	j=10	j=11	j=9	j=10	j=11
i=9	0	-1.0E-17	0	0	5.0E-12	0	0	5.0E-12	0
i=10	-1.0E-17	0	1.0E-17	5.0E-12	-1.0E-16	-5.0E-12	5.0E-12	-1.0E-16	-5.0E-12
i=11	0	1.0E-17	0	0	-5.0E-12	0	0	-5.0E-12	0

1.3 Block Thomas Algorithm

This part is implemented in the source file `tri_thomas.h/cpp`, which is an adaptation of the code provided on Canvas.

1.4 Approximate factorization scheme

Implicit Euler scheme (IE) with approximate factorization is implemented the subroutine `approximate_factorization()` in `navier-stokes.h/cpp`. I have verified my result after one time step against the given data on Canvas. (Actually, my results times Δt would match the given data, but it was because the approximate factorization that produced the given data was coded a bit differently, resulting in a factor of Δt difference.) My implicit Euler was correctly implemented.

2 Things to watch out for (Optimize my code)

2.1 Pressure oscillations

I did not have enough time to implement this.

2.2 Choice of β

I have experimented with a range of different values of β while setting $Re = 100$, $U_{top} = 1$, tolerance $= 10^{-6}$, $\Delta t = 0.1$, $h = w = 1$ on a 20×20 mesh. The convergence history of the change in P was plotted in Figure 1. (The convergence histories of du and dv were also plotted and they were more or less the same as dP's - usually a bit quicker, so I only presented the plot for dP for simplicity.) Based

on the convergence histories, it seems that $\beta = 0.75$ gives you a slight advantage than the default $\beta = 1$. $\beta = 0.5$ or even lower would give you a quick convergence at the first decades of iterations, then the benefit disappears as the error further reduces.

Therefore, I think $\beta = 0.75$ would be the optimal choice.

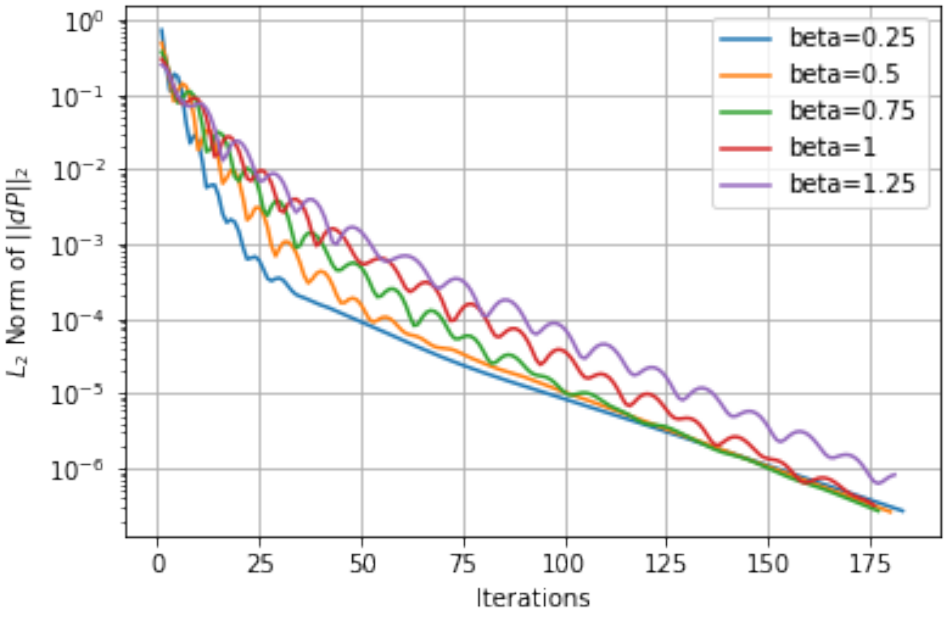

Figure 1: Convergence history of different choices of β.

2.3 Overrelaxation

I added overrelaxation in the implicit Euler scheme as the follwing:

$$U = U + \omega \cdot \delta U. \tag{1}$$

I have experimented with a range of different values of ω using the same setting as seciton 2.2 and $\beta = 0.75$. The convergence history of dP with different ω values is shown in Figure 2 as an example. Based on the convergence histories, the optimal overrelaxation parameter seems to be $\omega = 1.25$.

