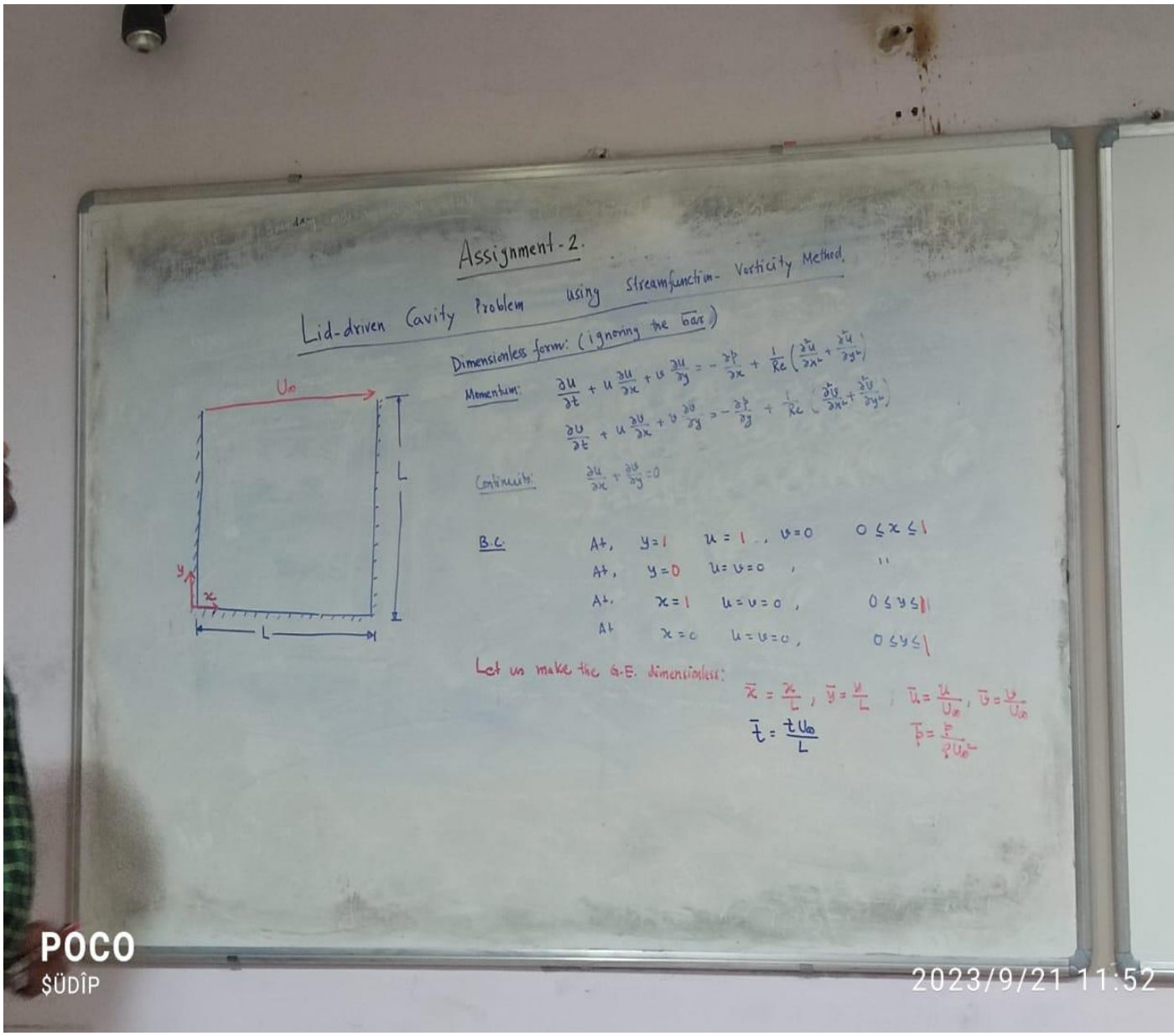
Name: - Swapnendu Chakraborty

BME UG3 Mechanical Engineering

Jadavpur University

Computational Fluid Dynamics -Assignment **2**

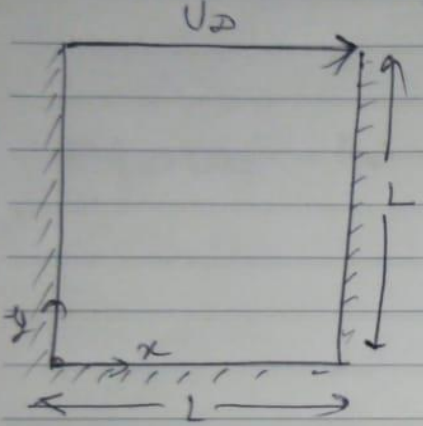
Problem Statement: -



Formulation: -

CFD - Assignment - 2

Lid-driven Cavity Problem using Stream-function Vorticity Method



Dimensionless form:- (ignoring the bars).

x-Momentum eq

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

y-Momentum eq :

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

Continuity $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$

B.C

At $y = 1$, $u = 1$, $v = 0$. $0 \le x \le 1$.
At $y = 0$, $u = 0$, $v = 0$ $0 \le x \le 1$.
At $x = 1$, $u = 0$, $v = 0$ $0 \le y \le 1$
At $x = 0$, $u = 0$, $v = 0$ $0 \le y \le 1$

Dimensionless parameters

$$\bar{x} = \frac{x}{L}, \quad \bar{y} = \frac{y}{L}, \quad \bar{u} = \frac{u}{U_0}, \quad \bar{v} = \frac{v}{U_0}, \quad \bar{p} = \frac{p}{\rho U_0^2}$$

$$\bar{t} = \frac{t\,U_0}{L}$$

By Stream function - Vorticity Method

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}. \qquad u = \frac{\partial \psi}{\partial y} \qquad v = -\frac{\partial \psi}{\partial x}.$$

$\omega \rightarrow$ vorticity function where $\psi \rightarrow$ stream function.

vorticity equation :-

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right)$$

$$\Rightarrow \boxed{\frac{\partial w}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial w}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial w}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right)}$$

Stream function

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -w.$$

Boundary conditions

No slip walls $\Rightarrow u = v = 0. \Rightarrow \psi = 0.$

$$w = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = 0.$$

Top plate :- (Moving wall) :

$$u = \frac{\partial \psi}{\partial y} \Rightarrow \psi = \int u\,dy + C \Rightarrow \int dy + C = y + C$$

$\therefore \psi = y + C$ At $y = 1, \psi = 0 \Rightarrow C = -1.$

$$\boxed{\psi = y - 1} \qquad 0 \le x \le 1.$$

$$\boxed{w_{i,j} = \frac{2(\psi_{i,j} + \Delta y - \psi_{i,j-1})}{(\Delta y)^2}}$$

One of the secrets of life is to make stepping stones out of stumbling blocks. – Jack Penn

For interior points at $t=0$,

$\Rightarrow \psi_{i,j}^{n+1} = 0$, $w_{i,j}^{n+1} = 0$,

$\psi_{i,j}^{n} > 0$, $w_{i,j}^{n} = 0$.

Time stepping $t = t + \Delta t$. $(\Delta t = 0.01 = 10^{-2}$ sec).

⇒ By discretizing the vorticity equation

$$\frac{w_{i,j}^{n+1} - w_{i,j}^{n}}{\Delta t} + \left(\frac{\psi_{i,j+1}^{n+1} - \psi_{i,j-1}^{n+1}}{2\Delta y}\right)\left(\frac{w_{i+1,j}^{n+1} - w_{i-1,j}^{n+1}}{2\Delta x}\right)$$

$$-\left(\frac{\psi_{i+1,j}^{n+1} - \psi_{i-1,j}^{n+1}}{2\Delta x}\right)\left(\frac{w_{i,j+1}^{n+1} - w_{i,j-1}^{n+1}}{2\Delta y}\right)$$

$$= \frac{1}{Re}\left[\left(\frac{w_{i+1,j}^{n+1} - 2w_{i,j}^{n+1} + w_{i-1,j}^{n+1}}{(\Delta x)^2}\right) + \left(\frac{w_{i,j+1}^{n+1} - 2w_{i,j}^{n+1} + w_{i,j-1}^{n+1}}{(\Delta y)^2}\right)\right].$$

$$\Rightarrow w_{i,j}^{n+1}\left[\frac{1}{\Delta t} + \frac{2}{Re(\Delta x)^2} + \frac{2}{Re(\Delta y)^2}\right].$$

Sunday 22

$$= \frac{w_{i,j}^{n}}{\Delta t} + \frac{(\psi_{i+1,j}^{n+1} - \psi_{i-1,j}^{n+1})(w_{i,j+1}^{n+1} - w_{i,j-1}^{n+1})}{4(\Delta x \Delta y)}$$

$$- \frac{(\psi_{i,j+1}^{n+1} - \psi_{i,j-1}^{n+1})(w_{i+1,j}^{n+1} - w_{i-1,j}^{n+1})}{4\Delta x \Delta y}.$$

$$+ \frac{1}{Re}\left[\frac{w_{i+1,j}^{n+1} + w_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{w_{i,j+1}^{n+1} + w_{i,j-1}^{n+1}}{(\Delta y)^2}\right].$$

$$\therefore (w_{i,j})_1^{n+1} = \left[\frac{w_{i,j}^n}{\Delta t} + \frac{(\psi_{i+1,j}^{n+1} - \psi_{i-1,j}^{n+1})(w_{i,j+1}^{n+1} - w_{i,j-1}^{n+1})}{4(\Delta x)(\Delta y)} - \frac{(\psi_{i,j+1}^{n+1} - \psi_{i,j-1}^{n+1})(w_{i+1,j}^{n+1} - w_{i-1,j}^{n+1})}{4\Delta x \Delta y} + \frac{1}{Re}\left[\frac{w_{i+1,j}^{n+1} + w_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{w_{i,j+1}^{n+1} + w_{i,j-1}^{n+1}}{(\Delta y)^2} \right] \right] \Bigg/ \left[\frac{1}{\Delta t} + \frac{2}{Re(\Delta x)^2} + \frac{2}{Re(\Delta y)^2} \right]$$

By discretizing the stream function

$$\frac{\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{(\Delta y)^2} = -(w_{i,j}^{n+1})_1$$

$$\Rightarrow \frac{\psi_{i+1,j}^{n+1} + \psi_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{\psi_{i,j+1}^{n+1} + \psi_{i,j-1}^{n+1}}{(\Delta y)^2} + (w_{i,j}^{n+1})_1 = \psi_{i,j}^{n+1} \left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} \right]$$

$$\therefore (\psi_{i,j}^{n+1})_1 = \left[\frac{\psi_{i+1,j}^{n+1} + \psi_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{\psi_{i,j+1}^{n+1} + \psi_{i,j-1}^{n+1}}{(\Delta y)^2} + (w_{i,j}^{n+1}) \right] \Bigg/ \left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} \right]$$

For steady-state analysis:-

$$\text{If } \textcircled{1} \Rightarrow \left| (\psi_{i,j}^{n+1})_1 - \psi_{i,j}^{n+1} \right| < \varepsilon_1 \quad \textcircled{1}$$

If $\Rightarrow \left|(\omega_{i,j}^{n+1})_1 - (\omega_{i,j}^{n})\right| < \varepsilon_2$.

