دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی مکانیک

پروژه دینامیک سیالات محاسباتی
حل عددی مسئله ی جریان تراکم ناپذیر سیال
در یک حفره ی دو بعدی

نگارش

هانیه عطریان

درسا طیبی

استاد درس

دکتر احمدپور

تیر ماه ۱۴۰۳

بسم الله الرحمن الرحیم

فهرست مطالب

بخش اول : معادلات حاکم و کد ملوار ۱

شرح معادلات ورتیسیه – تابع جریان ۱

خطوط جریان و خطوط تاوایی ثابت درون محفظه ٤

مقدار تاوایی بر روی تمامی دیوار ها ۲

بخش دوم - اصلاح کد به روش SOR ٦

شرح معادلات SOR ٦

پروفیل سرعت افقی بر روی خط عمودی و افقی میانی حفره ۷

پروفیل سرعت عمودی بر روی خط عمودی و افقی میانی حفره ۹

تنش برشی بر روی دیواره ی بالایی و پایینی ۱۱

بخش سوم - مقایسه دو روش ۱۳

کد ملوار ۱۳

روش SOR ۱۳

Under relaxation ۱۳

بخش چهارم - معادله حاکم بر سیال دو بعدی ۱٥

شرح معادلات دو بعدی ۱٥

کد های متلب ۱۷

کد ملوار اصلاح شده ۱۷

کد SOR ۱۹

کد بخش امتیازی ۲۹

منابع و مراجع ۳۲

بخش اول : معادلات حاکم و کد ملوار

شرح معادلات ورتیسیه – تابع جریان

معادلات تراکم ناپذیر ناویر استوکس برای دو بعد به شرح زیر می باشند :

$$X\ Momentum \rightarrow \rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial P}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\rho g_x$$

$$Y\ Momentum \rightarrow \rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial P}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\rho g_y$$

$$2D\ continuity \rightarrow \frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

برای حذف کردن ترم فشار از معادلات، از معادله ی مومنتوم x نسبت به y و از معادله ی مومنتوم y نسبت به x مشتق گرفته شده و در نهایت با هم جمع می شوند.

$$\frac{\partial^2 u}{\partial y\partial t}+\frac{\partial^2 u}{\partial x\partial y}+u\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial u\partial v}{\partial y^2}+v\frac{\partial^2 v}{\partial y^2}=-\frac{1}{\rho}\frac{\partial^2 P}{\partial x\partial y}+\frac{\mu}{\rho}\left(\frac{\partial^3 u}{\partial x^2\partial y}+\frac{\partial^3 u}{\partial y^3}\right)$$

$$\frac{\partial^2 v}{\partial x\partial t}+\frac{\partial^2 v}{\partial x\partial y}+v\frac{\partial^2 v}{\partial x\partial y}+\frac{\partial u\partial v}{\partial x^2}+u\frac{\partial^2 v}{\partial x^2}=-\frac{1}{\rho}\frac{\partial^2 P}{\partial x\partial y}+\frac{\mu}{\rho}\left(\frac{\partial^3 v}{\partial y^2\partial x}+\frac{\partial^3 v}{\partial x^3}\right)$$

$$\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)+u\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)+v\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)+\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)$$

$$=\frac{\mu}{\rho}\left(\frac{\partial^2}{\partial x^2}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)+\frac{\partial^2}{\partial y^2}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)\right)$$

ورتیسیه به صورت زیر تعریف می شود :

$$\Omega=\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}$$

با جایگذاری ورتیسیته در معادله تفاضل شده معادله ی انتقال ورتیسیته به دست می آید :

$$\frac{\partial \Omega}{\partial t}+u\frac{\partial \Omega}{\partial x}+v\frac{\partial \Omega}{\partial y}=\frac{\mu}{\rho}\left(\frac{\partial^2 \Omega}{\partial x^2}+\frac{\partial^2 \Omega}{\partial y^2}\right)$$

هم چنین می توان سرعت ها را به صورت تابعی از تابع جریان نوشت :

$$u = \frac{\partial\psi}{\partial y} \quad , \qquad v = \frac{\partial\psi}{\partial x}$$

با جایگذاری سرعت های فوق در معادله ی انتقال ورتیسیته این معادله به صورت زیر بازنویسی می شود.

$$\frac{\partial\Omega}{\partial t} + (\frac{\partial\psi}{\partial y})\frac{\partial\Omega}{\partial x} + (\frac{\partial\psi}{\partial x})\frac{\partial\Omega}{\partial y} = \frac{\mu}{\rho}\left(\frac{\partial^2\Omega}{\partial x^2} + \frac{\partial^2\Omega}{\partial y^2}\right)$$

$$\frac{\Omega_{i,j}^{n+1} - \Omega_{i,j}^{n}}{dt} = -\left(\frac{\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n}}{2dy}\right)\left(\frac{\Omega_{i+1,1}^{n} - \Omega_{i-1,j}^{n}}{2dx}\right) + \cdots$$

$$\left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n}}{2dx}\right)\left(\frac{\Omega_{i,j+1}^{n} - \Omega_{i,j-1}^{n}}{2dy}\right) + \frac{\mu}{\rho}\left(\frac{\Omega_{i+1,j}^{n} - 2\Omega_{i,j}^{n} + \Omega_{i-1,j}^{n}}{dx^2} + \frac{\Omega_{i,j+1}^{n} - 2\Omega_{i,j}^{n} + \Omega_{i,j-1}^{n}}{dy^2}\right)$$

$$\Omega_{i,j}^{n+1} = \Omega_{i,j}^{n} + (-\left(\frac{\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n}}{2h}\right)\left(\frac{\Omega_{i+1,1}^{n} - \Omega_{i-1,j}^{n}}{2h}\right) + \cdots$$

$$\left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n}}{2h}\right)\left(\frac{\Omega_{i,j+1}^{n} - \Omega_{i,j-1}^{n}}{2h}\right) + \frac{\mu}{\rho}\left(\frac{\Omega_{i+1,j}^{n} - 2\Omega_{i,j}^{n} + \Omega_{i-1,j}^{n} + \Omega_{i,j+1}^{n} - 2\Omega_{i,j}^{n} + \Omega_{i,j-1}^{n}}{h^2}\right)$$

$$\Omega_{i,j}^{n} = \frac{\psi_{i+1,j}^{n} - 2\psi_{i,j}^{n} + \psi_{i-1,j}^{n}}{dx^2} + \frac{\psi_{i,j+1}^{n} - 2\psi_{i,j}^{n} + \psi_{i,j-1}^{n}}{dy^2}$$

$$\psi_{i,j}^{n} = \frac{\Omega_{i,j}^{n}h^2 + \psi_{i+1,j}^{n} + \psi_{i-1,j}^{n} + \psi_{i,j+1}^{n} + \psi_{i,j-1}^{n}}{4}$$

شرط های مرزی تابع جریان :

$$Top \rightarrow \psi_{(:,N-1)} = -u_{(:,N)}dy - \Omega_{(:,N)}\frac{dy^2}{2} \rightarrow \Omega_{(:,N)}\frac{dy^2}{2} = -\psi_{(:,N-1)} - u_{(:,N)}dy$$

$$Bottom \rightarrow \psi_{(:,2)} = -u_{(:,1)}dy - \Omega_{(:,1)}\frac{dy^2}{2} \rightarrow \Omega_{(:,1)}\frac{dy^2}{2} = -\psi_{(:,2)} + u_{(:,1)}dy$$

$$Left \rightarrow \psi_{(2,:)} = -v_{(1,:)}dx - \Omega_{(1,:)}\frac{dx^2}{2} \rightarrow \Omega_{(1,:)}\frac{dx^2}{2} = -\psi_{(2,:)} - v_{(1,:)}dx$$

$$Right \rightarrow \psi_{(N-1,:)} = v_{(N,:)}dx - \Omega_{(N,:)}\frac{dx^2}{2} \rightarrow \Omega_{(N,:)}\frac{dx^2}{2} = -\psi_{(N-1,:)} - v_{(N,:)}dx$$

شرط های مرزی ورتیسیته :

$$Top \rightarrow \Omega_{(:,N)} = \frac{-2\psi_{(:,N-1)}}{dy^2} - \frac{2U_{wall}}{dy}$$

$$Bottom \rightarrow \Omega_{(:,1)} = \frac{-2\psi_{(:,2)}}{dy^2}$$

$$Left \rightarrow \Omega_{(1,:)} = \frac{-2\psi_{(2,:)}}{dx^2}$$

$$Right \rightarrow \Omega_{(N,:)} = \frac{-2\psi_{(N-1,:)}}{dx^2}$$

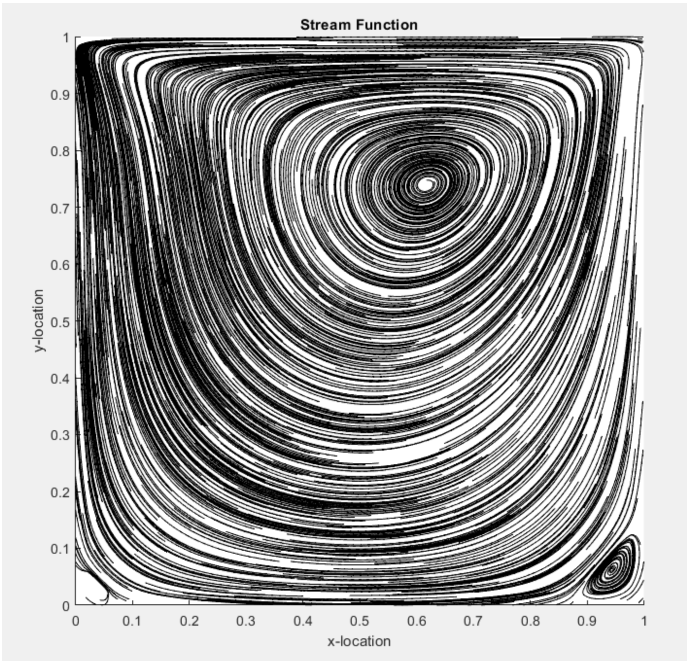
گسسته سازی سرعت ها نیز به شکل زیر می باشد.

$$u = \frac{\partial\psi}{\partial y} = \frac{\Psi_{i,j+1}^{n} + \Psi_{i,j-1}^{n}}{2h}$$

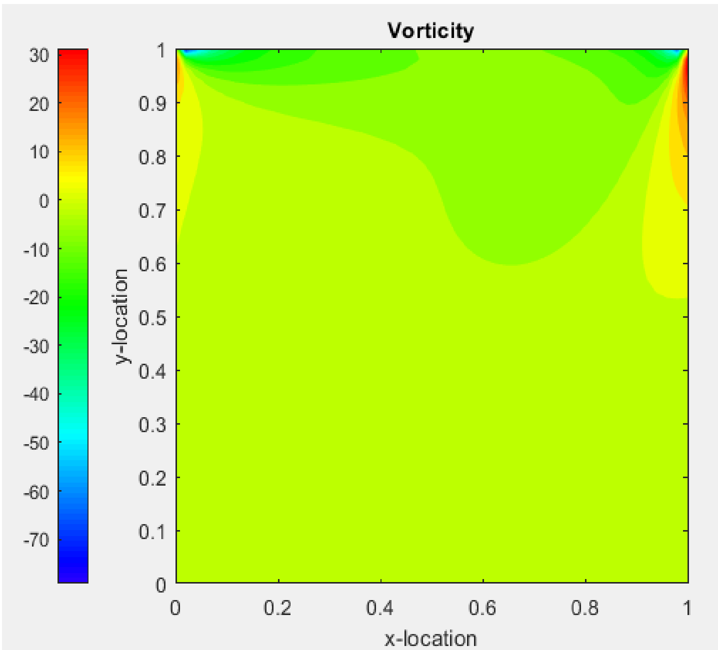
$$v = \frac{\partial\psi}{\partial x} = \frac{-\Psi_{i+1,j}^{n} + \Psi_{i-1,j}^{n}}{2h}$$