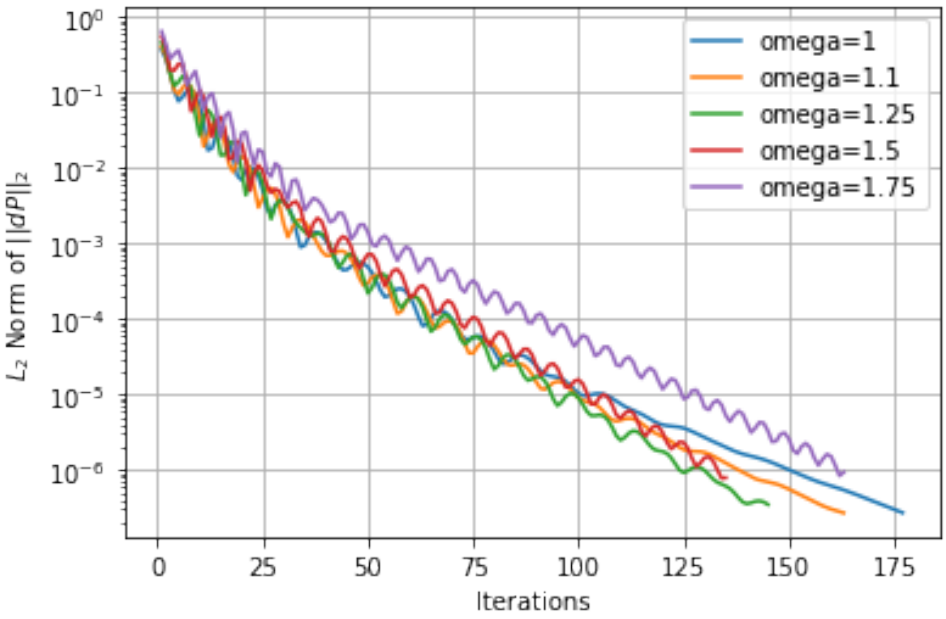

Figure 2: Convergence history of different choices of ω.

3 Flow in a Box with a Moving Top

In this section, the flow in a box of size 1 by 1 with $Re = 100$ and various top wall velocities U_{Top} is studied. The optimizations presented in Section 2 were not used in this section, i.e. , $\beta = 1$, $\omega = 1$, and no pressure smoothening were used in this section.

3.1 Validation case: Stability

To validate everything I have coded so far, I ran a case with $\Delta t = 0.05$, $U_{Top} = 0$ on a 20×20 mesh for 200 time steps. The convergence history is plotted in Figure 3. The change in P, u, v decreases with some oscillation, which is the expected behaviour of the the Navier-Stokes solver in use. The stability of the solver is confirmed.

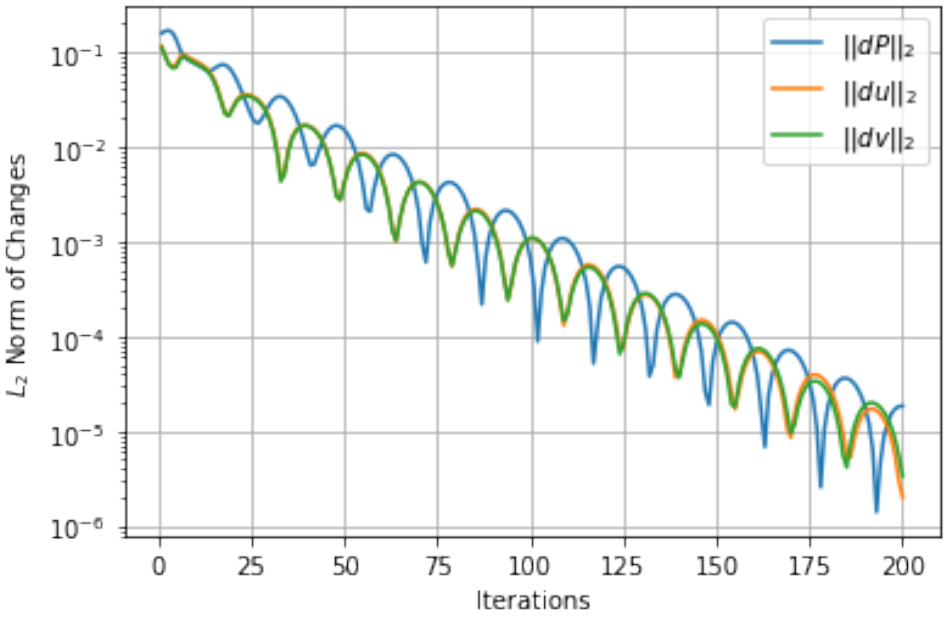

Figure 3: Convergence history of the case with $U_{Top} = 0$, $\Delta t = 0.05$ on a 20×20 mesh.

3.2 Basic Solution

In this section, cases of moving top wall velocities are studied. The time step is chosen to be $\Delta t = 0.1$ for faster convergence. Tolerance is set to be 10^{-6}.

3.2.1 Solution for $U_{Top} = 1$

First, I solved the problem for $U_{Top} = 1$ on a 20×20 mesh.