(11)

If condition (1) and condition (2) satisfies, then

$$u_{i,j} = \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2(\Delta y)} \Rightarrow \quad v_{i,j} = -\left(\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2\Delta x}\right).$$

For real U, V

$$U_{i,j} = u_{i,j} * U_\infty$$
$$V_{i,j} = v_{i,j} * U_\infty$$

For pressure At $t=0$, $P_{i,j} = 0$.

From the viscous flow and stream function pressure determinated is by

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \nabla^2 p = 2\left[\left(\frac{\partial^2 \psi}{\partial x^2} * \frac{\partial^2 \psi}{\partial y^2}\right) - \left(\frac{\partial^2 \psi}{\partial x \partial y}\right)^2\right]$$

$$\frac{\partial \psi}{\partial y} = \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2\Delta y}$$

$$\frac{\partial^2 \psi}{\partial x \partial y} = \frac{\left(\frac{\partial \psi}{\partial x}\right)_{i,j+1} - \left(\frac{\partial \psi}{\partial x}\right)_{i,j-1}}{2\Delta y}$$

$$\frac{\partial^2 \psi}{\partial x \partial y} = \frac{\left(\frac{\psi_{i+1,j+1} - \psi_{i+1,j+1}}{2\Delta x}\right) - \left(\frac{\psi_{i+1,j-1} - \psi_{i-1,j-1}}{2\Delta x}\right)}{2\Delta y}$$

$$\frac{\partial^2 \psi}{\partial x \partial y} = \frac{\psi_{i+1,j+1} - \psi_{i-1,j+1} - \psi_{i+1,j+1} + \psi_{i-1,j-1}}{4\Delta x \Delta y}.$$

Discretizing the pressure equation, we get

$$\frac{P_{i+1,j} - 2P_{i,j} + P_{i-1,j}}{(\Delta x)^2} + \frac{P_{i,j+1} - 2P_{i,j} + P_{i,j-1}}{(\Delta y)^2} = 2\left[\left(\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2}\right)\left(\frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{(\Delta y)^2}\right) - \left(\frac{\psi_{i+1,j+1} - \psi_{i+1,j-1} - \psi_{i-1,j+1} + \psi_{i-1,j-1}}{4\,\Delta x\,\Delta y}\right)^2\right]$$

$$\frac{P_{i+1,j} + P_{i-1,j}}{(\Delta x)^2} + \frac{P_{i,j+1} + P_{i,j-1}}{(\Delta y)^2} + 2\left(\frac{\psi_{i+1,j+1} - \psi_{i-1,j+1} - \psi_{i+1,j-1} + \psi_{i-1,j-1}}{4\,\Delta x\,\Delta y}\right)^2 - 2\left[\left(\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2}\right)\left(\frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{(\Delta y)^2}\right)\right]$$

$$\left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2}\right] = P_{i,j}$$

For real Pressure

$$P_i = P_{i,j} * \rho U_\infty^2$$

$$P_{i,j} = P_{i,j} * \frac{4}{\rho U_\infty L} * U_\infty^2$$

$$Re = \frac{\rho U_\infty (L)}{\mu}, \quad \rho = \frac{\mu Re}{U_\infty L}$$

$$\boxed{P_{i,j} = P_{i,j} * \frac{4 Re\, U_\infty}{L}}$$

Well-timed silence hath more eloquence than speech. – Martin Fraquhar Tupper

If condition (1) and (2) doesn't satisfy, then

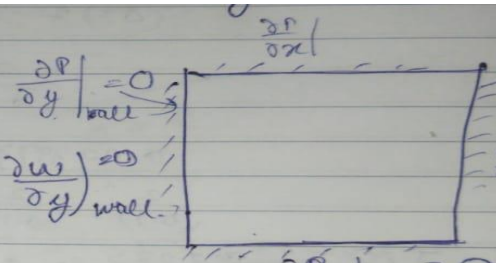
$$\psi_{i,j}^{p+1} = (\psi_{i,j})_1^{p+1}$$

$$w_{i,j}^{p+1} = (w_{i,j})_1^{p+1}.$$

For wall pressures:

For wall $y = 0$,

$$\left.\frac{\partial p}{\partial x}\right|_{wall} = +4\left.\frac{\partial^2 v}{\partial y^2}\right)_{wall} = -4\left.\frac{\partial w}{\partial y}\right)_{wall} = 0$$



$\Rightarrow$ If

$$\frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x} = -4\left(\frac{w_{i,j+1} - w_{i,j-1}}{\Delta y}\right).$$

December 2021

MONDAY 30 NOVEMBER

2-0-2-0

Week-49 (335-031)

→ Schematic Diagram for the algorithm

Step-1 Define the definitions of stream functions and vorticity and put all the boundary conditions

↓

Step-2 Initialise at interior points $\psi_{i,j}^{p+1} = 0$, $w_{i,j}^{p+1} = 0$

$\psi_{i,j}^{p} = 0$, $w_{i,j}^{p} = 0$.

↓

Step-3 Choose Time step $\Delta t = 0.01$ seconds.

↓

Step-4 Calculate vorticity at non-boundary points

↓

Step-5 Calculate vorticity at the walls.

↓

Step-6 Calculate stream functions at non-boundary points

↓

Step-7 Check convergence and steady state.

NO. $\psi_{i,j}^{p+1} = (\psi_{i,j}^{p+1})_{new}$, $w_{i,j}^{p+1} = (w_{i,j}^{p+1})_{new}$

Yes ↓

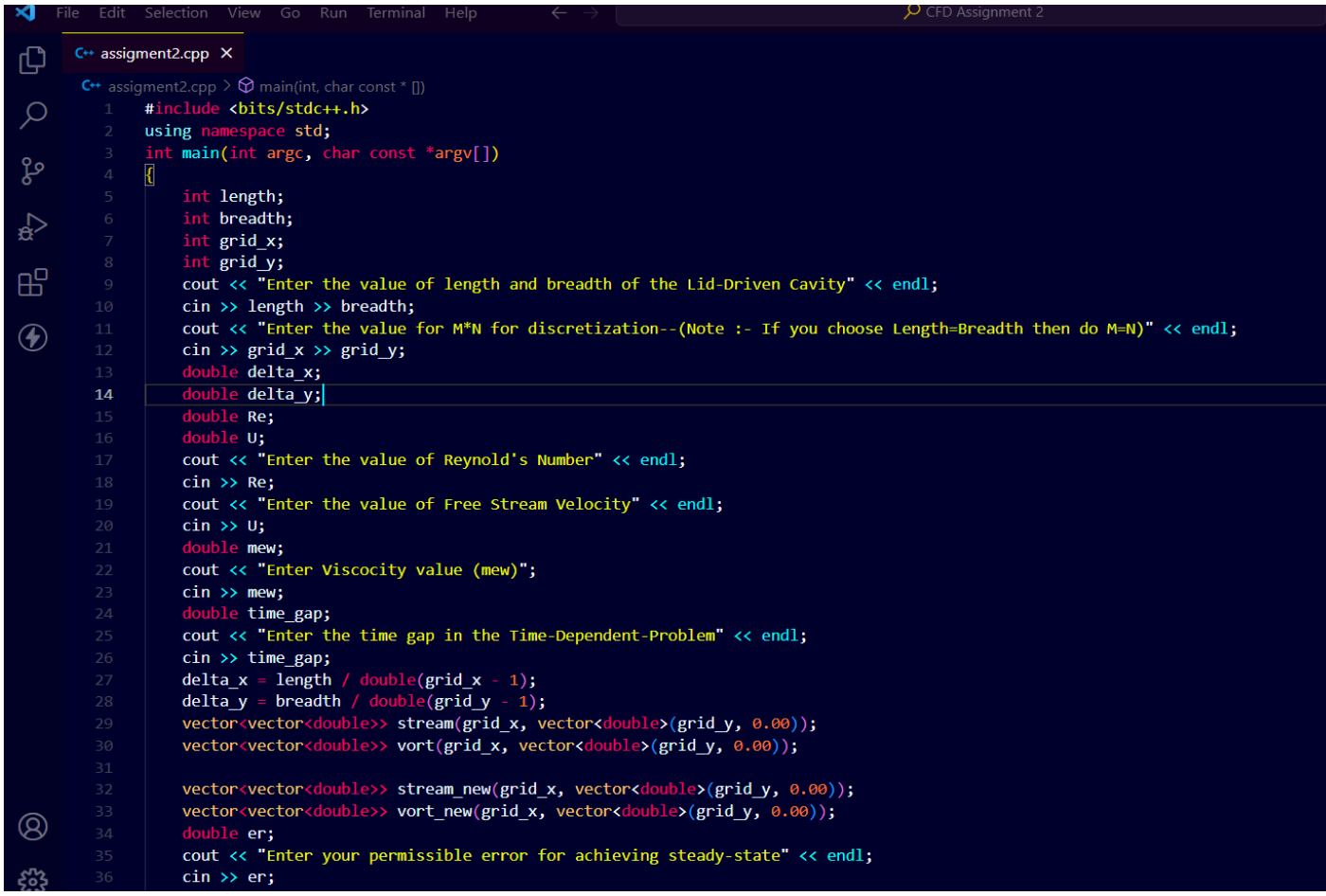
Step-8 With values of $\psi_{i,j}^{p+1}$, calculate U, V, P.