خطوط جریان و خطوط تاوایی ثابت درون محفظه

خطوط جریان درون محفظه و خطوط تاوایی ثابت برای رینولدز برابر با ۱۰۰ و سرعت دیواره برابر با ۱ متربرثانیه و تعداد نودهای برابر با ۵۲ به شرح زیر می باشد :



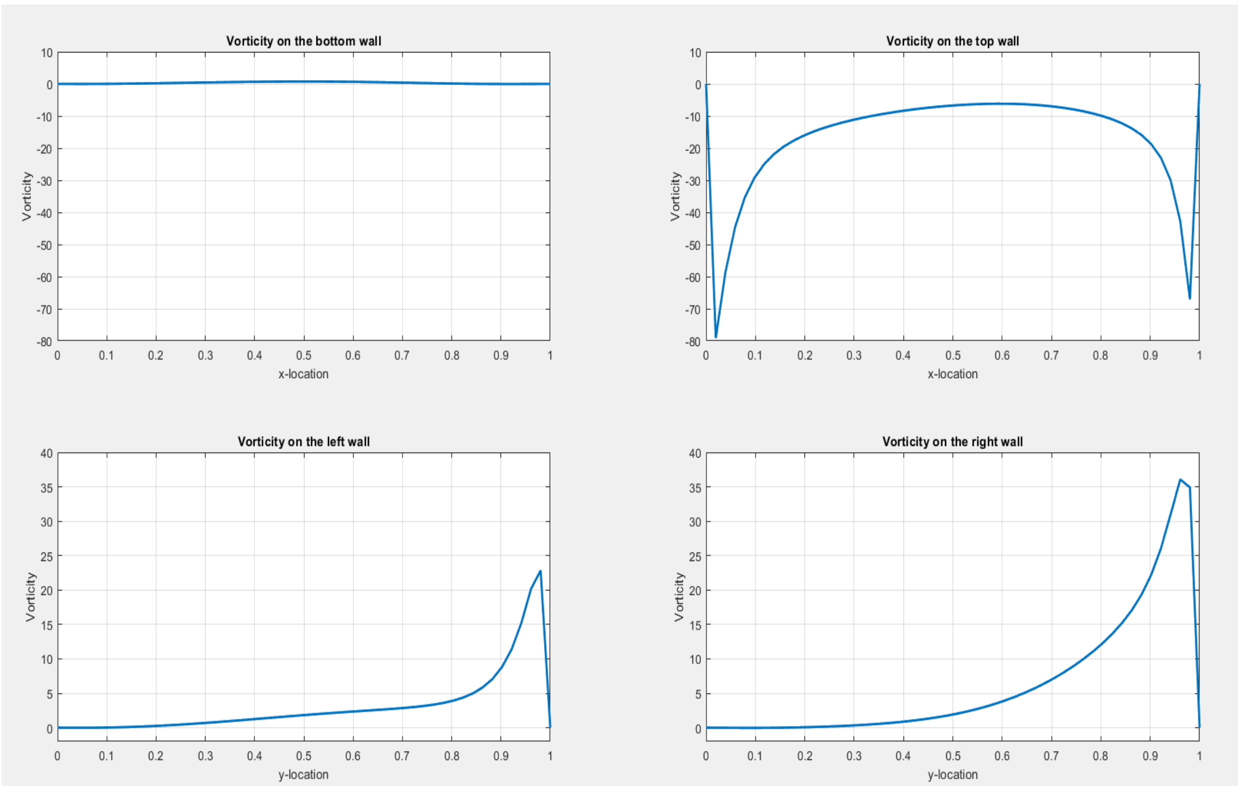
شکل ۱ – خطوط جریان داخل محفظه



شکل ۲ – خطوط تاوایی ثابت داخل محفظه

مقدار تاوایی بر روی تمامی دیوار ها

همچنین مقدار تاوایی بر روی ۴ دیوار به تفکیک به صورت زیر می باشد :



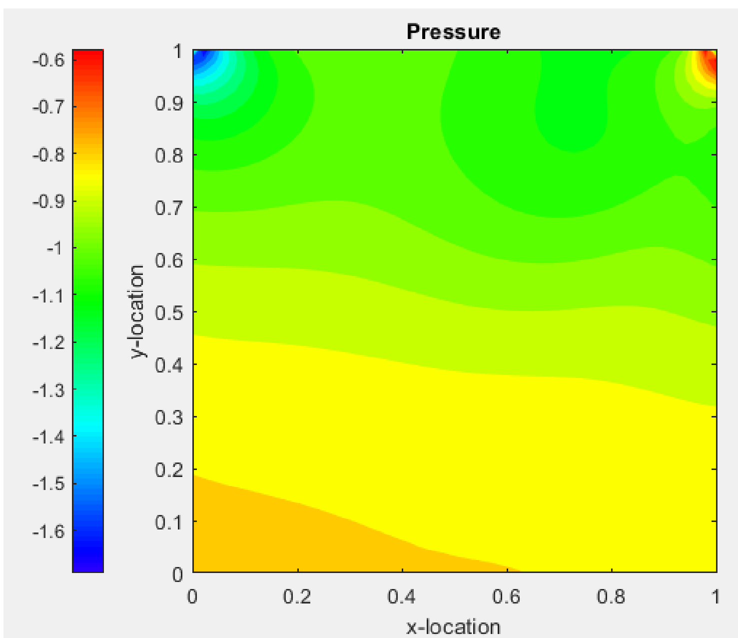
شکل ۳ – مقدار تاوایی بر روی دیوارها

کانتور فشار

برای به دست آوردن فشارها باید از جاگذاری سرعت ها در معادله تکانه اصلی استفاده نمود.

```
%%% PRESSURE TOP BOUNDARY CONDITION
P(I,Ny) = 1/3*(4*P(I,Ny-1)-P(I,Ny-2))+(2*mu)/(3*h)*(-5*v(I,Ny-1)+4*v(I,Ny-2)-v(I,Ny-3));
%%% PRESSURE BOTTOM BOUNDARY CONDITION
P(I,1) = 1/3*(4*P(I,2)-P(I,3))-(2*mu)/(3*h)*(-5*v(I,2)+4*v(I,3)-v(I,4));
%%% PRESSURE RIGHT BOUNDARY CONDITION
P(Nx,J) = 1/3*(4*P(Nx-1,J)-P(Nx-2,J))+(2*mu)/(3*h)*(-5*u(Nx-1,J)+4*u(Nx-2,J)-u(Nx-3,J));
%%% PRESSURE LEFT BOUNDARY CONDITION
P(1,J) = 1/3*(4*P(2,J)-P(3,J))-(2*mu)/(3*h)*(-5*u(2,J)+4*u(3,J)-u(4,J));
%%% PRESSURE FOR INNER NODES
P(i,j) = 0.25*(P(ip,j)+P(im,j)+P(i,jp)+P(i,jm))-rho/2*(1/(h^2)*(St(ip,j)-2*St(i,j)+....
St(im,j)).*(St(i,jp)-2*St(i,j)+St(i,jm))-1/(16*h^2)*(St(ip,jp)-St(ip,jm)-St(im,jp)+St(im,jm)).^2);
```

در نتیجه کانتور فشار حاصله با ۵۲ نود و رینولدز ۱۰۰ و سرعت دیواره ۱ متر بر ثانیه به صورت زیر است :



شکل ۴ – کانتور فشار

مقایسه سرعت ها

سرعت های نگارش شده در مقاله قیا در رینولدز برابر با ۱۰۰ و ۴۰۰ و تعداد نود های ۱۲۹ و سرعت دیواره برابر با ۱ متر بر ثانیه به شرح زیر می باشد :

TABLE I

Results for *u*-velocity along Vertical Line through Geometric Center of Cavity

129-grid pt. no.	y	Re						
		100	400	1000	3200	5000	7500	10,000
129	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
126	0.9766	0.84123	0.75837	0.65928	0.53236	0.48223	0.47244	0.47221
125	0.9688	0.78871	0.68439	0.57492	0.48296	0.46120	0.47048	0.47783
124	0.9609	0.73722	0.61756	0.51117	0.46547	0.45992	0.47323	0.48070
123	0.9531	0.68717	0.55892	0.46604	0.46101	0.46036	0.47167	0.47804
110	0.8516	0.23151	0.29093	0.33304	0.34682	0.33556	0.34228	0.34635
95	0.7344	0.00332	0.16256	0.18719	0.19791	0.20087	0.20591	0.20673
80	0.6172	−0.13641	0.02135	0.05702	0.07156	0.08183	0.08342	0.08344
65	0.5000	−0.20581	−0.11477	−0.06080	−0.04272	−0.03039	−0.03800	0.03111
59	0.4531	−0.21090	−0.17119	−0.10648	−0.86636	−0.07404	−0.07503	−0.07540
37	0.2813	−0.15662	−0.32726	−0.27805	−0.24427	−0.22855	−0.23176	−0.23186
23	0.1719	−0.10150	−0.24299	−0.38289	−0.34323	−0.33050	−0.32393	−0.32709
14	0.1016	−0.06434	−0.14612	−0.29730	−0.41933	−0.40435	−0.38324	−0.38000
10	0.0703	−0.04775	−0.10338	−0.22220	−0.37827	−0.43643	−0.43025	−0.41657
9	0.0625	−0.04192	−0.09266	−0.20196	−0.35344	−0.42901	−0.43590	−0.42537
8	0.0547	−0.03717	−0.08186	−0.18109	−0.32407	−0.41165	−0.43154	−0.42735
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

شکل ۵ – مقادیر u برای رینولدز های ۱۰۰ و ۴۰۰ در مقاله قیا

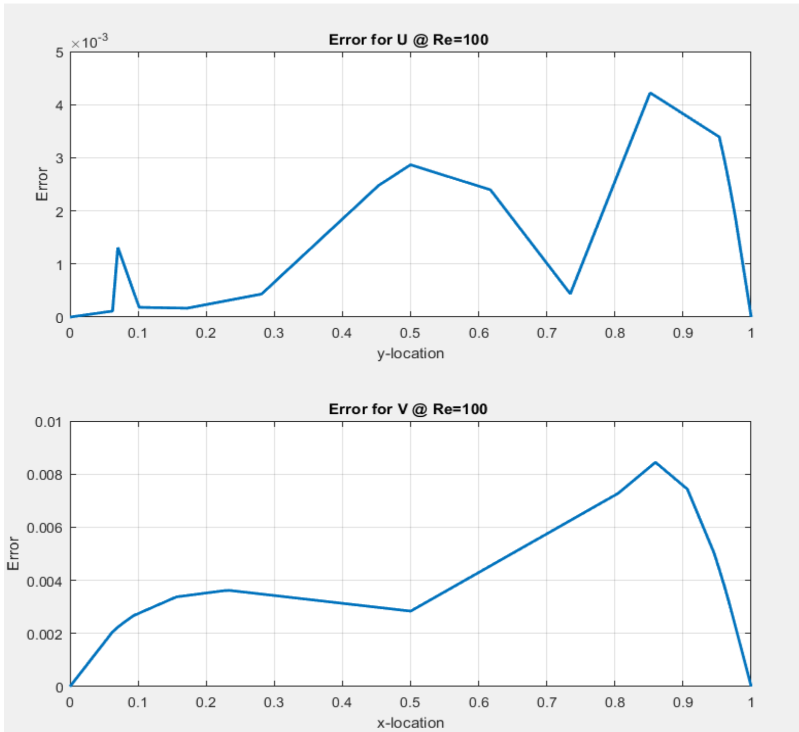
TABLE II

Results for *v*-Velocity along Horizontal Line through Geometric Center of Cavity