- The convergence history is plotted in Figure 4.
- The plot of u along the symmetry line ($x = 1/2$) is shown in Figure 5. The mid line u is found by interpolating the cell data on the left and right of the mid line, i.e.,

$$u_{Ni+\frac{1}{2},j} = (u_{Ni/2,j} + u_{Ni/2+1,j})/2. \tag{2}$$

We observe that $u = 1$ at the top and $u = 0$ at the bottom, and u reaches the minimum at about $y = 0.5$, which are the expected behaviours for such a problem.

- The contour plots of P, u and v are presented in Figure 6.

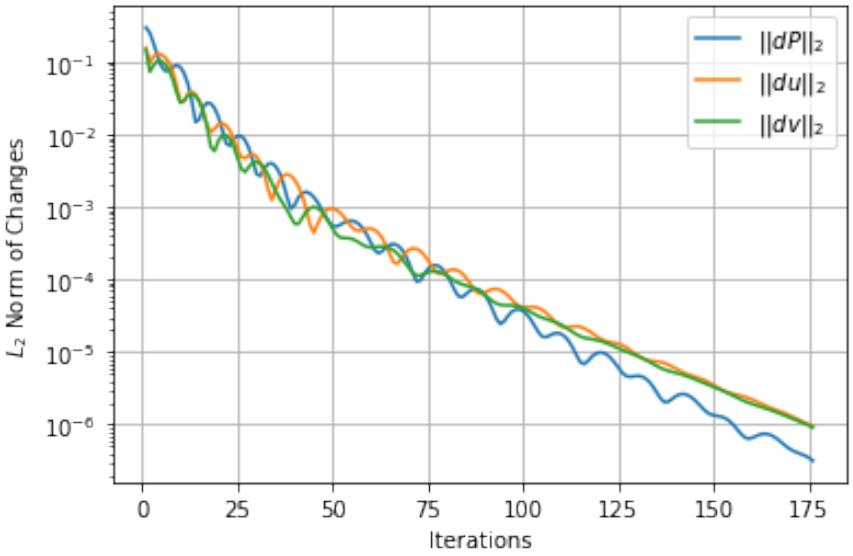

Figure 4: Convergence history of the case with $U_{Top} = 1$, $\Delta t = 0.1$ on a 20×20 mesh.

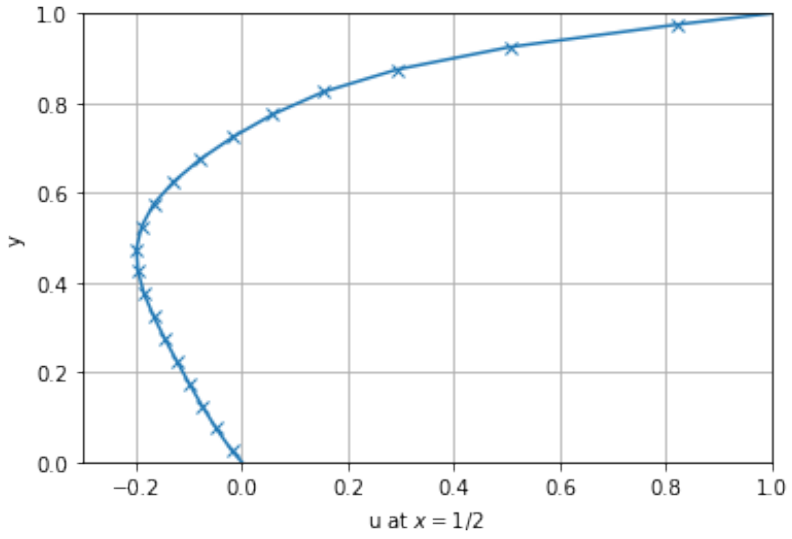

Figure 5: u along $x = 1/2$ for $U_{Top} = 1$, $\Delta t = 0.1$ on a 20×20 mesh.