↓

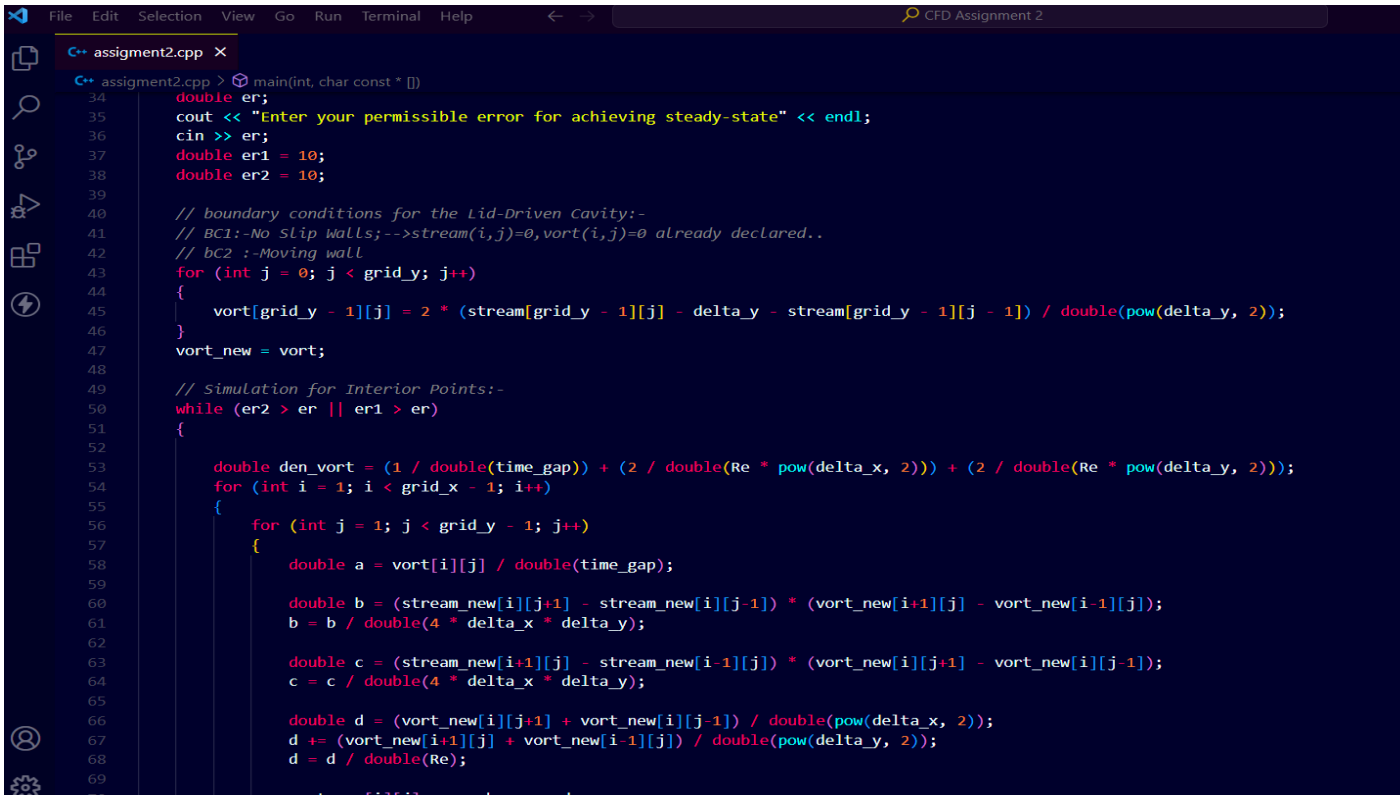
Step-9 Plot it on Tecplot 360

How much easier it is to be critical than to be correct. – Benjamin Disraeli

C++ Code:-



```cpp
#include <bits/stdc++.h>
using namespace std;
int main(int argc, char const *argv[])
{
    int length;
    int breadth;
    int grid_x;
    int grid_y;
    cout << "Enter the value of length and breadth of the Lid-Driven Cavity" << endl;
    cin >> length >> breadth;
    cout << "Enter the value for M*N for discretization--(Note :- If you choose Length=Breadth then do M=N)" << endl;
    cin >> grid_x >> grid_y;
    double delta_x;
    double delta_y;
    double Re;
    double U;
    cout << "Enter the value of Reynold's Number" << endl;
    cin >> Re;
    cout << "Enter the value of Free Stream Velocity" << endl;
    cin >> U;
    double mew;
    cout << "Enter Viscocity value (mew)";
    cin >> mew;
    double time_gap;
    cout << "Enter the time gap in the Time-Dependent-Problem" << endl;
    cin >> time_gap;
    delta_x = length / double(grid_x - 1);
    delta_y = breadth / double(grid_y - 1);
    vector<vector<double>> stream(grid_x, vector<double>(grid_y, 0.00));
    vector<vector<double>> vort(grid_x, vector<double>(grid_y, 0.00));

    vector<vector<double>> stream_new(grid_x, vector<double>(grid_y, 0.00));
    vector<vector<double>> vort_new(grid_x, vector<double>(grid_y, 0.00));
    double er;
    cout << "Enter your permissible error for achieving steady-state" << endl;
    cin >> er;
```



```cpp
    double er;
    cout << "Enter your permissible error for achieving steady-state" << endl;
    cin >> er;
    double er1 = 10;
    double er2 = 10;

    // boundary conditions for the Lid-Driven Cavity:-
    // BC1:-No Slip Walls;-->stream(i,j)=0,vort(i,j)=0 already declared..
    // bC2 :-Moving wall
    for (int j = 0; j < grid_y; j++)
    {
        vort[grid_y - 1][j] = 2 * (stream[grid_y - 1][j] - delta_y - stream[grid_y - 1][j - 1]) / double(pow(delta_y, 2));
    }
    vort_new = vort;

    // Simulation for Interior Points:-
    while (er2 > er || er1 > er)
    {

        double den_vort = (1 / double(time_gap)) + (2 / double(Re * pow(delta_x, 2))) + (2 / double(Re * pow(delta_y, 2)));
        for (int i = 1; i < grid_x - 1; i++)
        {
            for (int j = 1; j < grid_y - 1; j++)
            {
                double a = vort[i][j] / double(time_gap);

                double b = (stream_new[i][j+1] - stream_new[i][j-1]) * (vort_new[i+1][j] - vort_new[i-1][j]);
                b = b / double(4 * delta_x * delta_y);

                double c = (stream_new[i+1][j] - stream_new[i-1][j]) * (vort_new[i][j+1] - vort_new[i][j-1]);
                c = c / double(4 * delta_x * delta_y);

                double d = (vort_new[i][j+1] + vort_new[i][j-1]) / double(pow(delta_x, 2));
                d += (vort_new[i+1][j] + vort_new[i-1][j]) / double(pow(delta_y, 2));
                d = d / double(Re);
```

assigment2.cpp

```
                d += (vort_new[i+1][j] + vort_new[i-1][j]) / double(pow(delta_y, 2));
                d = d / double(Re);

                vort_new[i][j] = a + b - c + d;
                vort_new[i][j] = vort_new[i][j] / double(den_vort);
            }
        }
        // new vortex fomation at lower wall
        for (int j = 0; j < grid_y; j++)
        {
            vort_new[0][j] = -2 * (stream[1][j] - stream[0][j]) / double(pow(delta_y, 2));
        }

        // new vortex fomation at left wall
        for (int i = 0; i < grid_x; i++)
        {
            vort_new[i][0] = -2 * (stream[i][1] - stream[i][0]) / double(pow(delta_x, 2));
        }

        // new vortex fomation at right wall
        for (int i = 0; i < grid_x; i++)
        {
            vort_new[i][grid_y - 1] = 2 * (stream[i][grid_y - 1] - stream[i][grid_y - 2]) / double(pow(delta_x, 2));
        }
        // new vortex for the upper wall
        for (int j = 0; j < grid_y; j++)
        {
            vort_new[grid_y - 1][j] = 2 * (stream[grid_y - 1][j] - delta_y - stream[grid_y - 2][j]) / double(pow(delta_y, 2));
        }

        double den_stream = (2 / double(pow(delta_x, 2))) + (2 / double(pow(delta_y, 2)));

        double den_stream = (2 / double(pow(delta_x, 2))) + (2 / double(pow(delta_y, 2)));

        for (int i = 1; i < grid_x - 1; i++)
        {
            for (int j = 1; j < grid_y - 1; j++)
            {
                double x = (stream_new[i][j+1] + stream_new[i][j-1]) / double(pow(delta_x, 2));
                x += (stream_new[i+1][j] + stream_new[i-1][j]) / double(pow(delta_y, 2));
                x += vort_new[i][j];
                stream_new[i][j] = x;
                stream_new[i][j] = stream_new[i][j] / double(den_stream);
            }
        }
        er1 = 0, er2 = 0;
        for (int i = 1; i < grid_x-1; i++)
        {
            for (int j = 1; j < grid_y-1; j++)
            {
                if(abs(vort_new[i][j]-vort[i][j])>er1){
                    er1=abs(vort_new[i][j]-vort[i][j]);
                }
                if(abs(stream_new[i][j]-stream[i][j])>er2){
                    er2=abs(stream_new[i][j]-stream[i][j]);
                }
            }

        }
        cout <<"Error for Vorticity--->"<< er1 <<"---->  Error for Stream Function--->"<< er2 << endl;
        vort = vort_new;
        stream = stream_new;
    }
```

```cpp
    // calculation for U-direction and V-drection Velocities:-
    vector<vector<double>> U_direct(grid_x, vector<double>(grid_y, 0.00));
    vector<vector<double>> V_direct(grid_x, vector<double>(grid_y, 0.00));
    for (int i = 0; i < grid_x; i++)
    {
        U_direct[grid_x-1][i] = U;
    }

    for (int i = 1; i < grid_x - 1; i++)
    {
        for (int j = 1; j < grid_y - 1; j++)
        {
            U_direct[i][j] = (stream_new[i+1][j] - stream_new[i-1][j]) / double(2 * delta_y);
            U_direct[i][j] = U_direct[i][j] * U;
            V_direct[i][j] = -(stream_new[i][j+1] - stream_new[i][j-1]) / double(2 * delta_x);
            V_direct[i][j] = V_direct[i][j] * U;
        }
    }
    cout << "done2" << endl;

    vector<vector<double>> p(grid_x, vector<double>(grid_y, 0.00));
    double deno = (2 / double(pow(delta_x, 2))) + (2 / double(pow(delta_y, 2)));
    for (int i = 1; i < grid_x - 1; i++)
    {
        for (int j = 1; j < grid_y - 1; j++)
        {
            double psi = pow((stream_new[i + 1][j + 1] - stream_new[i +1][j - 1] - stream_new[i - 1][j + 1] + stream_new[i - 1][j - 1])
            double psi2 = ((stream_new[i][j+1] - (2 * stream_new[i][j]) + stream_new[i][j-1]) / double(pow(delta_x, 2))) * ((stream_new
            p[i][j] =2 * psi - (2 * psi2);
            p[i][j] = (p[i][j] * mew * Re * U) / length;
        }
    }
    cout << "done3" << endl;

    for (int j = 0; j < grid_y; j++)
    {
```

```cpp
                p[i][j] = (p[i][j] * mew * Re * U) / length;
            }
        }
        for (int j = 0; j < grid_y; j++)
        {
            p[0][j] = p[1][j];
            p[grid_y - 1][j] = p[grid_y - 2][j];
        }

        for (int i = 0; i < grid_x; i++)
        {
            p[i][0] = p[i][1];
            p[grid_x - 1][0] = p[grid_x - 2][0];
        }

        cout << "done" << endl;

        // Producting output in file
        //Contour Plots:---------------------
        ofstream out("Assignment_3.plt");
        string var = "VARIABLES=\"X_VAL\",\"Y_VAL\",\"U_VEl\",\"V-VEl\",\"Pressure\"\n";
        string zone = "ZONE F=POINT\n";
        string col = to_string(grid_x);
        string row = to_string(grid_y);
        string i = "I=" + col + ", J=" + row;
        out << var << zone << i << endl;
        for (int i = 0; i < grid_x; i++)
        {
            for (int j = 0; j < grid_y; j++)
            {
                double xi = i * delta_x;
                double yi = j * delta_y;
                string x = to_string(yi);
                x = x + ' ' + ' ';
                string y = to_string(xi);
                y = y + ' ' + ' ';
```