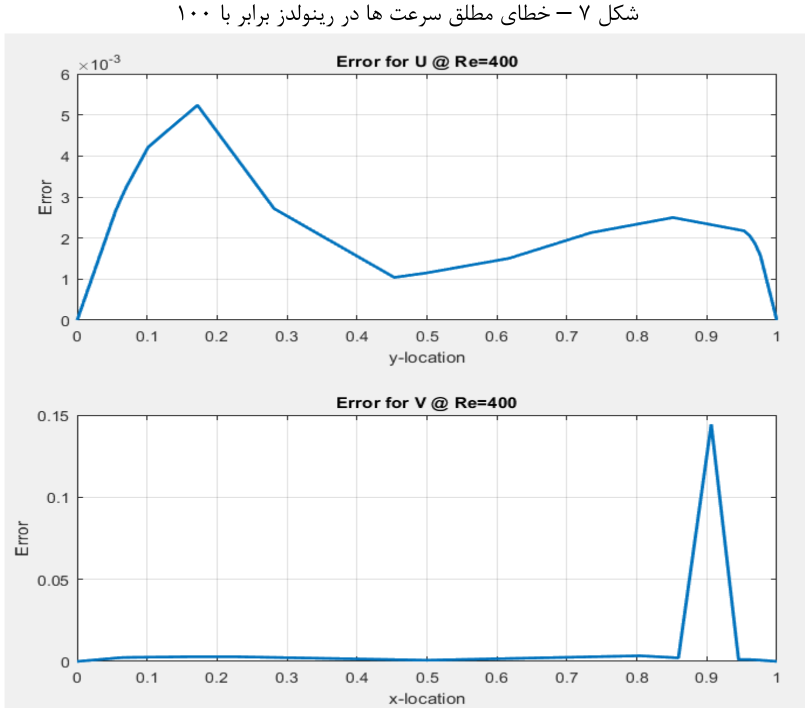
129-grid pt. no.	x	Re						
		100	400	1000	3200	5000	7500	10,000
129	1.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
125	0.9688	−0.05906	−0.12146	−0.21388	−0.39017	−0.49774	−0.53858	−0.54302
124	0.9609	−0.07391	−0.15663	−0.27669	−0.47425	−0.55069	−0.55216	−0.52987
123	0.9531	−0.08864	−0.19254	−0.33714	−0.52357	−0.55408	−0.52347	−0.49099
122	0.9453	−0.10313	−0.22847	−0.39188	−0.54053	−0.52876	−0.48590	−0.45863
117	0.9063	−0.16914	−0.23827	−0.51550	−0.44307	−0.41442	−0.41050	−0.41496
111	0.8594	−0.22445	−0.44993	−0.42665	−0.37401	−0.36214	−0.36213	−0.36737
104	0.8047	−0.24533	−0.38598	−0.31966	−0.31184	−0.30018	−0.30448	−0.30719
65	0.5000	0.05454	0.05186	0.02526	0.00999	0.00945	0.00824	0.00831
31	0.2344	0.17527	0.30174	0.32235	0.28188	0.27280	0.27348	0.27224
30	0.2266	0.17507	0.30203	0.33075	0.29030	0.28066	0.28117	0.28003
21	0.1563	0.16077	0.28124	0.37095	0.37119	0.35368	0.35060	0.35070
13	0.0938	0.12317	0.22965	0.32627	0.42768	0.42951	0.41824	0.41487
11	0.0781	0.10890	0.20920	0.30353	0.41906	0.43648	0.43564	0.43124
10	0.0703	0.10091	0.19713	0.29012	0.40917	0.43329	0.44030	0.43733
9	0.0625	0.09233	0.18360	0.27485	0.39560	0.42447	0.43979	0.43983
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

شکل ۶ – مقادیر v برای رینولدز های ۱۰۰ و ۴۰۰ در مقاله قیا

در نهایت مقایسه ی سرعت های به دست آمده و سرعت های مقاله ی قیا در رینولدز های ۱۰۰ و ۴۰۰ با تعداد نود ۱۲۹ و سرعت دیواره برابر با ۱ متر برثانیه به صورت زیر است :



شکل ۷ – خطای مطلق سرعت ها در رینولدز برابر با ۱۰۰



شکل ۸ – خطای مطلق سرعت ها در رینولدز برابر با ۴۰۰

بخش دوم - اصلاح کد به روش SOR

شرح معادلات SOR

برای استخراج گسسته سازی برای اجرای کد مورد نظر نیاز به پیاده سازی روش upwind و SOR روی معادله انتقال ورتیسیته به طور همزمان می باشد.

جملات جابجایی در معادله زیر باید به گونه ای بازنویسی شوند که با توجه به جهت حرکت موج رونده، داده های گسسته سازی شده از بالادست برداشته شوند.

$$\frac{\partial\Omega}{\partial t}+u\frac{\partial\Omega}{\partial x}+v\frac{\partial\Omega}{\partial y}=\frac{\mu}{\rho}\left(\frac{\partial^2\Omega}{\partial x^2}+\frac{\partial^2\Omega}{\partial y^2}\right)$$

$$u\frac{\partial\Omega}{\partial x}=\max(u_{i,j},0)\frac{\Omega_{i,j}^n-\Omega_{i-1,j}^n}{\Delta x}-\max(-u_{i,j},0)\frac{\Omega_{i+1,j}^n-\Omega_{i,j}^n}{\Delta x}$$

$$v\frac{\partial\Omega}{\partial y}=\max(v_{i,j},0)\frac{\Omega_{i,j}^n-\Omega_{i,j-1}^n}{\Delta y}-\max(-v_{i,j},0)\frac{\Omega_{i,j+1}^n-\Omega_{i,j}^n}{\Delta y}$$

همچنین برای روش تکراری SOR ضریبی به اسم ضریب over relaxation(W) تعریف می شود که تعریف این روش به شرح زیر می باشد:

$$\Omega_{i,j}^{n+1}=\Omega_{i,j}^n+\frac{\omega}{a_{i,j}}\left(Q_{i,j}-a_{i+1,j}\Omega_{i+1,j}^n-a_{i-1,j}\Omega_{i-1,j}^{n+1}-a_{i,j+1}\Omega_{i,j+1}^n-a_{i,j-1}\Omega_{i,j-1}^{n+1}-a_{i,j}\Omega_{i,j}^n\right)$$

$$\Omega_{i,j}^{n+1}=(1-\omega)\Omega_{i,j}^n+\frac{\omega}{a_{i,j}}\left(Q_{i,j}-a_{i+1,j}\Omega_{i+1,j}^n-a_{i-1,j}\Omega_{i-1,j}^{n+1}-a_{i,j+1}\Omega_{i,j+1}^n-a_{i,j-1}\Omega_{i,j-1}^{n+1}\right)$$

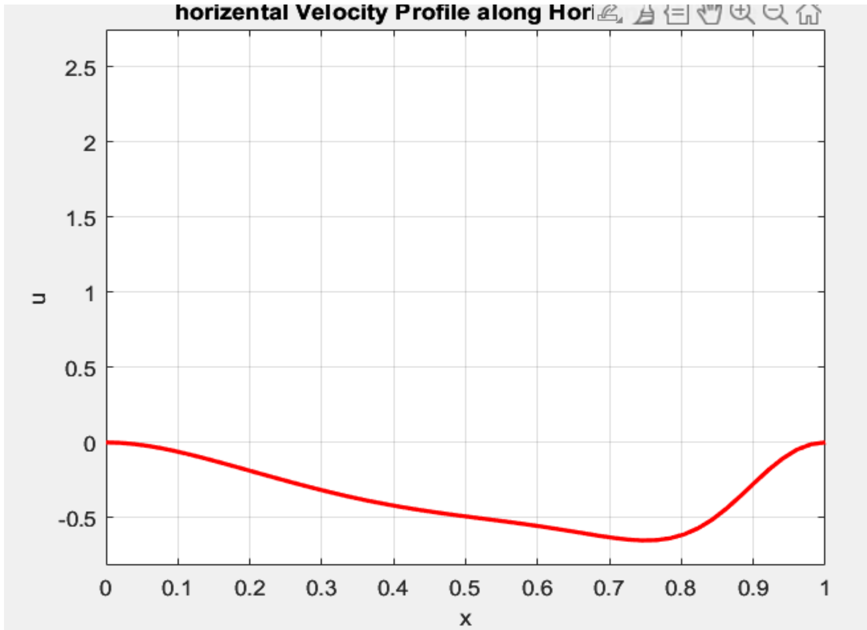
و در نهایت کد با $\omega = 1.5$ به فرم زیر حاصل می شود :

```
OMEGA = 1.5;
Vo(i, j) =Vop(i, j) + (OMEGA ./ (1 - 4*dt*(mu/(rho*h^2) - dt*(s_x1 ./ h) -...
           dt*(s_x2 ./ h) - dt*(s_y1 ./ h) - dt*(s_y2 ./ h))) .* ((mu/(rho*h^2)) *...
           (Vop(ip, j) + Vop(im, j) + Vop(i, jp) + Vop(i, jm) - 4 * Vop(i, j)) - ...
           (s_x1 ./ h) .* (Vop(i, j) - Vop(im, j)) + (s_x2 ./ h ).* (Vop(ip, j) - Vop(i, j)) - ...
           (s_y1 ./ h) .* (Vop(i, j) - Vop(i, jm)) + (s_y2 ./ h) .* (Vop(i, jp) - Vop(i, j))) * dt);
```

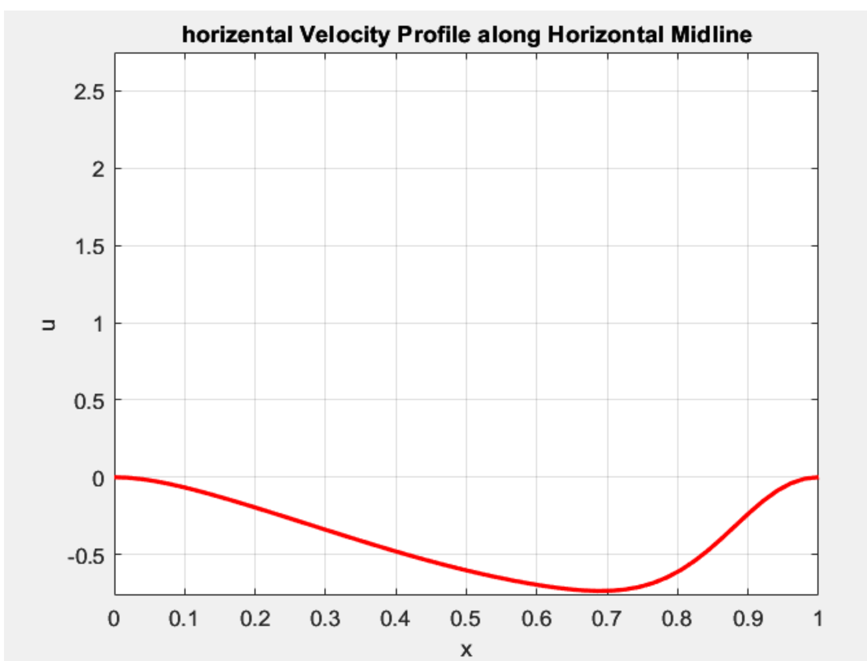
پروفیل سرعت افقی بر روی خط عمودی و افقی میانی حفره

در این قسمت با تعداد نودهای ۵۲ کد عددی اصلاح شده با روش بالادست و SOR با یکدیگر مقایسه شده اند.