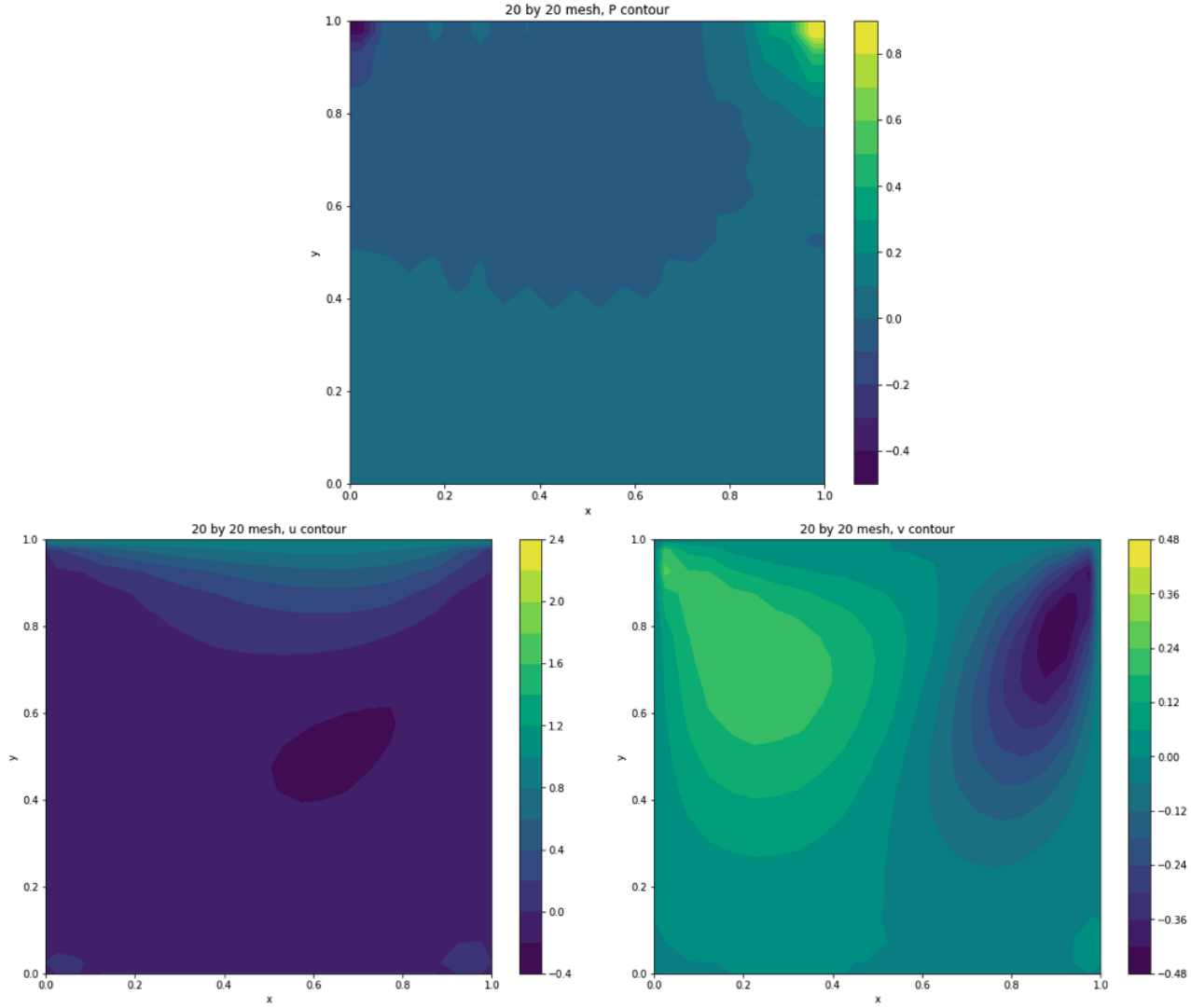

Figure 6: Contour plots for $U_{Top} = 1$, $\Delta t = 0.1$. Top: P; bottom left: u; bottom right: v.

3.2.2 Sanity check: Symmetry

In this section, we solve a problem that is the symmetric version of Section 3.2.1 with $U_{Top} = -1$, and same settings otherwise. The solution fields should be exactly the opposite of those in Section 3.2.1. In fact, plotting the combination of the two set of solutions: $u_{U_{top}=1}(x, y) + u_{U_{top}=-1}(1 - x, y)$, they should add up to almost zero, see Figure 7. The sum is all of size 10^{-6} or lower, which is the size of the

tolerance. This indicates that my solver passes the sanity check.