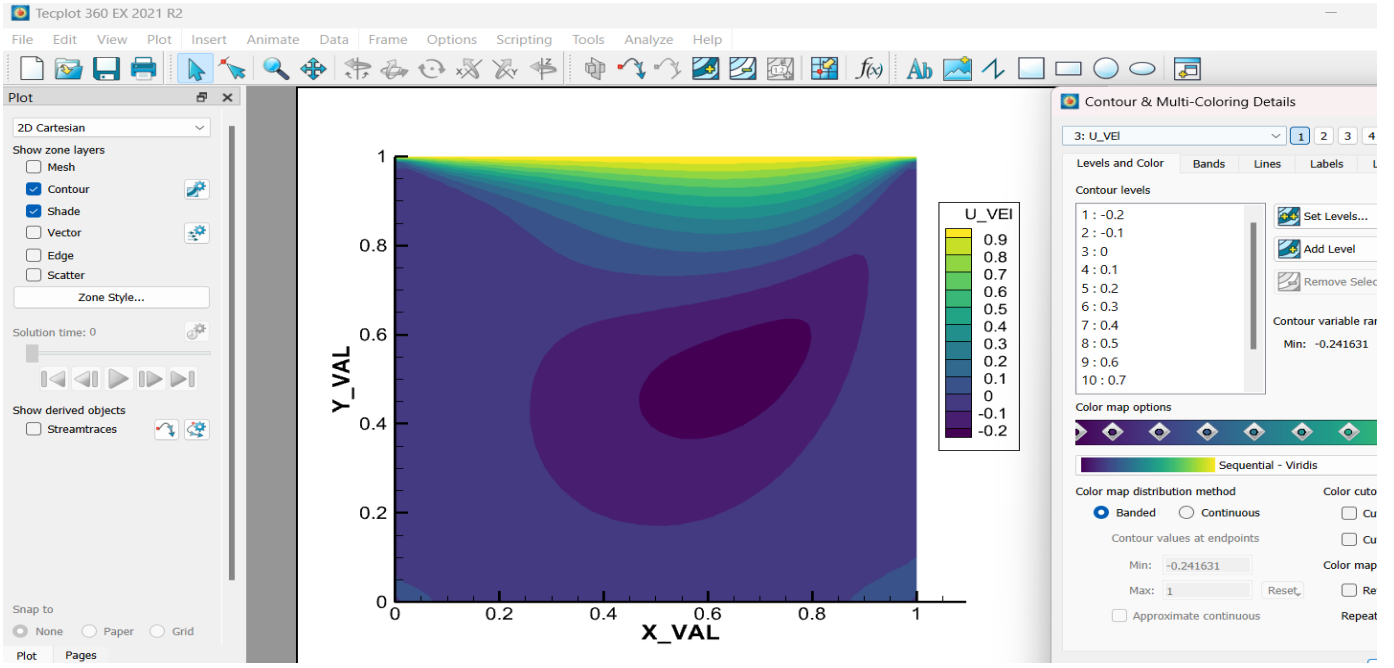
```
            double yi = j * delta_y;
            string x = to_string(yi);
            x = x + ' ' + ' ';
            string y = to_string(xi);
            y = y + ' ' + ' ';
            string U_vel = to_string(U_direct[i][j]);
            U_vel = U_vel + ' ' + ' ';
            string V_vel = to_string(V_direct[i][j]);
            V_vel = V_vel + ' ' + ' ';
            string pressure = to_string(p[i][j]);
            pressure = pressure + ' ' + ' ';
            string psi = to_string(stream_new[i][j]);
            psi = psi + ' ' + ' ';
            string vorti = to_string(vort_new[i][j]);
            vorti = vorti + ' ' + ' ';

            out << x << y << U_vel << V_vel << pressure <<psi<<vorti<< endl;
        }
    }
    cout << "done4" << endl;

// U_vs_Y(mid-plane) PLOT------------------------->
    ofstream outs("U_vs_Y(mid-plane).plt");
    string vars = "VARIABLES=\"Y_VAL\",\"U_Vel\"";
    string zones = "ZONE F=POINT\n";
    string rows = to_string(grid_y);
    string is ="J=" + row;
    outs << vars << zones << is << endl;
    for (int i = 0; i < grid_x; i++)
    {
            double xi = i * delta_x;
            string y = to_string(xi);
            y = y + ' ' + ' ';
            string U_vel = to_string(U_direct[i][50]);
            U_vel = U_vel + ' ' + ' ';
            outs<<y << U_vel << endl;
    }
```

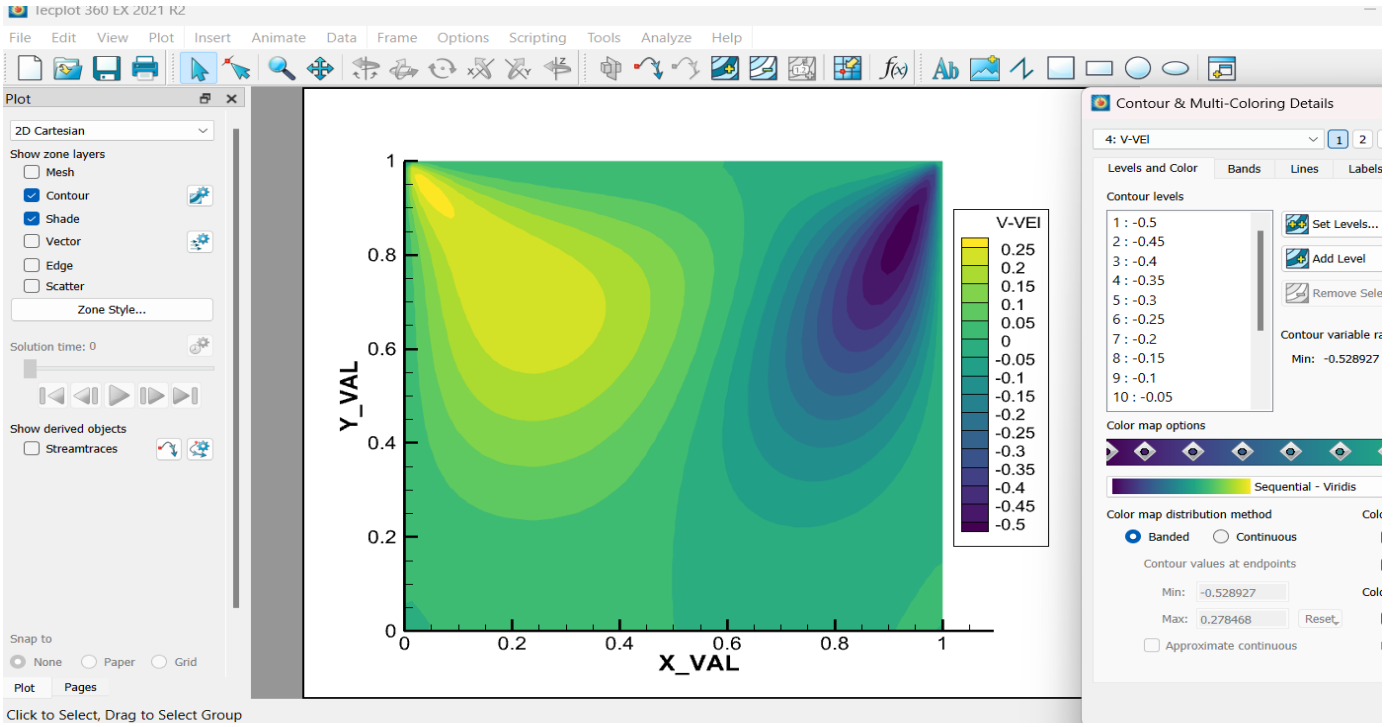
```
    string zones = "ZONE F=POINT\n";
    string rows = to_string(grid_y);
    string is ="J=" + row;
    outs << vars << zones << is << endl;
    for (int i = 0; i < grid_x; i++)
    {
            double xi = i * delta_x;
            string y = to_string(xi);
            y = y + ' ' + ' ';
            string U_vel = to_string(U_direct[i][50]);
            U_vel = U_vel + ' ' + ' ';
            outs<<y << U_vel << endl;
    }

    //  v_vs_Y(mid-plane) PLOT-------------------------->
    ofstream outsv("v_vs_Y(mid-plane).plt");
    string varsv = "VARIABLES=\"Y_VAL\",\"V_Vel\"";
    string zonesv = "ZONE F=POINT\n";
    string rowsv = to_string(grid_y);
    string isv ="J=" + row;
    outsv << varsv << zonesv << isv << endl;
    for (int i = 0; i < grid_x; i++)
    {
            double xi = i * delta_x;
            string y = to_string(xi);
            y = y + ' ' + ' ';
            string U_vel = to_string(V_direct[i][50]);
            U_vel = U_vel + ' ' + ' ';
            outsv<<y << U_vel << endl;
    }
    return 0;
}
```