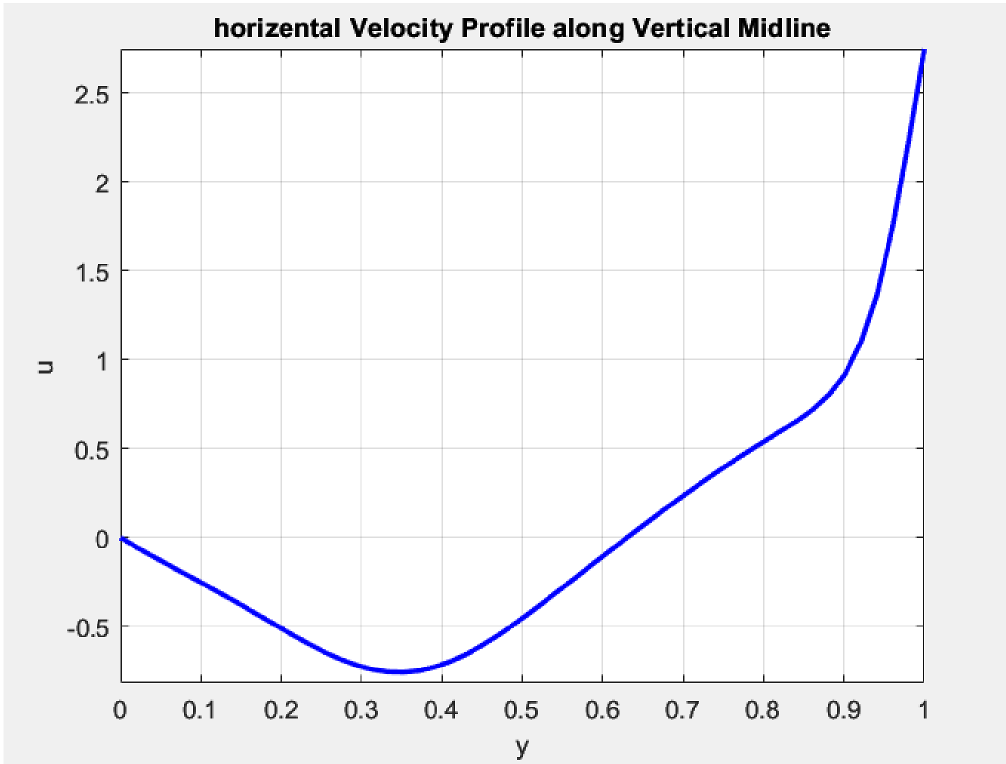
سرعت دیواره ۲٫۷۵ متر بر ثانیه می باشد و نمودارهای مربوطه در زیر اورده شده است. لازم به ذکر است هر دو کد در حالت گذرا گسسته سازی شده اند.



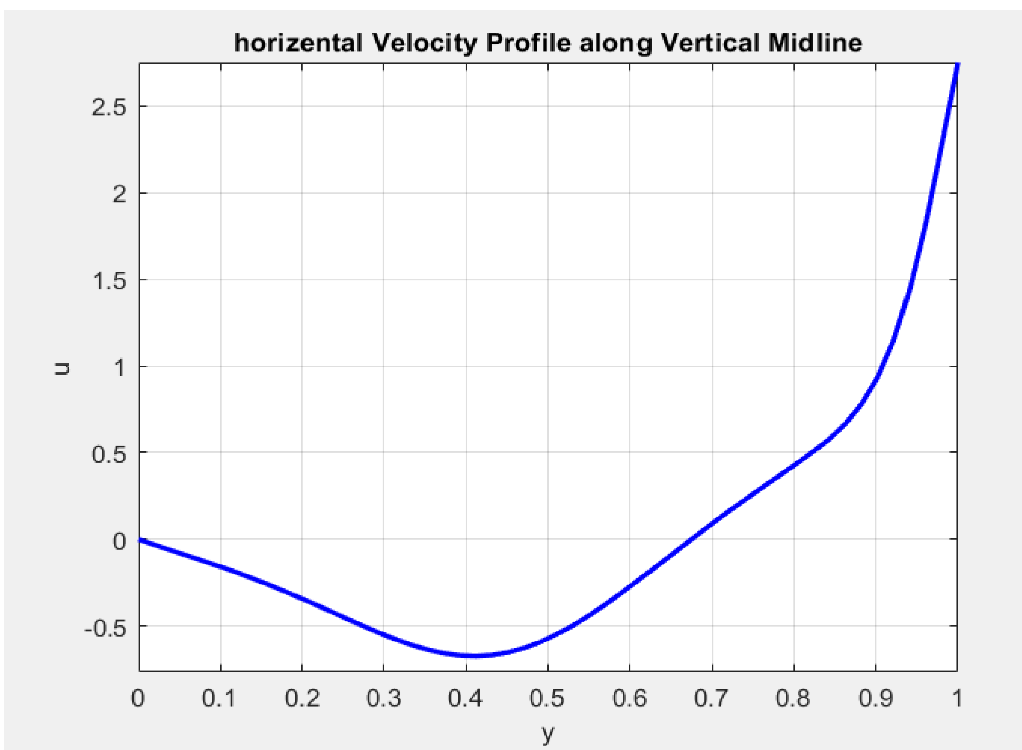
شکل ۹ – سرعت در راستای افقی روی خط افقی میانی حفره در روش ملوار



شکل ۱۰ – سرعت در راستای افقی روی خط افقی میانی حفره در روش SOR

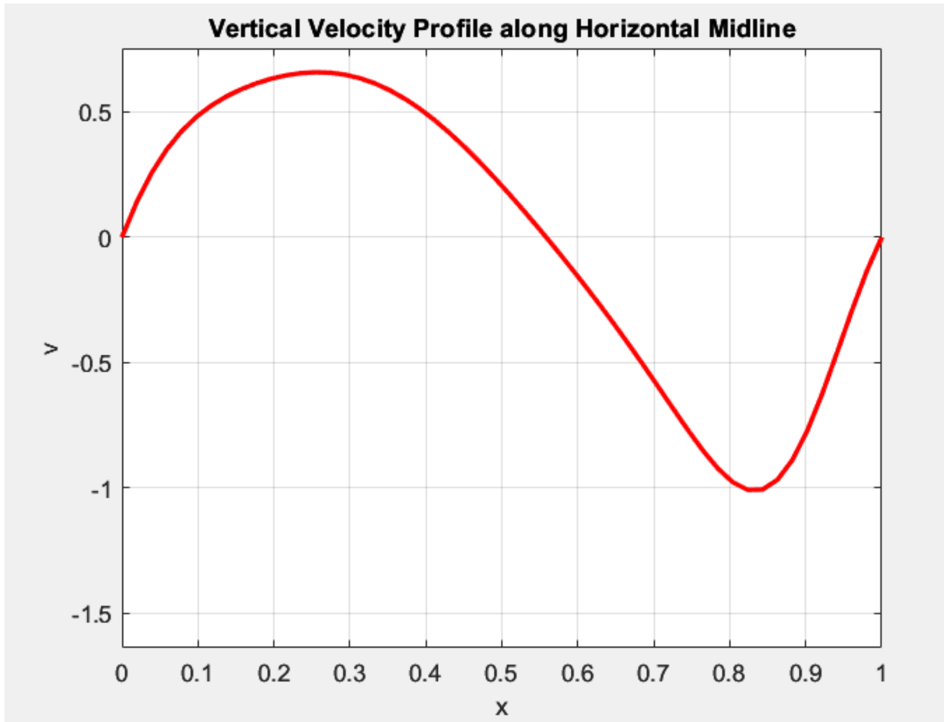


شکل ۱۱ – سرعت در راستای افقی روی خط عمودی میانی حفره در روش ملوار

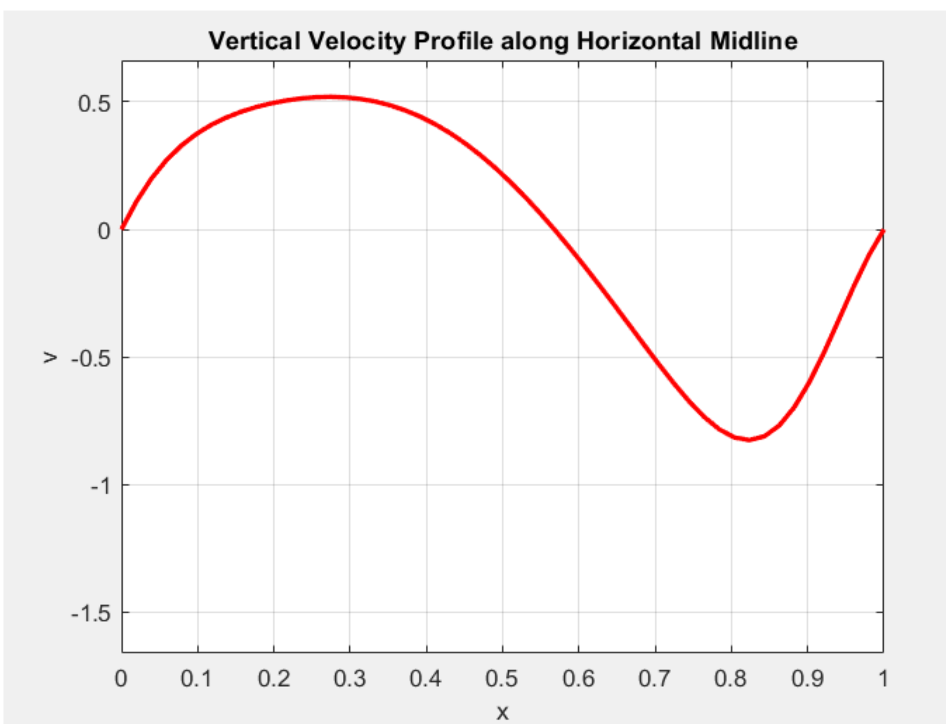


شکل ۱۲ – سرعت در راستای افقی روی خط عمودی میانی حفره در روش SOR

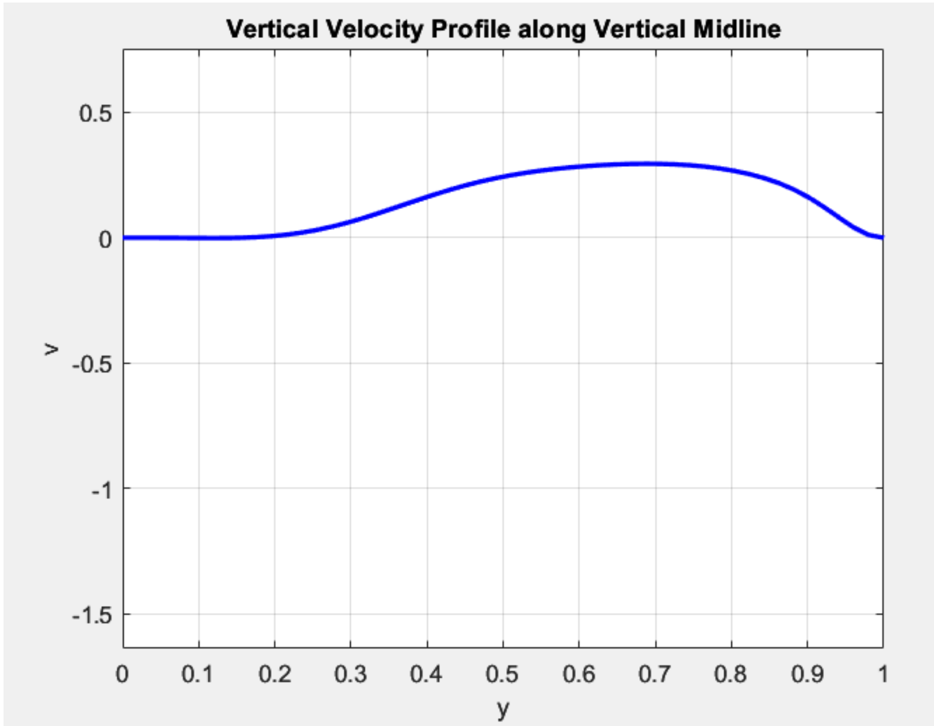
پروفیل سرعت عمودی بر روی خط عمودی و افقی میانی حفره