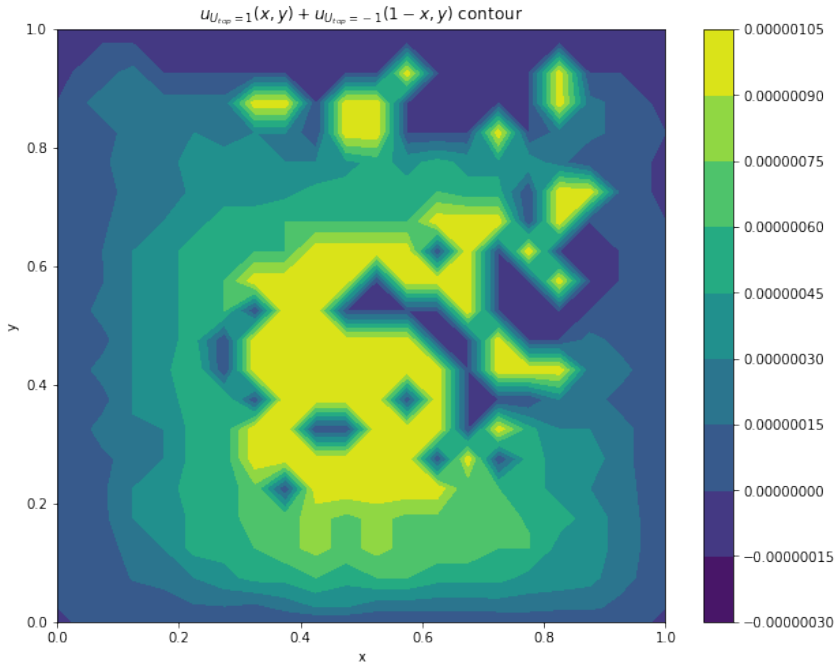

Figure 7: Sanity check: contour of $u_{U_{top}=1}(x,y) + u_{U_{top}=-1}(1-x,y)$.

3.2.3 Grid convergence

In this section, we study the grid convergence for the problem of $U_{top} = 1$. Results on meshes from 10×10 up to 160×160 are obtained. Results of the $u(y)$ profile along the symmetry line for each mesh are plotted in Figure 8, with both the full profile and a zoomed in profile near $y = 1/2$. The largest discrepancy occurs at around $y = 1/2$, where the velocity shear is the largest. Looking at the zoomed in u profile, it seems that the result on a 80×80 mesh and that on a 160×160 are almost overlapping, suggesting that 80×80 mesh might be fine enough to reach grid convergence, but this need to be backed up quantitatively.

To study mesh convergence quantitatively, I studied the numerical error of the minimum value of u along $x = 1/2$. The minimum value is found by interpolating the three adjacent data points that have the smallest values using a second order polynomial, and then report the theoretical minimum value of the interpolated polynomial. The grid convergence report as per ASME JFE guidelines is presented in

Table 3. On the 80×80 mesh, the GCI has already dropped below 1%. Therefore, I conclude that a 80×80 mesh is sufficient for grid convergence of u along $x = 1/2$.

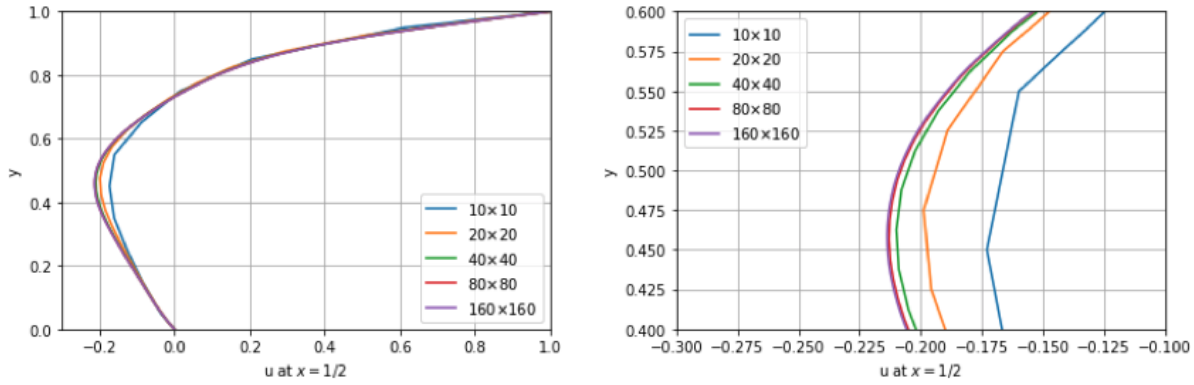

Figure 8: u along $x = 1/2$ for $U_{Top} = 1$ on different meshes. Left: full profile; right: zoomed in near $y = 0.5$.

Table 3: Grid convergence report of min(u) along $x = 1/2$.

Mesh	min(u)	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
10*10	-0.17297						
20*20	-0.19923	0.02626	-0.13181				
40*40	-0.20999	0.01076	-0.05126	1.29	-0.21747	3.43857%	4.45127%
80*80	-0.21304	0.00304	-0.01428	1.82	-0.21424	0.55964%	0.70349%
160*160	-0.21380	0.00076	-0.00355	2.00	-0.21405	0.11799%	0.14766%

4 Exploratory case: Effect of h

In this section, we study the lid driven flow problem for a taller box with $Re = 300$, $U_{top} = 1$, $w = 1$, $h = 2.5$. Results were obtained with $tolerance = 10^{-9}$ and meshes of $\Delta x = \Delta y = 1/\{20, 40, 80, 160, 320\}$, respectively.