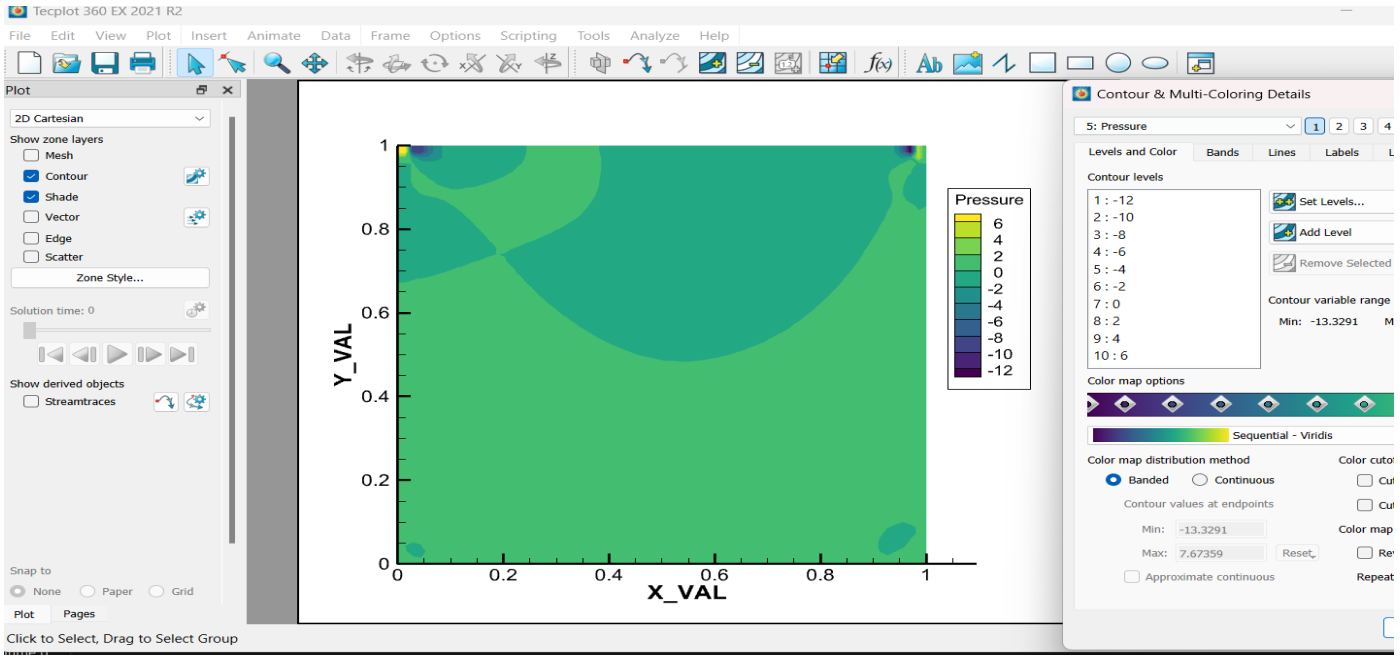
Results: -(Length=1m, Breadth=1m, Viscosity=0.001 Ns/m^2, Free Stream Velocity=1m/s, permissible Error=10^{-5}, Time Gap=0.01sec)

1)Reynolds's Number=100

→Grid Size=70x70

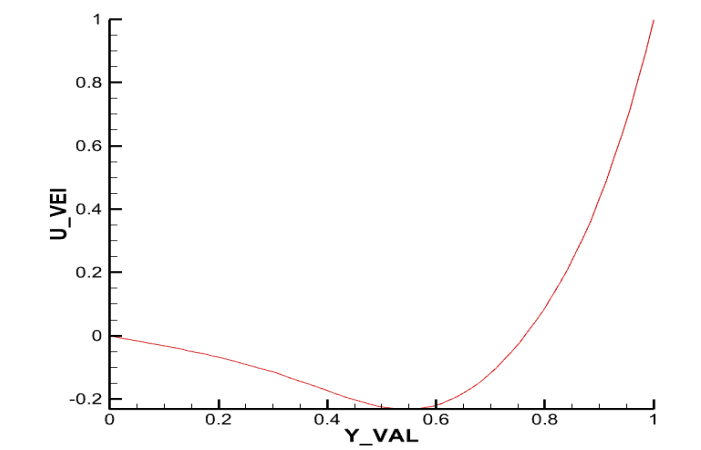


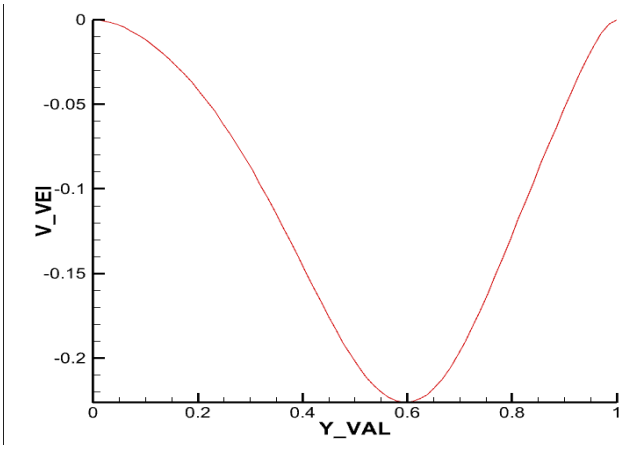


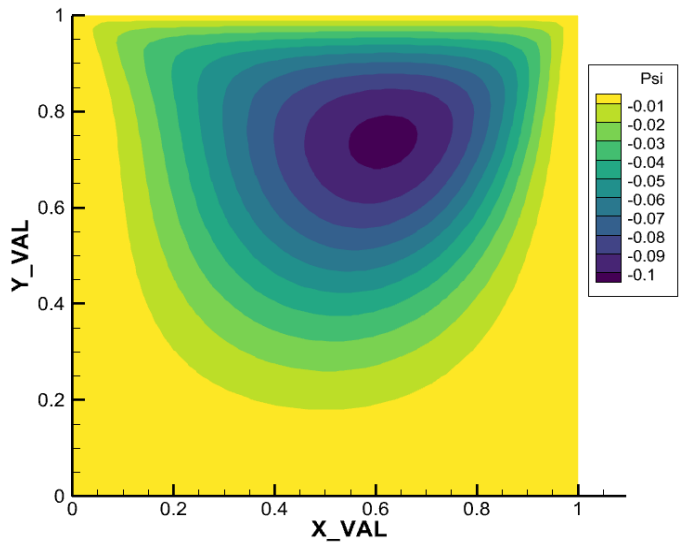


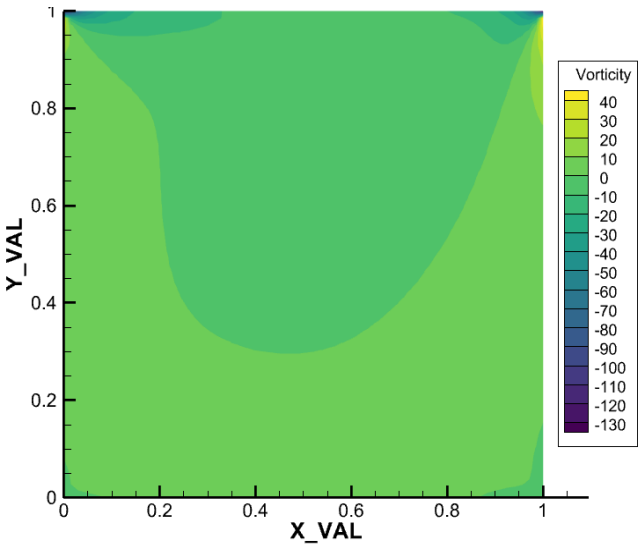
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -