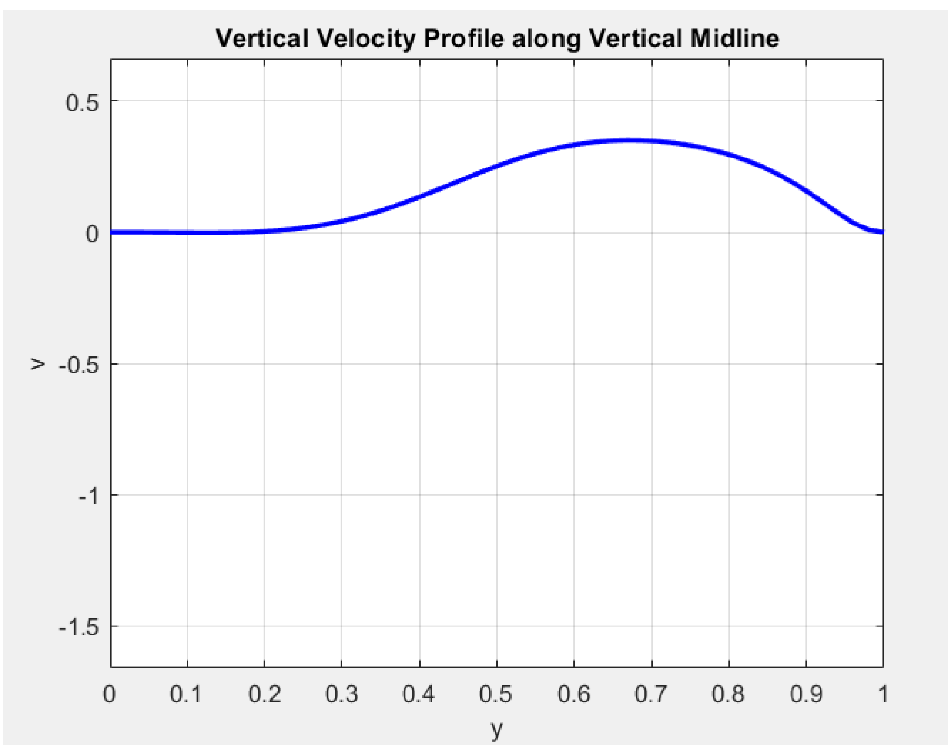
شکل ۱۳ – سرعت در راستای عمودی روی خط افقی میانی حفره در روش ملوار



شکل ۱۴ – سرعت در راستای عمودی روی خط افقی میانی حفره در روش SOR



شکل ۱۵ – سرعت در راستای عمودی روی خط عمودی میانی حفره در کد ملوار



شکل ۱۶ – سرعت در راستای افقی روی خط عمودی میانی حفره در روش SOR

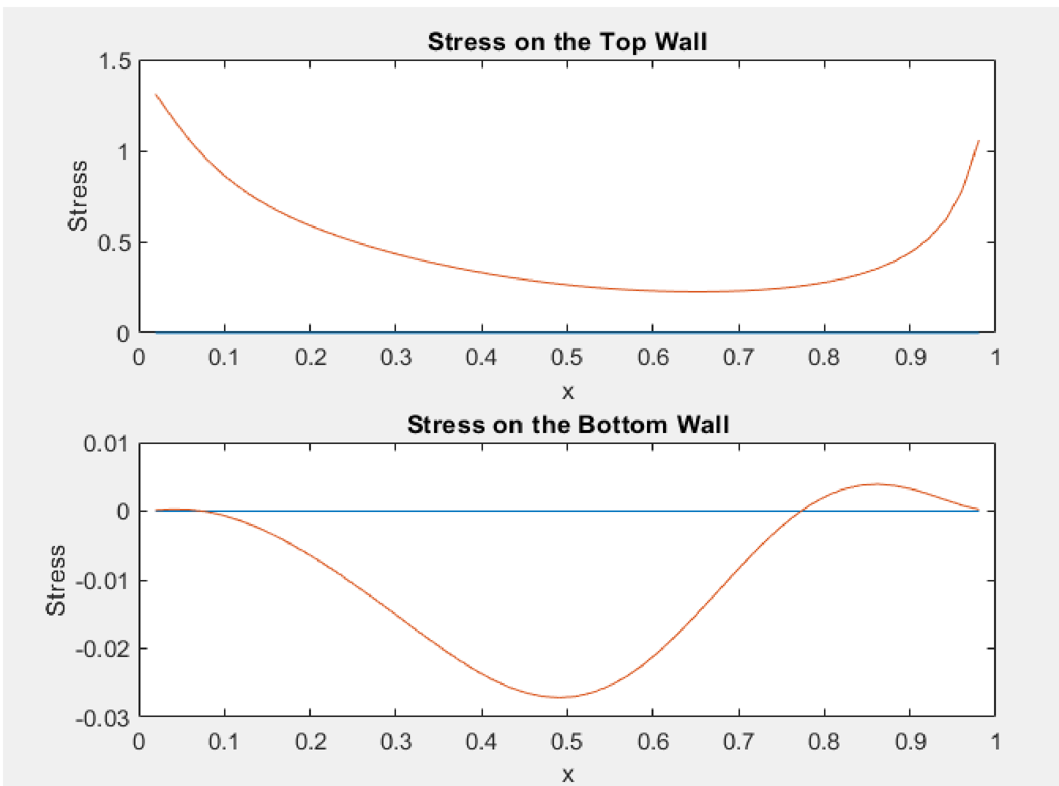
تنش برشی بر روی دیواره ی بالایی و پایینی

برای محاسبه ی تنش های برش روی دیواره های بالا و پایین به شکل زیر عمل میکنیم:

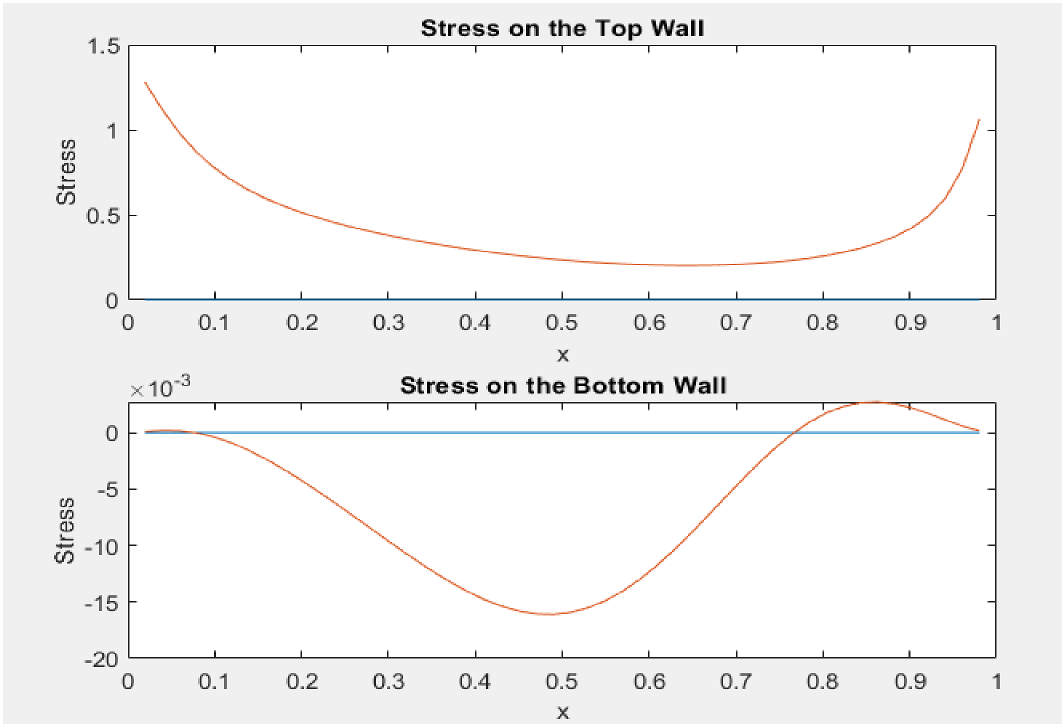
$$\tau = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})$$

تنش برشی در دیواره ی پایینی و بالایی در کد :

```
stress_top(2,:) = mu*((u(i,Ny)-u(i,Ny-1))/h + (v(ip,Ny)-v(i,Ny))/h)
stress_bottom(2,:) = mu*((u(i,2)-u(i,1))/h + (v(ip,1)-v(i,1))/h)
```



شکل ۱۷ – تنش برشی روی دیواره ی بالایی و پایینی در کد ملوار

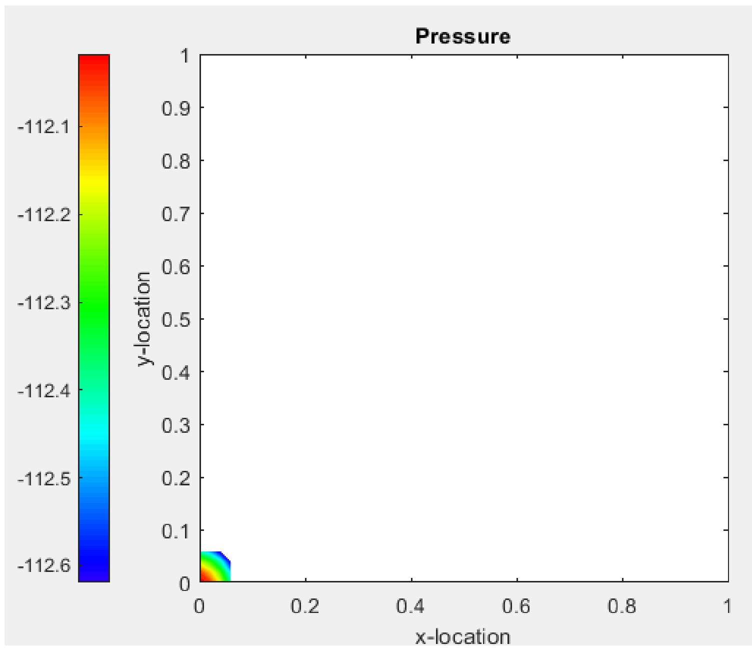


شکل ۱۸ – تنش برشی روی دیواره ی بالایی و پایینی در در روش SOR

بخش سوم - مقایسه دو روش

کد ملوار

با تغییر عدد رینولدز و سرعت دیواره بالایی در کد ملوار از رینولدز برابر با ۱۳۲۶ و سرعت دیواره برابر با ۱۳٫۲۶ متر بر ثانیه به بعد نمودار ها دچار اختلال می شوند و نتایجی ارائه نمی دهند.