4.1 General visualization

The goal is to find out how many vortices appear in this flow configuration and the strength of them. First, given the velocity fields, we can do some post processing to get the velocity magnitude field:

$$|velocity|_{i,j} = \sqrt{u_{i,j}^2 + v_{i,j}^2} \tag{3}$$

and the vorticity field:

$$vorticity = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \approx \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}. \tag{4}$$

Then plot the streamlines using (u, v), and plot the velocity magnitude field and the vorticity contour. See Figure 9. Based on the streamline plot, we observe that **there are 4 vortices**: a top one, a bottom one, and two small ones at the left and right bottom corners. In next section, we describe the procedure to detect the exact location of the vortices and calculate its strength.

4.2 Vortex detection procedure

Each vortex and its strength is determined through the following steps:

1. Narrow down the target area for searching vortex by looking at the streamline plot. For example, to detect the top vortex, we look at the domain of $[0.4 \leq x \leq 0.8] \times [2 \leq y \leq 2.2]$.

2. Within the target area, find the cell (i_v, j_v) with the smallest velocity magnitude.

3. Since u change direction from $j_v - 1$ to $j_v + 1$, we interpolate (y, u) at the 3 cells $(i_v, j_v - 1)$, (i_v, j_v) and $(i_v, j_v + 1)$ using a 2nd order polynomial; then find the y location of the vortex y_{loc} by setting the interpolated $u(y) = 0$ and solve for the root.

4. Similarly, for x_{loc}, we interpolate (x, v) at the cells $(i_v - 1, j_v)$, (i_v, j_v) and $(i_v + 1, j_v)$ using a 2nd order polynomial, and solving for x in $v(x) = 0$.

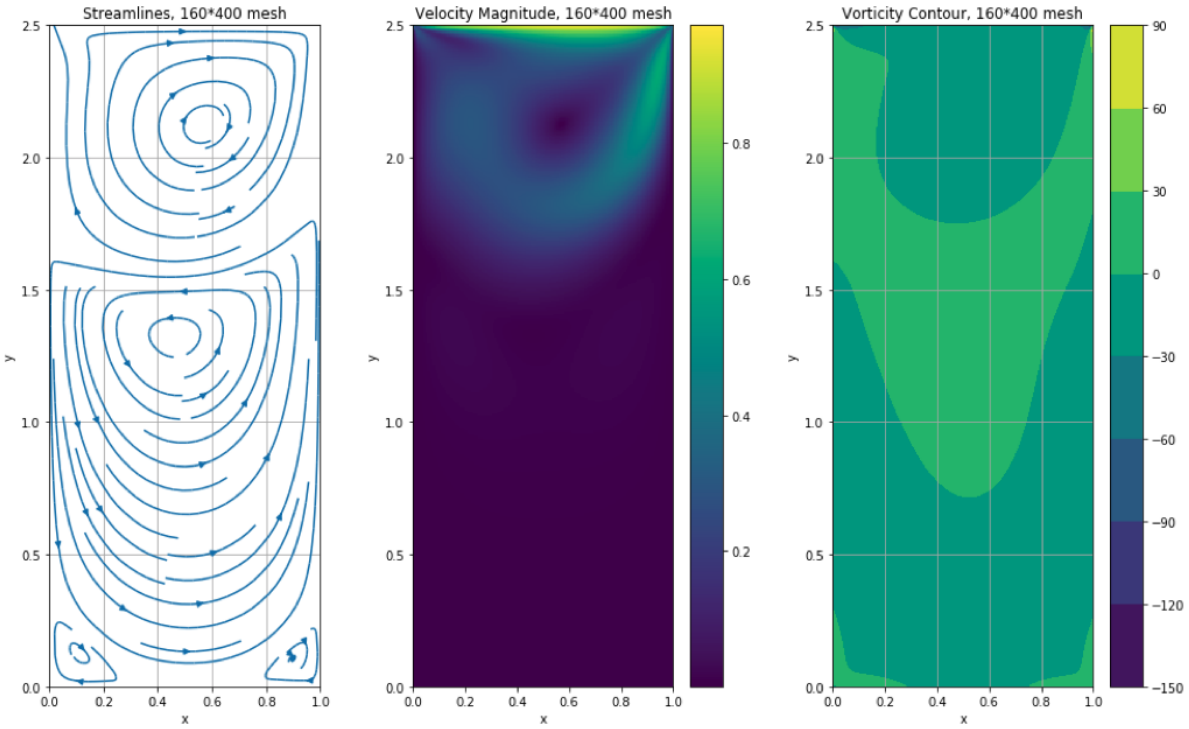

Figure 9: Streamlines, velocity magnitude and vorticity contour for lid driven flow in a taller box.