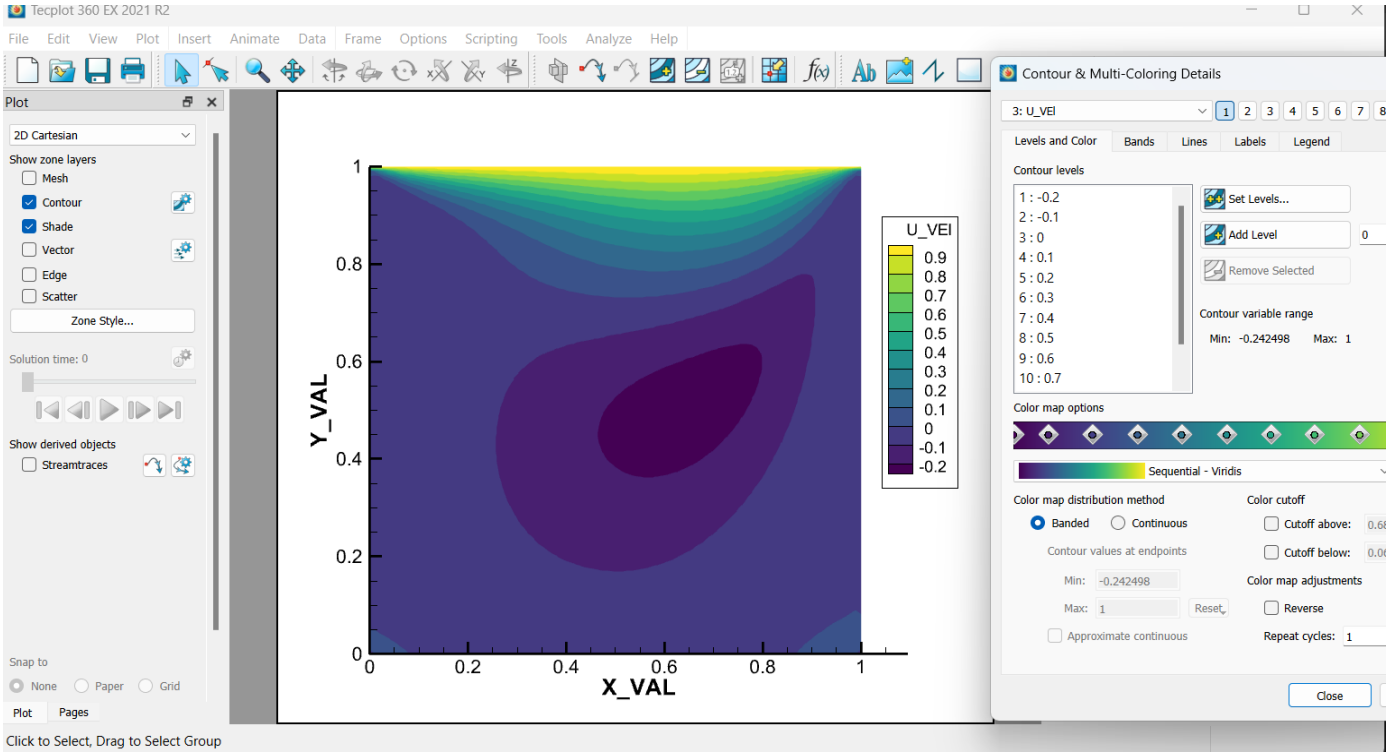


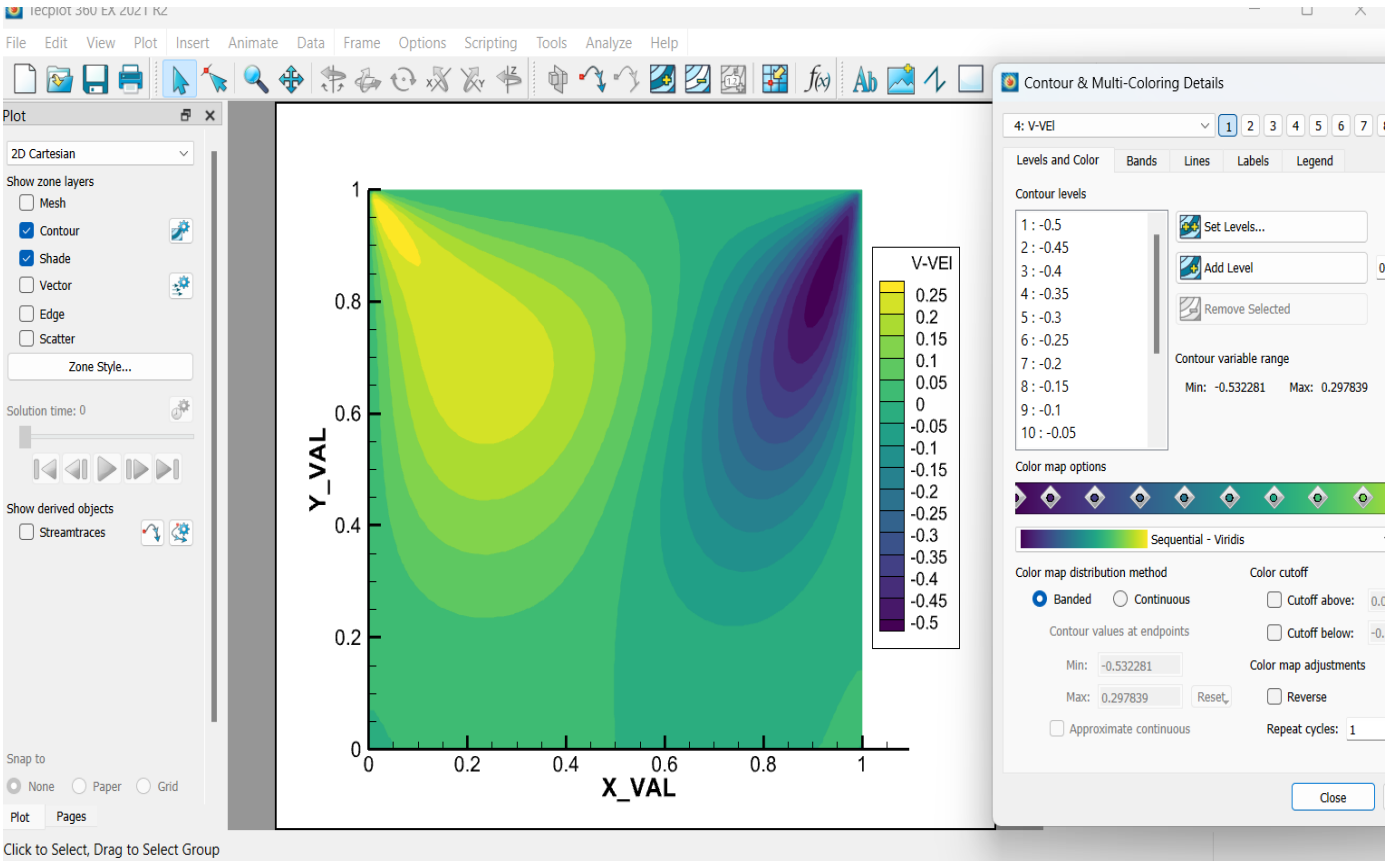


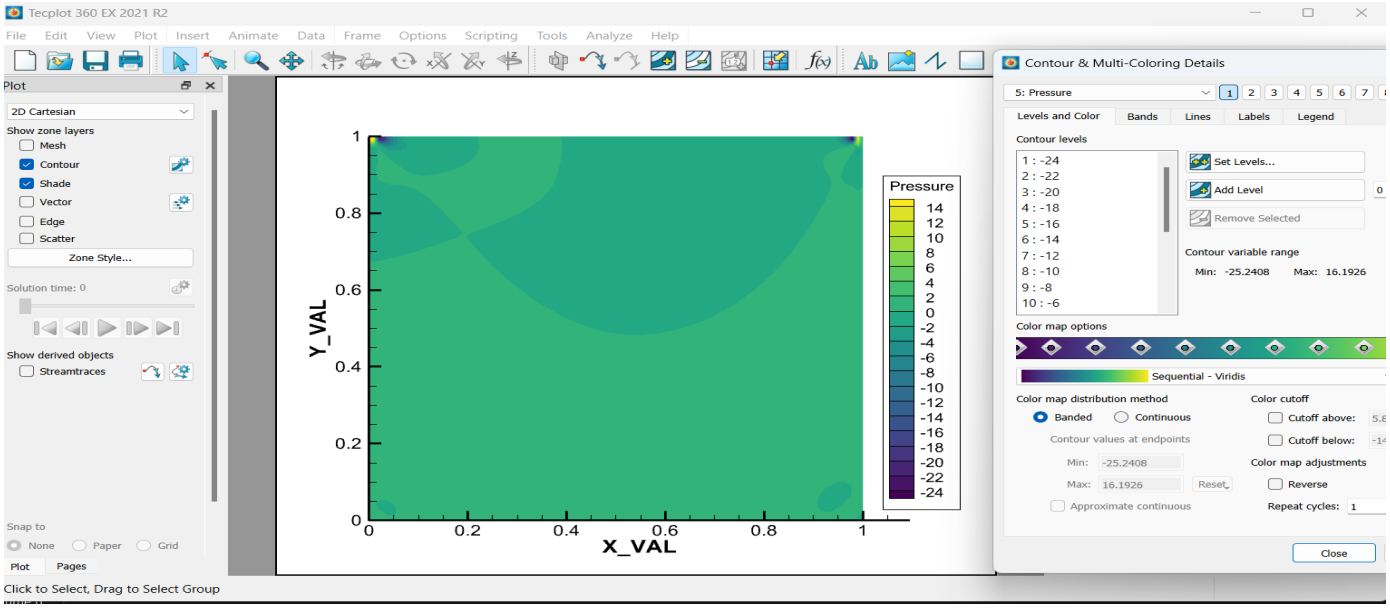




→Grid Size=100x100

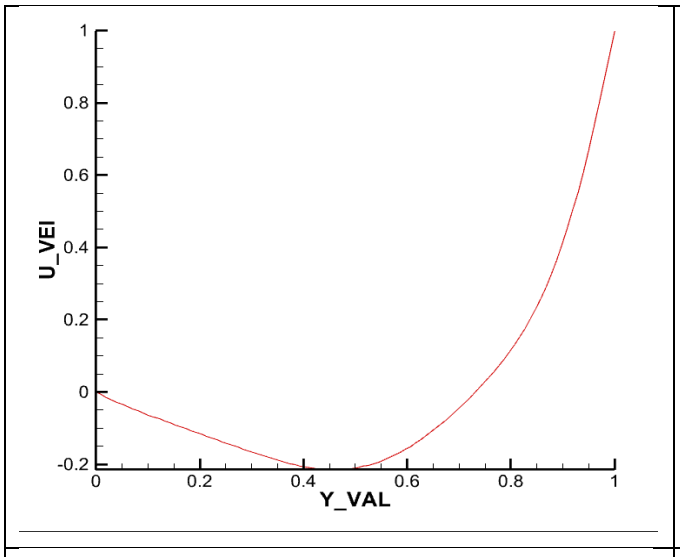


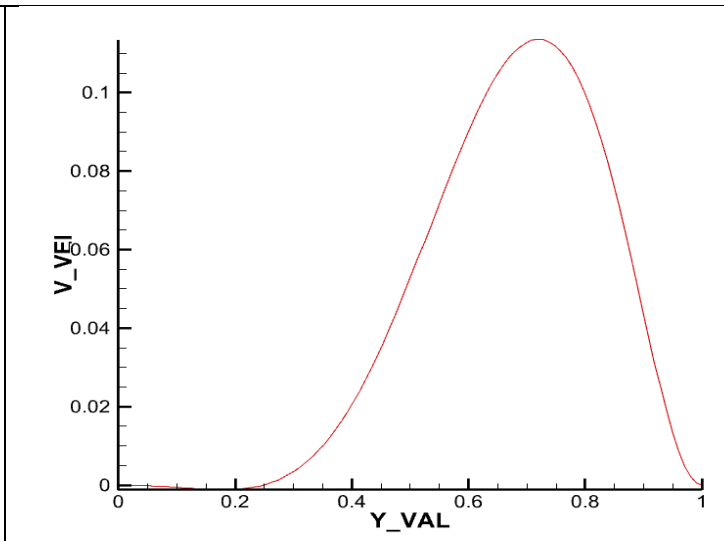


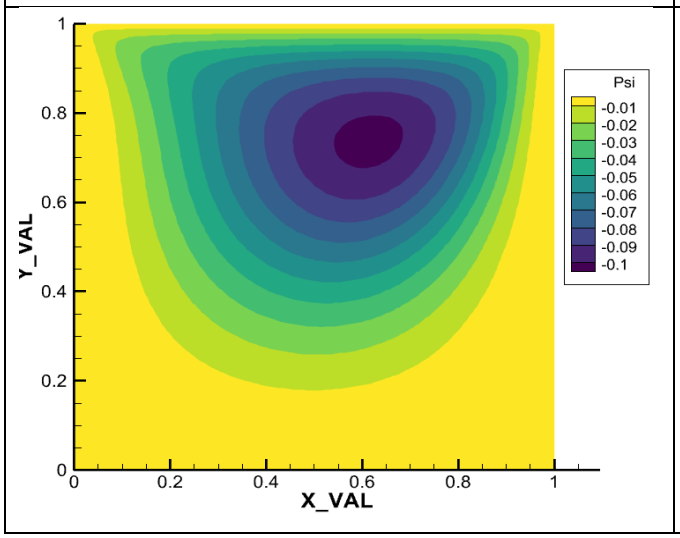


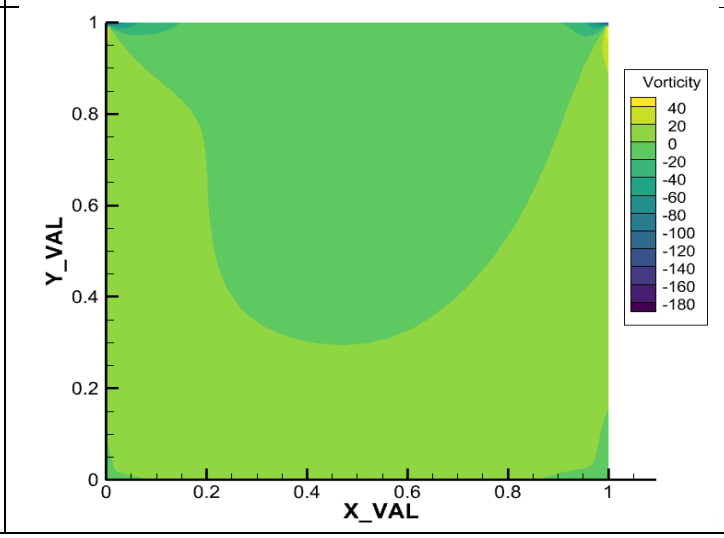
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -

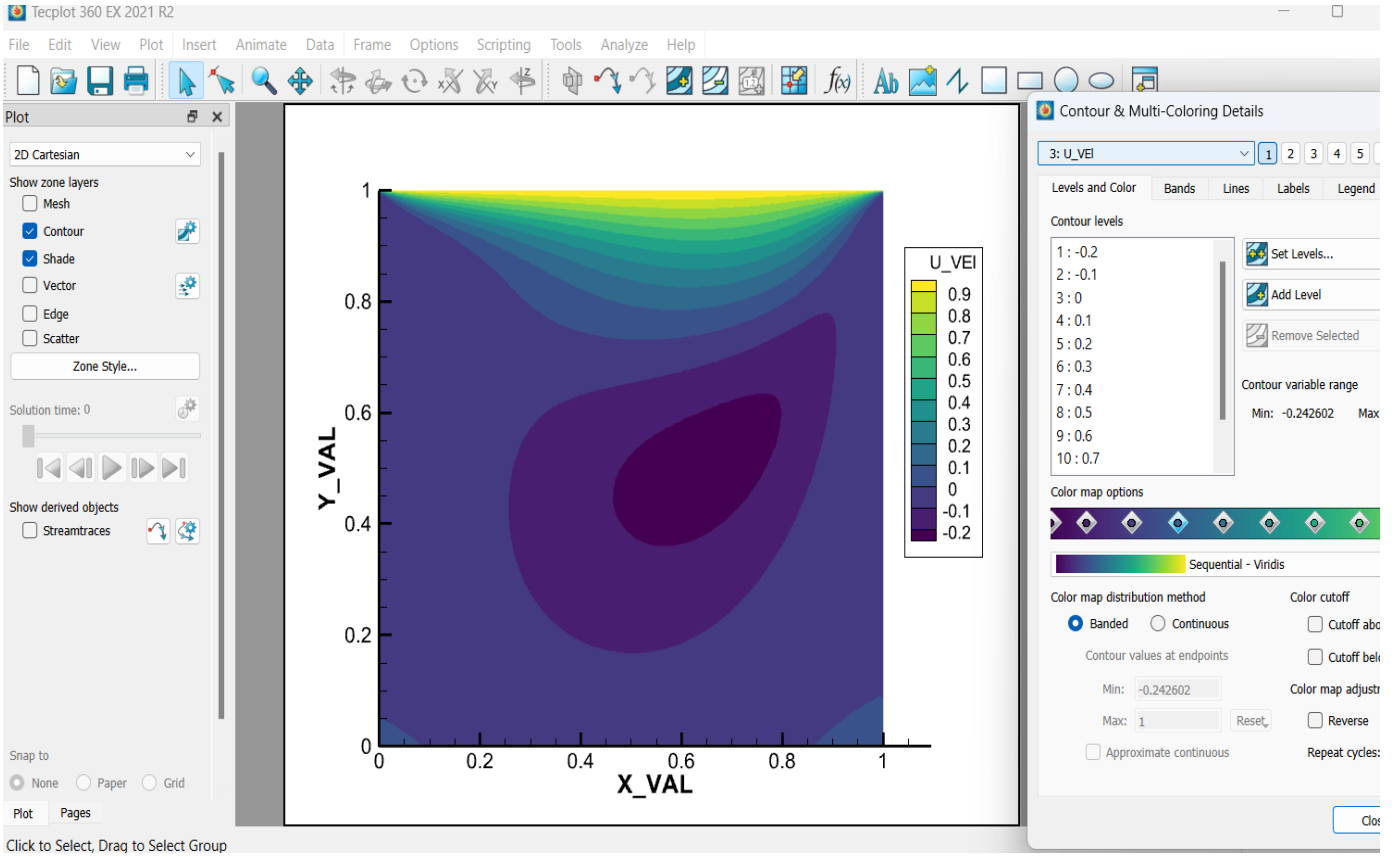


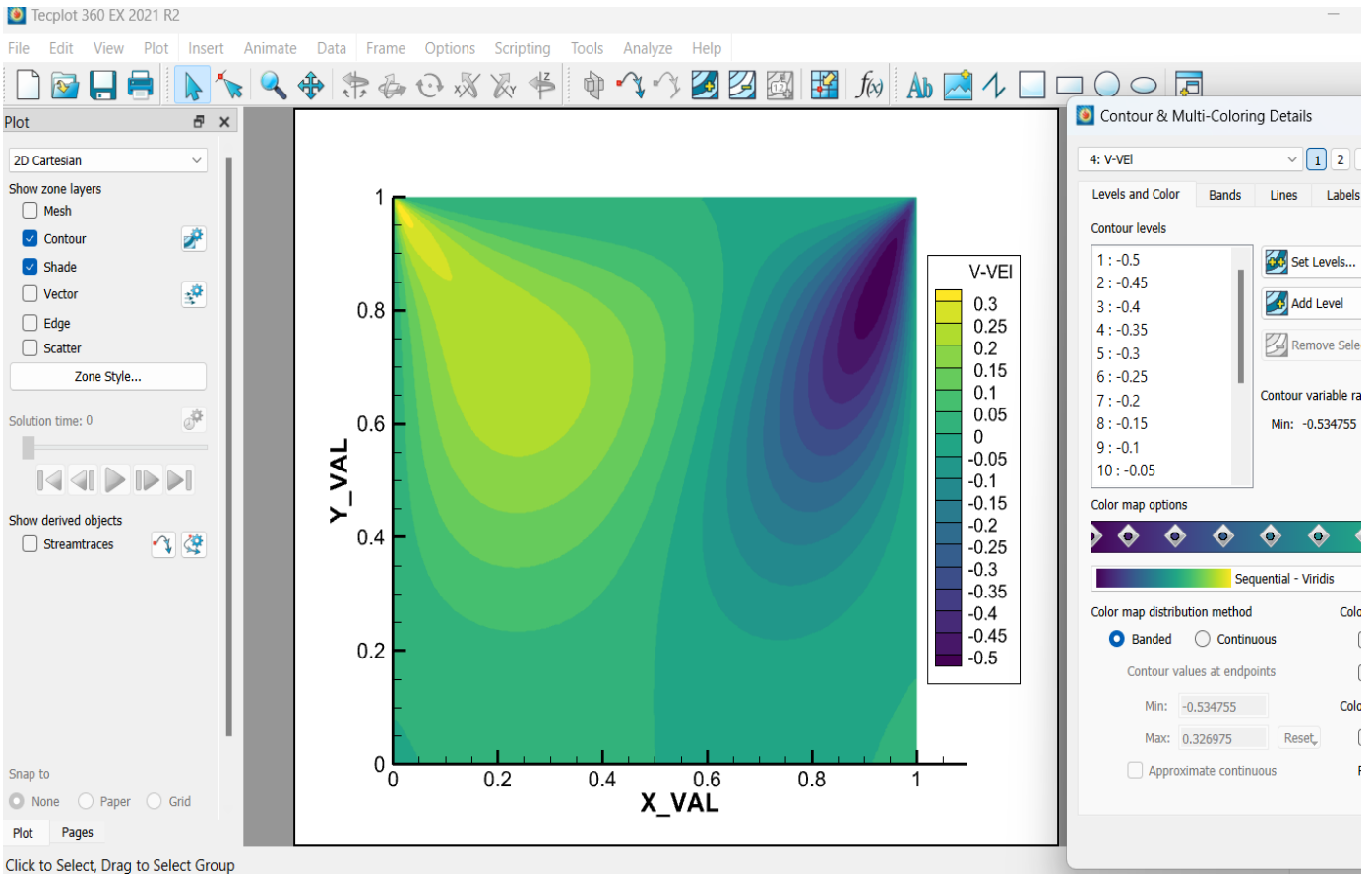


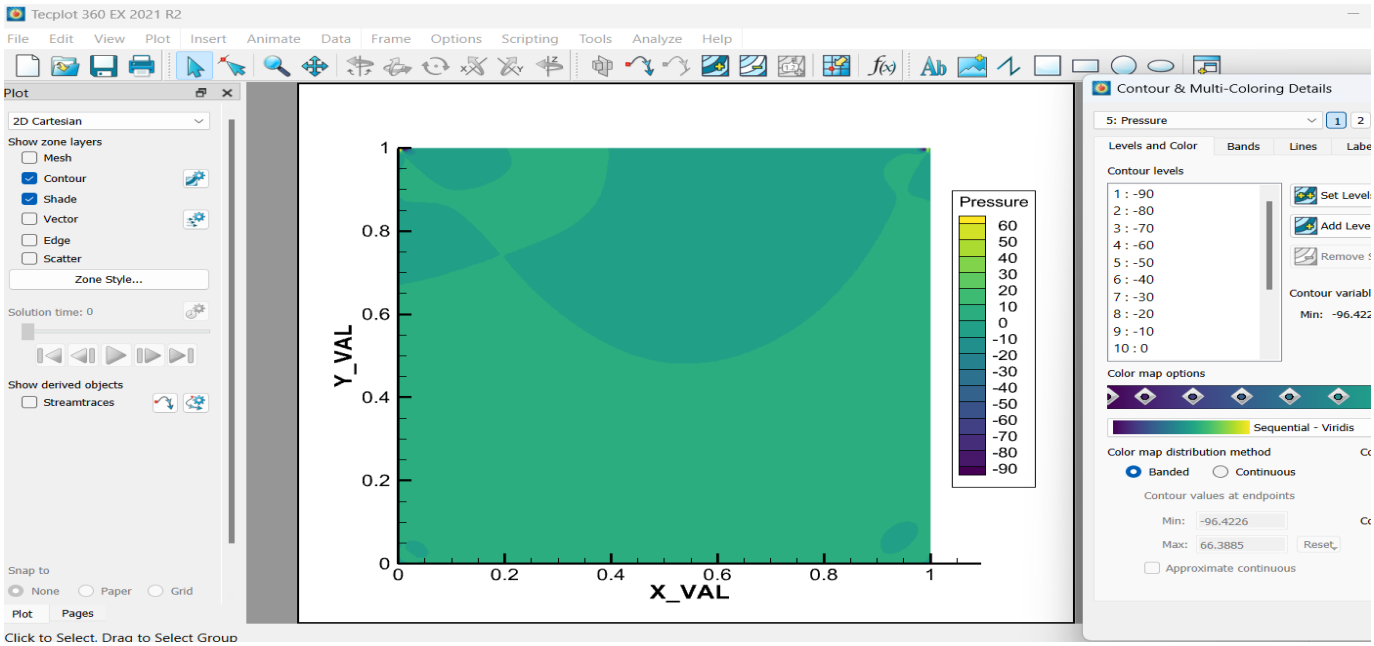




→Grid Size=200x200

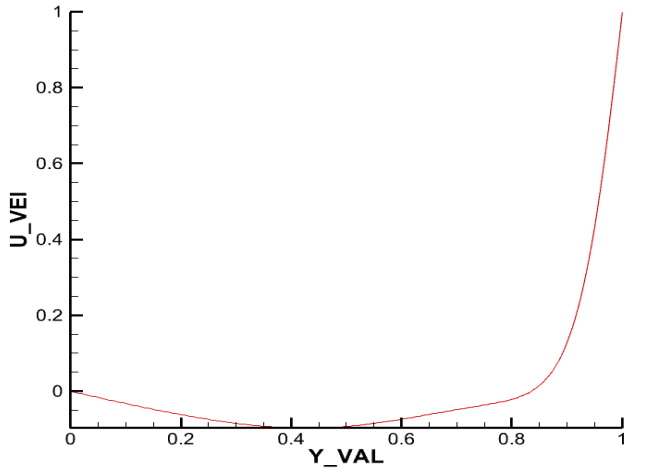


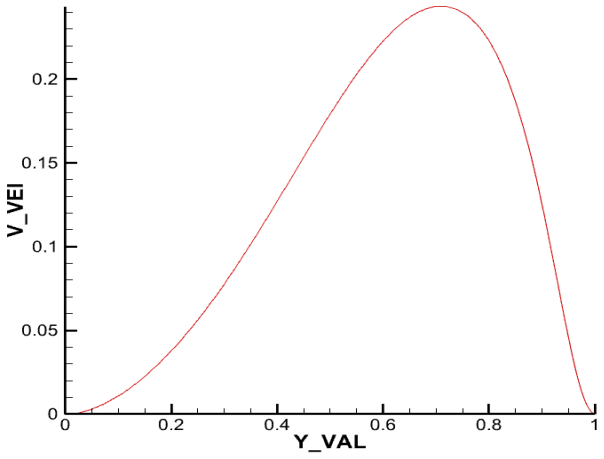