شکل ۱۹ – نمونه کانتور فشار ناصحیح در رینولدز برابر با ۱۳۲۷

روش SOR

در کدمتلب با روش SOR نیز مشاهده می شود با گذر از عددرینولدز برابر با ۱۷۲۵ و سرعت دیواره بالایی برابر با ۱۷٫۲۵ متر بر ثانیه، نمودار ها دچار اختلال می شوند.

Under relaxation

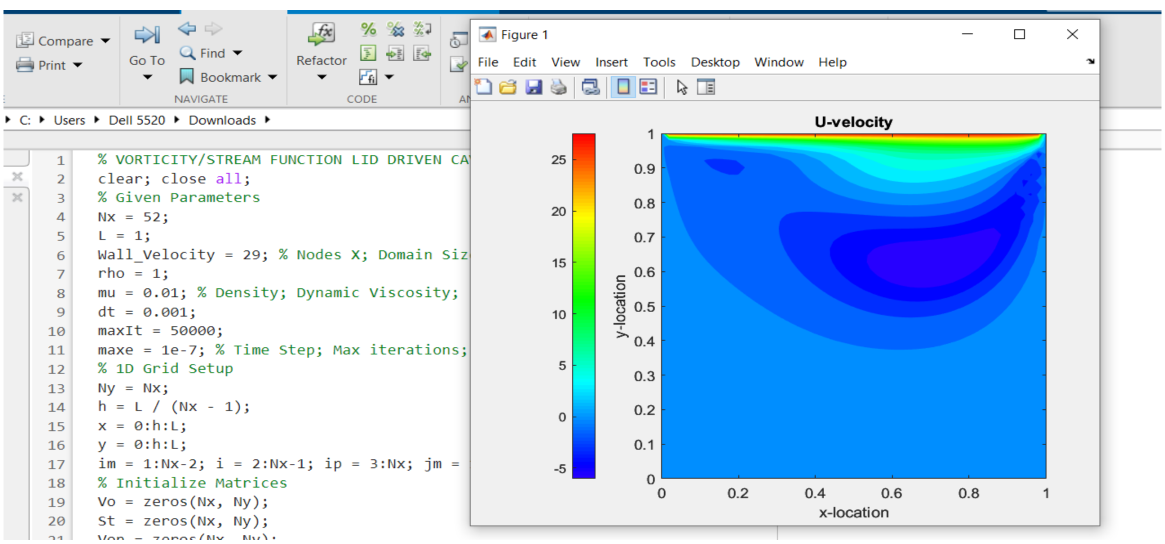
در این بخش از خواسته ی سوال، ضریب under relaxation بکار برده می شود که به همگرایی بهتر کمک میکند و مقداری بین صفر و یک دارد و به شکل زیر تعریف می شود:

$$\psi_{i,j}^{(k+1)} = \psi_{i,j}^{(k)} + W_s\left(\psi_{i,j}^{*} - \psi_{i,j}^{(k)}\right)$$
$$\Omega_{i,j}^{(k+1)} = \Omega_{i,j}^{(k)} + W_v\left(\Omega_{i,j}^{*} - \Omega_{i,j}^{(k)}\right)$$
$$0 < W_s < 1;\ 0 < W_v < 1$$

شکل ۲۰ – نحوه کارکرد ضریب under relaxation

```
OMEGA = 1.5;
w=0.8;
Vo(i, j) =Vop(i, j) +w*( (OMEGA ./ (1 - 4*dt*(mu/(rho*h^2) - dt*(s_x1 ./ h) -...
           dt*(s_x2 ./ h) - dt*(s_y1 ./ h) - dt*(s_y2 ./ h))) .* ((mu/(rho*h^2)) * (Vop(ip, j) +...
           Vop(im, j) + Vop(i, jp) + Vop(i, jm) - 4 * Vop(i, j)) - (s_x1 ./ h) .* ...
          (Vop(i, j) - Vop(im, j)) + (s_x2 ./ h ).* (Vop(ip, j) - Vop(i, j)) - ...
          (s_y1 ./ h) .* (Vop(i, j) - Vop(i, jm)) + (s_y2 ./ h) .* (Vop(i, jp) - Vop(i, j))) * dt));
```

همانطور که در شکل زیر پیداست با اضافه کردن ضریب under relaxation عدد رینولدز تا عدد ۲۹۰۰۰ نیز افزوه شده است اما در رینولدزهای بالاتر از ۲۹۲۵۰ دیگر نمودارها رسم نمی شوند .پس واضح است که این ضریب بازه ی سرعت دیواره ی بالا را افزایش داده است.



شکل ۲۱ – نمونه ای از استفاده از ضریب under relaxation با رینولدز ۲۹۰۰۰

بخش چهارم - معادله حاکم بر سیال دو بعدی

شرح معادلات دو بعدی

در این قسمت معادله حاکم بر انتقال ورتیسیته به شکل دائمی حل می شود در نتیجه در کد قسمت SOR گسسته سازی زمانی در نظر گرفته نمی شود و معادله به شکل زیر با روش گوس سایدل حل شده است :

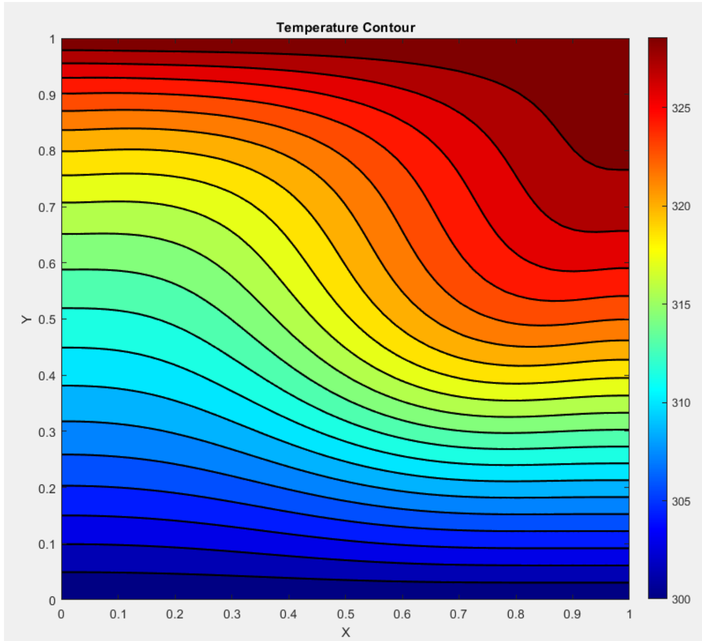
$$u\frac{\partial\Omega}{\partial x}+v\frac{\partial\Omega}{\partial y}=\frac{\mu}{\rho}\left(\frac{\partial^2\Omega}{\partial x^2}+\frac{\partial^2\Omega}{\partial y^2}\right)$$

همچنین معادلات انرژی با این معادلات حل می شود، به این صورت که سرعت های (u و v) محاسبه شده از حل معادلات پواسون انتقال ورتیسیته و تابع جربان، در معادله دما جاگذاری می شود و دما در مش مشخص شده محاسبه می گردد.

$$u\frac{\partial \mathrm{T}}{\partial x}+v\frac{\partial \mathrm{T}}{\partial y}=\frac{\mu}{\rho}\left(\frac{\partial^2\mathrm{T}}{\partial x^2}+\frac{\partial^2\mathrm{T}}{\partial y^2}\right)$$

نحوه حل شدن معادله انرژی همانند انتقال ورتیسیته است که به شکل دائمی و با روش تفاضل بالادست (upwinding) حل می شود. شرایط مرزی نیز برای دیوارهای اطراف با مشتق یک طرفه (نیومن) و دریشله در کد مشخص شده است.

درشکل زیر برای سرعت دیواره بالایی ۱۰ متربر ثانیه کانتور دمایی رسم شده است همانطور که در شکل مشاهده می شود بر روی دیواره های جانبی شیب ثابت است که نشان دهنده ی شرط مرزی نیومن صفر می باشد و در دیواره های بالایی و پایینی دمای ثابت مشهود است.



شکل ۲۲ – کانتور دما بر روی محفظه

بر اساس کانتور دمایی که از حل معادله انرژی به دست آمده است، توزیع عدد ناسلت (Nu) در دیواره بالایی و پایینی می‌تواند اطلاعات مهمی را درباره انتقال حرارت در این نواحی ارائه دهد. برای این منظور، ابتدا باید تعریف عدد ناسلت را بررسی کنیم. عدد ناسلت بیانگر نسبت انتقال حرارت همرفتی به انتقال حرارت هدایتی در یک سیال است.

- **دیواره بالایی(x=L)** دمای دیواره بالایی بالاتر است (330 K) و از آنجا که گرادیان دما به طور معمول در این ناحیه بالا خواهد بود، می‌توان نتیجه گرفت که انتقال حرارت در این ناحیه بیشتر است. بنابراین عدد ناسلت در این ناحیه احتمالاً بیشتر خواهد بود.
- **دیواره پایینی (x=0)** :دمای دیواره پایینی کمتر است (300 K) و گرادیان دما در این ناحیه نیز می‌تواند کمتر باشد. بنابراین عدد ناسلت در این ناحیه احتمالاً کمتر از دیواره بالایی خواهد بود.

بنابراین، به طور کلی می‌توان گفت که عدد ناسلت در دیواره بالایی به دلیل گرادیان دمایی بیشتر، بالاتر است و انتقال حرارت همرفتی در این ناحیه بیشتر خواهد بود. این نتیجه با مشاهده کانتور دمایی و توزیع دما در دیواره‌ها هماهنگ است.