5. To get the vorticity at the vortex location, we

   - interpolate the vorticity at $(i_v, j_v - 1)$, (i_v, j_v) and $(i_v, j_v + 1)$ using a 2nd order polynomial, getting $vort1(y)$;
   - interpolate the vorticity at $(i_v - 1, j_v)$, (i_v, j_v) and $(i_v + 1, j_v)$ using a 2nd order polynomial, getting $vort2(x)$;
   - calculate the vorticity by $[vort1(y_{loc}) + vort2(x_{loc})]/2$. This is the vortex strength that we will report.

This procedure is done for each of the four vortices. The velocity magnitude at each target area is plotted in Figure 10. The darkest point with the least velocity magnitude is where the vortex is located. It is worth mentioning that the two bottom corner vortices are very weak, and they only start to appear in very fine meshes of size $\Delta x \leq 1/160$ - in coarser meshes, the velocity magnitudes oscillates slightly and it was not possible to detect the vortex properly.

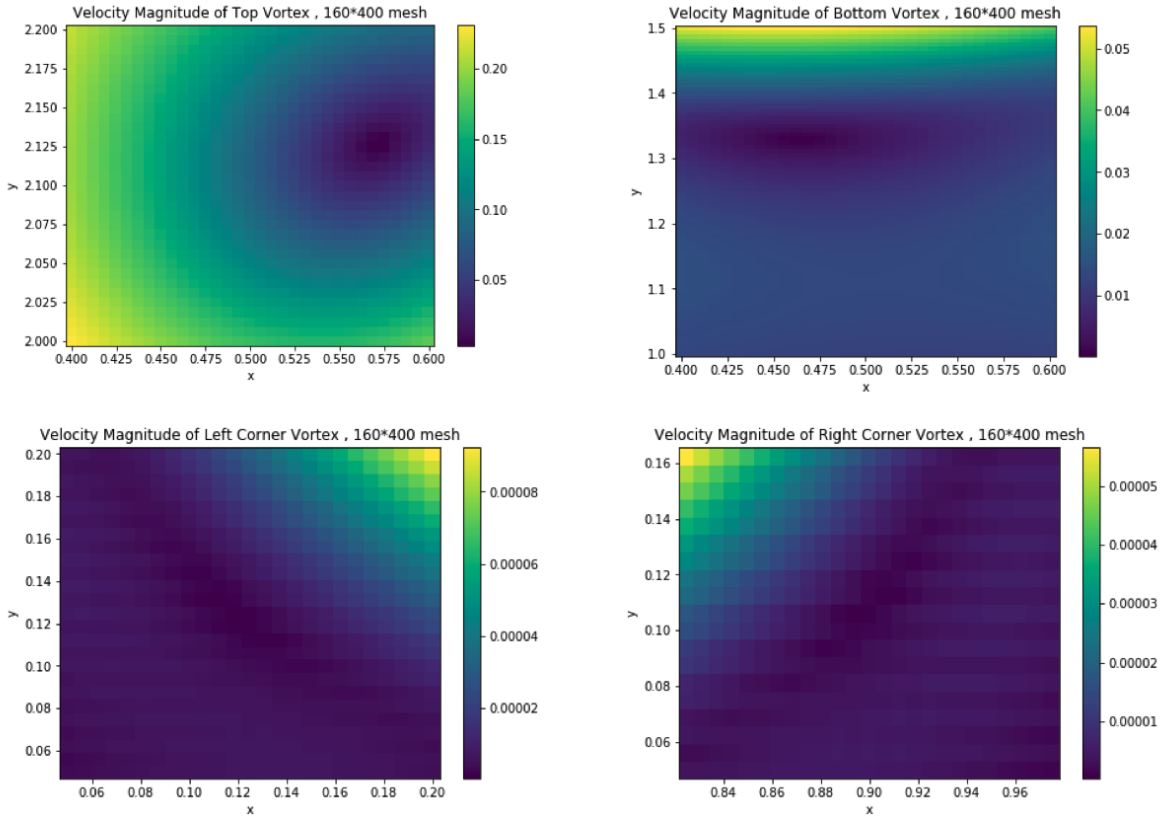

Figure 10: Velocity magnitude at the target areas for the 4 vortices.

4.3 Report of the vortex location and strength

There are 4 vortices in this flow configuration. Their location and strength are summarized as follows:

- Top vortex: at $(0.56816 \pm 0.001\%, 2.12374 \pm 0.007\%)$, vorticity: $-2.41436 \pm 0.004\%$.
- Bottom vortex: at $(0.46063 \pm 0.03\%, 1.32767 \pm 0.015\%)$, vorticity: $0.25486 \pm 0.003\%$.
- Left corner vortex: at $(0.11557 \pm 0.06\%, 0.12434 \pm 0.013\%)$, vorticity: $-0.00038 \pm 0.03\%$.
- Right corner vortex: at $(0.89811 \pm 0.007\%, 0.10745 \pm 0.05\%)$, vorticity: $-0.00030 \pm 0.02\%$.