کد های متلب

کد ملوار اصلاح شده

```
%% VORTICITY/STREAM FUNCTION LID DRIVEN CAVITY FLOW SOLVER JOE MOLVAR
clear;
clc;
close all
%%% GIVENS
Nx = 52; %Nodes
L = 1; %Domain Size
rho = 1;% Density
mu = 0.01; %Dynamic Viscosity
Re=100; %Reynolds
Wall_Velocity =1; %Velocity
dt = 0.001;%Time Step
maxIt = 50000; %Max iter
maxe = 1e-7; %Max error
%%% SETUP 1D GRID
Ny = Nx; h=L/(Nx-1); x = 0:h:L; y = 0:h:L;
im = 1:Nx-2; i = 2:Nx-1; ip = 3:Nx; jm = 1:Ny-2; j = 2:Ny-1; jp = 3:Ny;
I=1:Nx; J=1:Ny;
%%% PRELOCATE MATRIXES
Vo = zeros(Nx,Ny); St = Vo; Stp = Vo; Vop = Vo; u = Vo; v = Vo; P = Vo;
epsilon = 0;
%%% VELOCITY ON THE UPPER WALL(NO SLIP CONDITION)
u(2:Nx-1,Ny) = Wall_Velocity;
%%% SOLVE LOOP SIMILAR TO GAUSS-SIEDEL METHOD
for iter = 1:maxIt
%%% CREATE BOUNDARY CONDITIONS
Vo(1:Nx,Ny) = -2*St(1:Nx,Ny-1)/(h^2) - Wall_Velocity*2/h; % Top
Vo(1:Nx,1) = -2*St(1:Nx,2) /(h^2); % Bottom
Vo(1,1:Ny) = -2*St(2,1:Ny) /(h^2); % Left
Vo(Nx,1:Ny) = -2*St(Nx-1,1:Ny)/(h^2); % Right
%%% PARTIALLY SOLVE VORTICITY TRANSPORT EQUATION
Vop = Vo;
Stp = St;
Vo(i,j) = Vop(i,j) + ...
(-1*(St(i,jp)-St(i,jm))/(2*h) .* (Vop(ip,i)-Vop(im,j))/(2*h)+...
(St(ip,j)-St(im,j))/(2*h) .* (Vop(i,jp)-Vop(i,jm))/(2*h)+...
mu/rho*(Vop(ip,j)+Vop(im,j)-4*Vop(i,j)+Vop(i,jp)+Vop(i,jm))/(h^2))*dt;
%%% PARTIALLY SOLVE ELLIPTICAL VORTICITY EQUATION FOR STREAM FUNCTION
St(i,j) = (Vo(i,j)*h^2 + St(ip,j) + St(i,jp) + St(i,jm) + St(im,j))/4;
%%% CREATE VELOCITY FROM STREAM FUNCTION
u(i,j) = (St(i,jp)-St(i,jm))/(2*h); v(i,j) = (-St(ip,j)+St(im,j))/(2*h);
%%% PRESSURE TOP BOUNDARY CONDITION
P(I,Ny) = 1/3*(4*P(I,Ny-1)-P(I,Ny-2))+(2*mu)/(3*h)*(-5*v(I,Ny-1)+4*v(I,Ny-2)-
v(I,Ny-3));
%%% PRESSURE BOTTOM BOUNDARY CONDITION
P(I,1) = 1/3*(4*P(I,2)-P(I,3))-(2*mu)/(3*h)*(-5*v(I,2)+4*v(I,3)-v(I,4));
%%% PRESSURE RIGHT BOUNDARY CONDITION
P(Nx,J) = 1/3*(4*P(Nx-1,J)-P(Nx-2,J))+(2*mu)/(3*h)*(-5*u(Nx-1,J)+4*u(Nx-2,J)-
u(Nx-3,J));
%%% PRESSURE LEFT BOUNDARY CONDITION
```

```
P(1,J) = 1/3*(4*P(2,J)-P(3,J))-(2*mu)/(3*h)*(-5*u(2,J)+4*u(3,J)-u(4,J));
%%% PRESSURE FOR INNER NODES
P(i,j) = 0.25*(P(ip,j)+P(im,j)+P(i,jp)+P(i,jm))-rho/2*(1/(h^2)*(St(ip,j)-
2*St(i,j)+....
St(im,j)).*(St(i,jp)-2*St(i,j)+St(i,jm))-1/(16*h^2)*(St(ip,jp)-St(ip,jm)-
St(im,jp)+St(im,jm)).^2);
%%% CALCULATING EPSILON
epsilon(1,iter) = max(abs(St-Stp),[],'all');
%%% CHECK FOR CONVERGENCE
if iter > 10
error = max(max(Vo - Vop));
if error < maxe
break;
end
end
end
%%% PLOTS
cm = hsv(ceil(100/0.7)); cm = flipud(cm(1:100,:));
figure(1); contourf(x,y,u',23,'LineColor','none');
title('U-velocity'); xlabel('x-location'); ylabel('y-location')
axis('equal',[0 L 0 L]); colormap(cm); colorbar('westoutside');
figure(2); plot(y,u(round(Ny/2),:));
title('Centerline x-direction velocity');
xlabel('y/L'); ylabel('u/U'); axis('square'); xlim([0 L]); grid on
N = 1000; xstart = max(x)*rand(N,1); ystart = max(y)*rand(N,1);
[X,Y] = meshgrid(x,y);
figure(3); h=streamline(X,Y,u',v',xstart,ystart,[0.1, 200]);
title('Stream Function'); xlabel('x-location'); ylabel('y-location')
axis('equal',[0 L 0 L]); set(h,'color','k')
figure(4);
contourf(x,y,P',20,'LineColor','none');
title('Pressure'); xlabel('x-location'); ylabel('y-location');
axis('equal',[0 L 0 L]); colormap(cm); colorbar('westoutside');
figure(5);
subplot(2,2,1);
plot(x,Vo(:,1),'LineWidth',2);
title('Vorticity on the bottom wall'); xlabel('x-location');
ylabel('Vorticity'); ylim([-80 10]);
grid on
subplot(2,2,2);
plot(x,Vo(:,Ny),'LineWidth',2);
title('Vorticity on the top wall'); xlabel('x-location');
ylabel('Vorticity'); ylim([-80 10]);
grid on
subplot(2,2,3);
plot(y,Vo(1,:),'LineWidth',2);
title('Vorticity on the left wall'); xlabel('y-location');
ylabel('Vorticity'); ylim([-2 40]);
grid on
subplot(2,2,4);
plot(y,Vo(Nx,:),'LineWidth',2);
title('Vorticity on the right wall'); xlabel('y-location');
ylabel('Vorticity'); ylim([-2 40]);
grid on
figure(6);
contourf(x,y,Vop',20,'LineColor','none');
```

```
title('Vorticity'); xlabel('x-location'); ylabel('y-location');
axis('equal',[0 L 0 L]); colormap(cm); colorbar('westoutside');
```

کد SOR

```
% VORTICITY/STREAM FUNCTION LID DRIVEN CAVITY FLOW SOLVER BY JOE MOLVAR

clear; close all;

% Given Parameters

Nx = 52;

L = 1;

Wall_Velocity = 2.75; % Nodes X; Domain Size; Wall Velocity

rho = 1;

mu = 0.01; % Density; Dynamic Viscosity;

dt = 0.001;

maxIt = 50000;

maxe = 1e-7; % Time Step; Max iterations; Max error

% 1D Grid Setup

Ny = Nx;

h = L / (Nx - 1);

x = 0:h:L;

y = 0:h:L;

im = 1:Nx-2; i = 2:Nx-1; ip = 3:Nx; jm = 1:Ny-2; j = 2:Ny-1; jp = 3:Ny;

% Initialize Matrices

Vo = zeros(Nx, Ny);

St = zeros(Nx, Ny);

Vop = zeros(Nx, Ny);

u = zeros(Nx, Ny);

v = zeros(Nx, Ny);

u(2:Nx-1, Ny) = Wall_Velocity;

Vop = Vo;

% Solve Loop (Similar to Gauss-Siedel Method)

for iter = 1:maxIt

    % Boundary Conditions

    Vo(:, Ny) = -2 * St(:, Ny-1) / (h^2) - Wall_Velocity * 2 / h; % Top
```

```
    Vo(:, 1) = -2 * St(:, 2) / (h^2); % Bottom
    Vo(1, :) = -2 * St(2, :) / (h^2); % Left
    Vo(Nx, :) = -2 * St(Nx-1, :) / (h^2); % Right
    % Partially Solve Vorticity Transport Equation
    s_x1 = max(u(i, j), 0);
    s_x2 = max(-u(i, j), 0);
    s_y1 = max(v(i, j), 0);
    s_y2 = max(-v(i, j), 0);
    % SOR Calculation
    OMEGA = 1.5;
    Vo(i, j) =Vop(i, j) + (OMEGA ./ (1 - 4*dt*(mu/(rho*h^2) - dt*(s_x1 ./ h)
-...
                  dt*(s_x2 ./ h) - dt*(s_y1 ./ h) - dt*(s_y2 ./ h))) .*
((mu/(rho*h^2)) * (Vop(ip, j) + Vop(im, j) + Vop(i, jp) + Vop(i, jm) - 4 *
Vop(i, j)) - ...
        (s_x1 ./ h) .* (Vop(i, j) - Vop(im, j)) + (s_x2 ./ h ).* (Vop(ip, j)
- Vop(i, j)) - ...
         (s_y1 ./ h) .* (Vop(i, j) - Vop(i, jm)) + (s_y2 ./ h) .* (Vop(i, jp)
- Vop(i, j))) * dt);
    % Partially Solve Elliptical Vorticity Equation for Stream Function
    St(i, j) = (Vo(i, j) * h^2 + St(ip, j) + St(i, jp) + St(i, jm) + St(im,
j)) / 4;
    u(i, j) = (St(i, jp) - St(i, jm)) / (2 * h);
    v(i, j) = (-St(ip, j) + St(im, j)) / (2 * h);
    % Check for Convergence
    if iter > 10
        error = max(max(abs(Vo - Vop)));
        if error < maxe
            break;
        end
    end
    Vop = Vo;
end
% Plots
```

```
cm = hsv(ceil(100 / 0.7));
cm = flipud(cm(1:100, :));
figure(1);
contourf(x, y, u', 23, 'LineColor', 'none');
title('U-velocity');
xlabel('x-location');
ylabel('y-location');
% contourf(x, y, v', 23, 'LineColor', 'none');
% title('V-velocity');
% xlabel('x-location');
% ylabel('y-location');
axis equal;
axis([0 L 0 L]);
colormap(cm);
colorbar('westoutside');
figure(2);
plot(y, u(round(Ny/2), :));
title('Centerline x-direction velocity');
xlabel('y/L');
ylabel('u/U');
axis square;
xlim([0 L]);
grid on;
N = 1000;
xstart = L * rand(N, 1);
ystart = L * rand(N, 1);
[X, Y] = meshgrid(x, y);
figure(3);
h = streamline(X, Y, u', v', xstart, ystart, [0.1, 200]);
title('Stream Function');
xlabel('x-location');
ylabel('y-location');
```

```
axis equal;
set(h, 'color', 'k');
% a and b
%Find indices for the middle vertical and horizontal lines

%Define the midpoints for vertical and horizontal midlines
midpoint_x = round(Nx / 2);
midpoint_y = round(Ny / 2);

% Extract the vertical velocity profile (v) along the midlines
v_vertical_midline = v(midpoint_x, :);
v_horizontal_midline = v(:, midpoint_y);

% Plot the vertical velocity profile along the vertical midline
figure;
plot(y, v_vertical_midline, 'b-', 'LineWidth', 2);
xlabel('y');
ylabel('v');
title('Vertical Velocity Profile along Vertical Midline');
grid on;
ylim([min(v(:)), max(v(:))]);

% Plot the vertical velocity profile along the horizontal midline
figure;
plot(x, v_horizontal_midline, 'r-', 'LineWidth', 2);
xlabel('x');
ylabel('v');
title('Vertical Velocity Profile along Horizontal Midline');
grid on;
ylim([min(v(:)), max(v(:))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
% Extract the vertical velocity profile (v) along the midlines
```

```
u_vertical_midline = u(midpoint_x, :);
u_horizontal_midline = u(:, midpoint_y);

% Plot the vertical velocity profile along the vertical midline
figure;
plot(y, u_vertical_midline, 'b-', 'LineWidth', 2);
xlabel('y');
ylabel('u');
title('horizental Velocity Profile along Vertical Midline');
grid on;
ylim([min(u(:)), max(u(:))]);

% Plot the vertical velocity profile along the horizontal midline
figure;
plot(x, u_horizontal_midline, 'r-', 'LineWidth', 2);
xlabel('x');
ylabel('u');
title('horizental Velocity Profile along Horizontal Midline');
grid on;
ylim([min(u(:)), max(u(:))]);
%**********************part c
 stress_top(2,:) = mu*((u(i,Ny)-u(i,Ny-1))/h + (v(ip,Ny)-v(i,Ny))/h)
 stress_bottom(2,:) = mu*((u(i,2)-u(i,1))/h + (v(ip,1)-v(i,1))/h)

%**********************part c
%  stress_top(2,:) = mu*((u(i,Ny)-u(i,Ny-1))/h + (v(ip,Ny)-v(i,Ny))/h)
%  stress_bottom(2,:) = mu*((u(i,2)-u(i,1))/h + (v(ip,1)-v(i,1))/h)
%  %%% PLOT STRESS
% figure;
% subplot(2,1,1);
% plot(x(2:end-1), stress_top);
% title('Stress on the Top Wall');
```

```
% xlabel('x');
% ylabel('Stress');
%
% subplot(2,1,2);
% plot(x(2:end-1), stress_bottom);
% title('Stress on the Bottom Wall');
% xlabel('x');
% ylabel('Stress');
```