The detailed report as per ASME JFE guidlines is shown in Table 4 to 7.

Seeing from the tables, for the first two vortices, the GCI drops below 1% even at the 80×200 mesh, so they have reached grid convergence even on a relatively coarse mesh. But there *is* one mystery that

I cannot quite explain: I am not sure why the orders of `x_loc` and `y_loc` for the top vortex did not converge to 2 as I had expected; however, these quantities for the bottom vortex did have orders that converge to 2.

For the two weak corner vortices, one would need much finer meshes to reach mesh convergence. I do not have the computational resources to compute on a third mesh (640×1600) to calculate the GCI. Based on the two sets of results that I do have, the approximate relative errors are below 5%, so it is a good indication that these quantities are at least close to grid convergence. If more computation resources are allowed, results on finer meshes should be obtained to assess the grid convergence for the two corner vortices.

Table 4: Top vortex: location, strength and error bounds

Mesh	x_loc	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	0.58598						
40*100	0.57111	0.01487	0.02604				
80*200	0.56835	0.00276	0.00486	2.43	0.56771	0.11103%	0.13863%
160*400	0.56812	0.00022	0.00040	3.62	0.56810	0.00351%	0.00438%
320*800	0.56815	0.00003	0.00006	2.74	0.56816	0.00104%	0.00130%
Mesh	y_loc	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	2.12842						
40*100	2.12400	0.00442	0.00208				
80*200	2.12392	0.00008	0.00004	5.85	2.12392	0.00006%	0.00008%
160*400	2.12336	0.00056	0.00027	-2.88	2.12401	0.03073%	0.03843%
320*800	2.12358	0.00023	0.00011	1.32	2.12374	0.00715%	0.00894%
Mesh	vorticity	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	-2.21874						
40*100	-2.36504	0.14630	-0.06186				
80*200	-2.40149	0.03645	-0.01518	2.01	-2.41358	0.50099%	0.62939%
160*400	-2.41347	0.01198	-0.00496	1.61	-2.41934	0.24253%	0.30390%
320*800	-2.41453	0.00106	-0.00044	3.50	-2.41463	0.00428%	0.00535%

Table 5: Bottom vortex: location, strength and error bounds

Mesh	x_loc	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	0.48645						
40*100	0.46951	0.01694	0.03607				
80*200	0.46291	0.00660	0.01426	1.36	0.45870	0.91867%	1.13788%
160*400	0.46117	0.00174	0.00377	1.92	0.46055	0.13511%	0.16866%
320*800	0.46076	0.00041	0.00089	2.08	0.46063	0.02782%	0.03476%

Mesh	y_loc	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	1.28776						
40*100	1.31658	0.02882	0.02189				
80*200	1.32455	0.00797	0.00602	1.85	1.32760	0.22934%	0.28733%
160*400	1.32690	0.00235	0.00177	1.76	1.32788	0.07368%	0.09217%
320*800	1.32748	0.00058	0.00044	2.02	1.32767	0.01432%	0.01790%

Mesh	vorticity	Difference	Apprx Rel Err	Order	Extrap Value	Extrap Rel Err	GCI
20*50	0.17392						
40*100	0.23011	0.05619	0.24419				
80*200	0.24564	0.01552	0.06320	1.86	0.25156	2.35563%	3.01557%
160*400	0.25515	0.00951	0.03727	0.71	0.27018	5.56387%	7.36460%
320*800	0.25487	0.00028	0.00108	5.11	0.25486	0.00322%	0.00402%

Table 6: Left corner vortex: location, strength and error bounds

Mesh	x_loc	Difference	Apprx Rel Err
160*400	0.10828		
320*800	0.11557	0.00729	0.06306
Mesh	y_loc	Difference	Apprx Rel Err
160*400	0.12588		
320*800	0.12434	0.00154	0.01238
Mesh	vorticity	Difference	Apprx Rel Err
160*400	-0.00037		
320*800	-0.00038	0.00001	-0.03161

Table 7: Right corner vortex: location, strength and error bounds

Mesh	x_loc	Difference	Apprx Rel Err
160*400	0.89178		
320*800	0.89811	0.00633	0.00705
Mesh	y_loc	Difference	Apprx Rel Err
160*400	0.10196		
320*800	0.10745	0.00549	0.05111
Mesh	vorticity	Difference	Apprx Rel Err
160*400	-0.00031		
320*800	-0.00030	0.00001	-0.02032