under relaxation کد

```
% VORTICITY/STREAM FUNCTION LID DRIVEN CAVITY FLOW SOLVER BY JOE MOLVAR
clear; close all;
% Given Parameters
Nx = 52;
L = 1;
Wall_Velocity = 29.25; % Nodes X; Domain Size; Wall Velocity
rho = 1;
mu = 0.01; % Density; Dynamic Viscosity;
dt = 0.001;
maxIt = 50000;
maxe = 1e-7; % Time Step; Max iterations; Max error
% 1D Grid Setup
Ny = Nx;
h = L / (Nx - 1);
x = 0:h:L;
y = 0:h:L;
im = 1:Nx-2; i = 2:Nx-1; ip = 3:Nx; jm = 1:Ny-2; j = 2:Ny-1; jp = 3:Ny;
% Initialize Matrices
Vo = zeros(Nx, Ny);
St = zeros(Nx, Ny);
Vop = zeros(Nx, Ny);
```

```
u = zeros(Nx, Ny);
v = zeros(Nx, Ny);
u(2:Nx-1, Ny) = Wall_Velocity;
Vop = Vo;
% Solve Loop (Similar to Gauss-Siedel Method)
for iter = 1:maxIt
    % Boundary Conditions
    Vo(:, Ny) = -2 * St(:, Ny-1) / (h^2) - Wall_Velocity * 2 / h; % Top
    Vo(:, 1) = -2 * St(:, 2) / (h^2); % Bottom
    Vo(1, :) = -2 * St(2, :) / (h^2); % Left
    Vo(Nx, :) = -2 * St(Nx-1, :) / (h^2); % Right
    % Partially Solve Vorticity Transport Equation
    s_x1 = max(u(i, j), 0);
    s_x2 = max(-u(i, j), 0);
    s_y1 = max(v(i, j), 0);
    s_y2 = max(-v(i, j), 0);
    % SOR Calculation
    OMEGA = 1.5;
    w=0.8;
    Vo(i, j) =Vop(i, j) +w*( (OMEGA ./ (1 - 4*dt*(mu/(rho*h^2) - dt*(s_x1 ./
h) -...
                dt*(s_x2 ./ h) - dt*(s_y1 ./ h) - dt*(s_y2 ./ h))) .*
((mu/(rho*h^2)) * (Vop(ip, j) +...
                Vop(im, j) + Vop(i, jp) + Vop(i, jm) - 4 * Vop(i, j)) - (s_x1
./ h) .* ...
               (Vop(i, j) - Vop(im, j)) + (s_x2 ./ h ).* (Vop(ip, j) - Vop(i,
j)) - ...
               (s_y1 ./ h) .* (Vop(i, j) - Vop(i, jm)) + (s_y2 ./ h) .*
(Vop(i, jp) - Vop(i, j))) * dt));
    % Partially Solve Elliptical Vorticity Equation for Stream Function
   Stp = St;
St(i,j) = Stp(i,j) + w*((Vo(i,j)*h^2 + St(ip,j) + St(i,jp) + St(i,jm) +
St(im,j))/4 - Stp(i,j));
    u(i, j) = (St(i, jp) - St(i, jm)) / (2 * h);
```

```
    v(i, j) = (-St(ip, j) + St(im, j)) / (2 * h);
    % Check for Convergence
    if iter > 10
        error = max(max(abs(Vo - Vop)));
        if error < maxe
            break;
        end
    end
    Vop = Vo;
end
% Plots
cm = hsv(ceil(100 / 0.7));
cm = flipud(cm(1:100, :));
figure(1);
contourf(x, y, u', 23, 'LineColor', 'none');
title('U-velocity');
xlabel('x-location');
ylabel('y-location');
% contourf(x, y, v', 23, 'LineColor', 'none');
% title('V-velocity');
% xlabel('x-location');
% ylabel('y-location');
axis equal;
axis([0 L 0 L]);
colormap(cm);
colorbar('westoutside');
figure(2);
plot(y, u(round(Ny/2), :));
title('Centerline x-direction velocity');
xlabel('y/L');
ylabel('u/U');
axis square;
```

```
xlim([0 L]);
grid on;
N = 1000;
xstart = L * rand(N, 1);
ystart = L * rand(N, 1);
[X, Y] = meshgrid(x, y);
figure(3);
h = streamline(X, Y, u', v', xstart, ystart, [0.1, 200]);
title('Stream Function');
xlabel('x-location');
ylabel('y-location');
axis equal;
set(h, 'color', 'k');
% a and b
%Find indices for the middle vertical and horizontal lines

%Define the midpoints for vertical and horizontal midlines
midpoint_x = round(Nx / 2);
midpoint_y = round(Ny / 2);

% Extract the vertical velocity profile (v) along the midlines
v_vertical_midline = v(midpoint_x, :);
v_horizontal_midline = v(:, midpoint_y);

% Plot the vertical velocity profile along the vertical midline
figure;
plot(y, v_vertical_midline, 'b-', 'LineWidth', 2);
xlabel('y');
ylabel('v');
title('Vertical Velocity Profile along Vertical Midline');
grid on;
ylim([min(v(:)), max(v(:))]);
```

```
% Plot the vertical velocity profile along the horizontal midline
figure;
plot(x, v_horizontal_midline, 'r-', 'LineWidth', 2);
xlabel('x');
ylabel('v');
title('Vertical Velocity Profile along Horizontal Midline');
grid on;
ylim([min(v(:)), max(v(:))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
% Extract the vertical velocity profile (v) along the midlines
u_vertical_midline = u(midpoint_x, :);
u_horizontal_midline = u(:, midpoint_y);

% Plot the vertical velocity profile along the vertical midline
figure;
plot(y, u_vertical_midline, 'b-', 'LineWidth', 2);
xlabel('y');
ylabel('u');
title('horizental Velocity Profile along Vertical Midline');
grid on;
ylim([min(u(:)), max(u(:))]);

% Plot the vertical velocity profile along the horizontal midline
figure;
plot(x, u_horizontal_midline, 'r-', 'LineWidth', 2);
xlabel('x');
ylabel('u');
title('horizental Velocity Profile along Horizontal Midline');
grid on;
ylim([min(u(:)), max(u(:))]);
```

کد بخش امتیازی

```
clear; close all
%%*************steady state equation*****************
%%% GIVENS
Nx = 52; L = 1; Wall_Velocity = 10; % Nodes X; Domain Size; Velocity
rho = 1; mu = 0.01;  k=0.3; % Density; Dynamic Viscosity;
 maxIt = 50000; maxe = 1e-7; % Time Step; Max iter; Max error
%%% SETUP 1D GRID
Ny = Nx; h = L / (Nx - 1); x = 0:h:L; y = 0:h:L;
im = 1:Nx-2; i = 2:Nx-1; ip = 3:Nx; jm = 1:Ny-2; j = 2:Ny-1; jp = 3:Ny;
%%% PRELOCATE MATRICES
Vo = zeros(Nx, Ny); St = Vo; Vop = Vo; u = Vo; v = Vo; T = 300 * ones(Nx,
Ny); % Initial temperature


%%% VELOCITY ON THE UPPER WALL (NO-SLIP CONDITION)
u(2:Nx-1, Ny) = Wall_Velocity;


%%% SOLVE LOOP SIMILAR TO GAUSS-SEIDEL METHOD
for iter = 1:maxIt
    %%% CREATE VELOCITY FROM STREAM FUNCTION
    u(i, j) = (St(i, jp) - St(i, jm)) / (2*h);
    v(i, j) = (-St(ip, j) + St(im, j)) / (2*h);


     T(1, 1:Ny) = T(2, 1:Ny);
    T(Nx, 1:Ny) = T(Nx-1, 1:Ny);
    T(1:Nx, 1) = 300;
    T(1:Nx, Ny) = 330;
    %%% TEMPERATURE EQUATION (ENERGY EQUATION)
    % Temporal discretization
    Tn = T;
    T(i,j) =(k * (Tn(ip, j) + Tn(im, j) + Tn(i, jp) + Tn(i, jm)) + ...
```

```
        max(u(i, j), 0) .* h .* Tn(im, j) + max(-u(i, j), 0) .* h .* Tn(ip,
j) + ...
        max(v(i, j), 0) .* h .* Tn(i, jm) + max(-v(i, j), 0) .* h .* Tn(i,
jp)) ./ ...
        (max(u(i, j), 0) .* h + max(-u(i, j), 0) .* h + max(v(i, j), 0) .* h
+ max(-v(i, j), 0) .* h + 4 * k);




    %%% SAVING THE VALUE OF VORTICITY AT THE LAST STEP
    Vop = Vo;

    %%% CREATE BOUNDARY CONDITIONS
    Vo(1:Nx, Ny) = -2 * St(1:Nx, Ny-1) / (h^2) - Wall_Velocity * 2 / h; % Top
    Vo(1:Nx, 1) = -2 * St(1:Nx, 2) / (h^2); % Bottom
    Vo(1, 1:Ny) = -2 * St(2, 1:Ny) / (h^2); % Left
    Vo(Nx, 1:Ny) = -2 * St(Nx-1, 1:Ny) / (h^2); % Right

    %%% PARTIALLY SOLVE VORTICITY TRANSPORT EQUATION
    Vo(i, j) = (mu / rho * (Vop(ip, j) + Vop(im, j) + Vop(i, jp) + Vop(i,
jm)) + ...
        max(u(i, j), 0) .* h .* Vop(im, j) + max(-u(i, j), 0) .* h .* Vop(ip,
j) + ...
        max(v(i, j), 0) .* h .* Vop(i, jm) + max(-v(i, j), 0) .* h .* Vop(i,
jp)) ./ ...
        (max(u(i, j), 0) .* h + max(-u(i, j), 0) .* h + max(v(i, j), 0) .* h
+ max(-v(i, j), 0) .* h + 4 * mu / rho);

    %%% PARTIALLY SOLVE ELLIPTIC VORTICITY EQUATION FOR STREAM FUNCTION
    St(i, j) = (Vo(i, j) * h^2 + St(ip, j) + St(i, jp) + St(i, jm) + St(im,
j)) / 4;



    %%% CHECK FOR CONVERGENCE
    if iter > 10
```

```
        error = max(max(Vo - Vop));
        if error < maxe
            break;
        end
    end
end

% Plot temperature contour
contourf(x, y, T', 20, 'LineWidth', 1.5);
colorbar;
colormap('jet');
xlabel('X');
ylabel('Y');
title('Temperature Contour');
```

منابع و مراجع

[1] High-Re Solutions for Incompressible Flow Using the Navier-Stoles Equations and a Multigrid Method – U. Ghia, K.N. Ghia and C.T. Shin

[2]A fast and short Matlab code to solve the lid driven cavity flow problem using the vorticity-stream function formulation – Joe Molvar