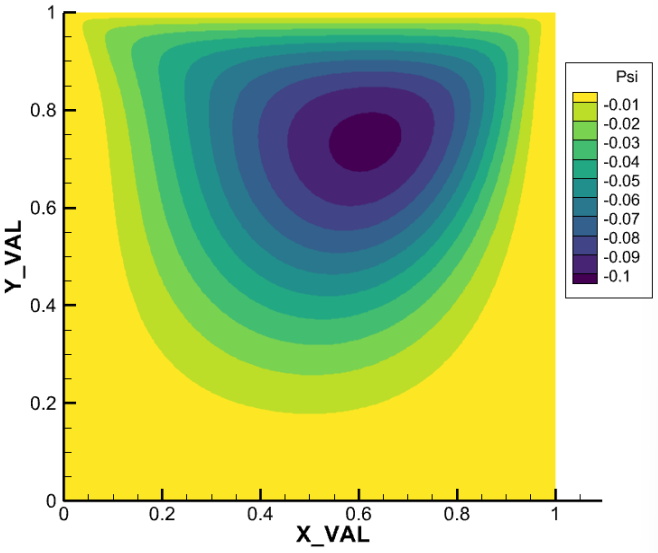


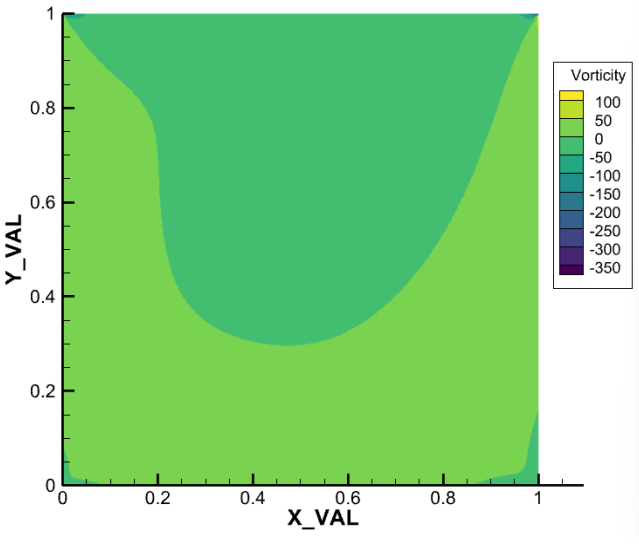
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -



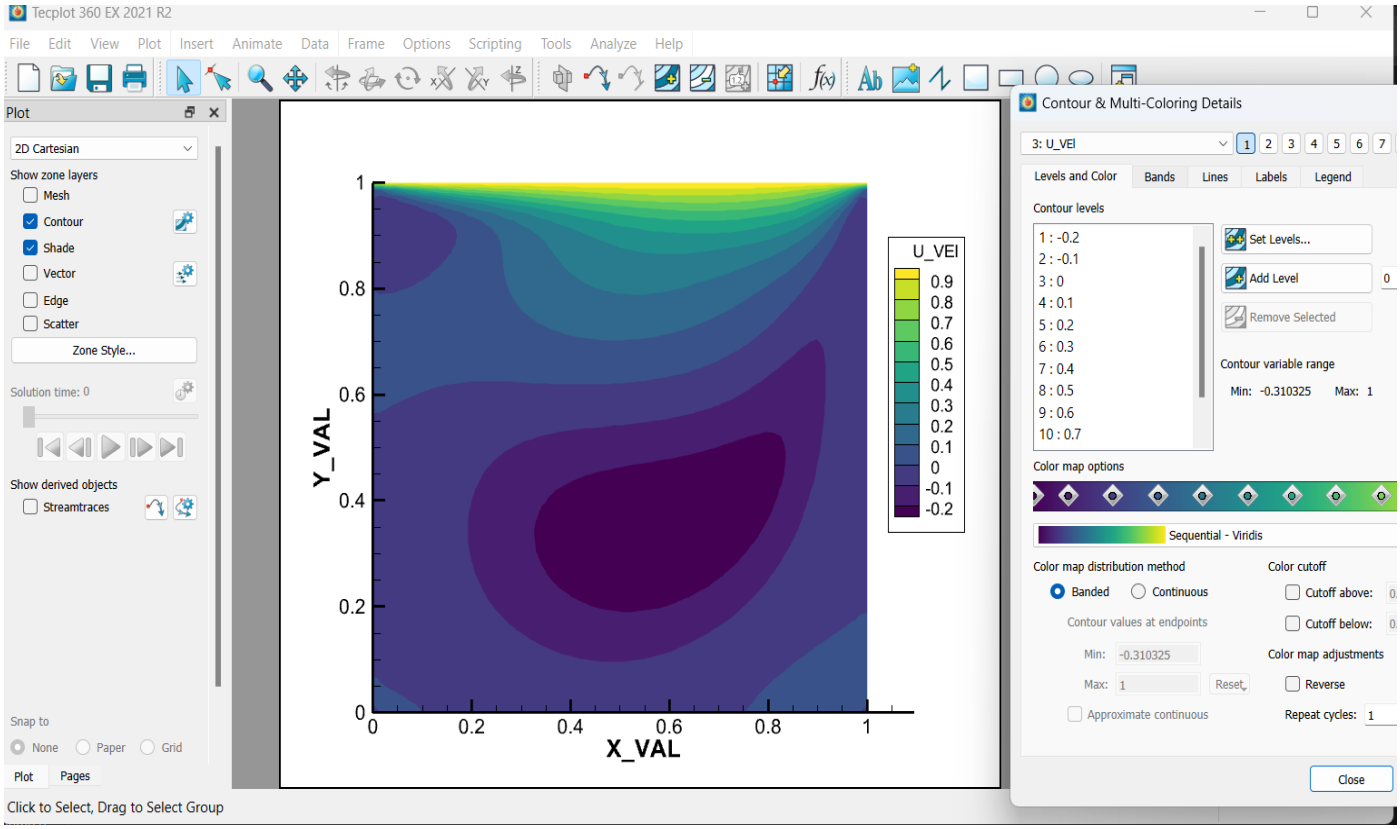


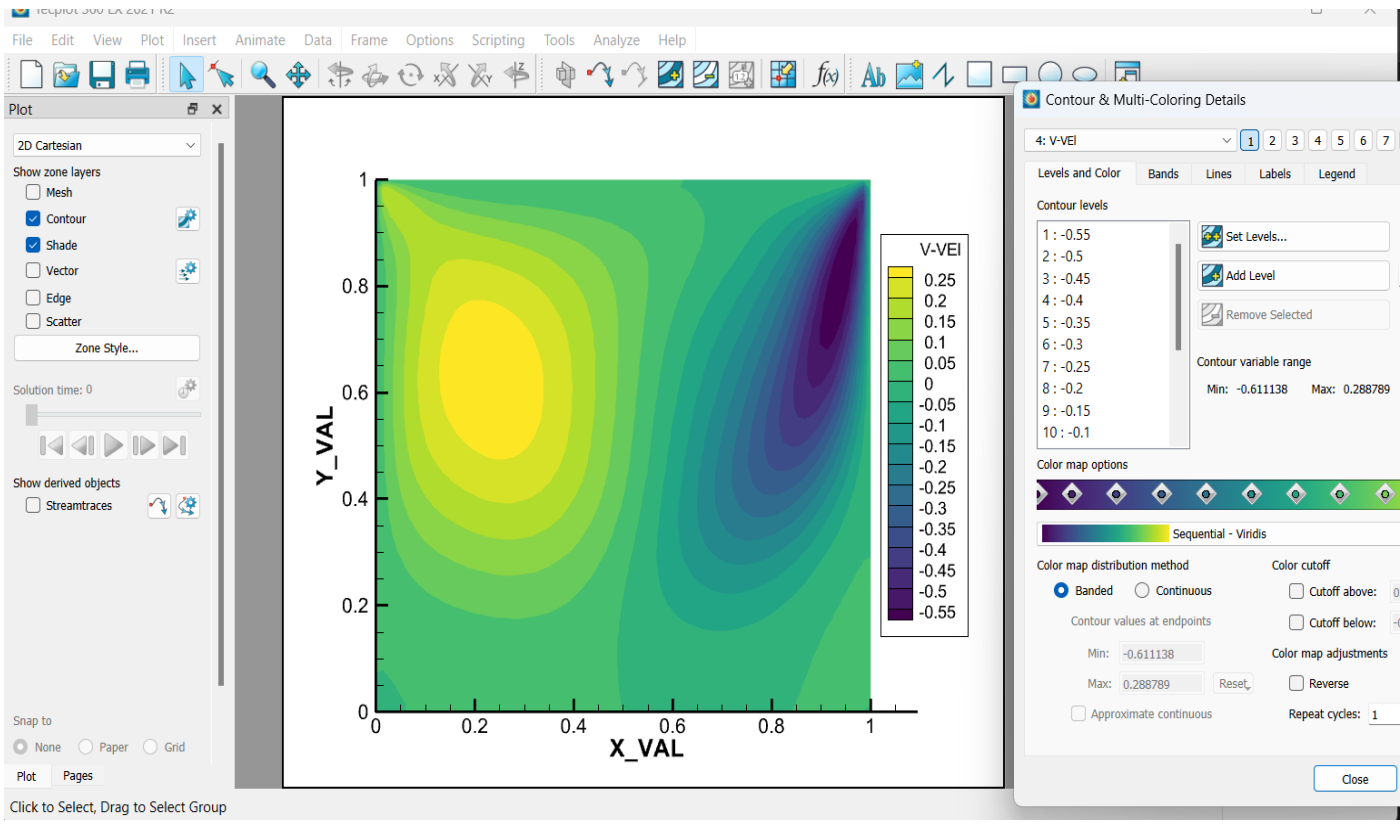


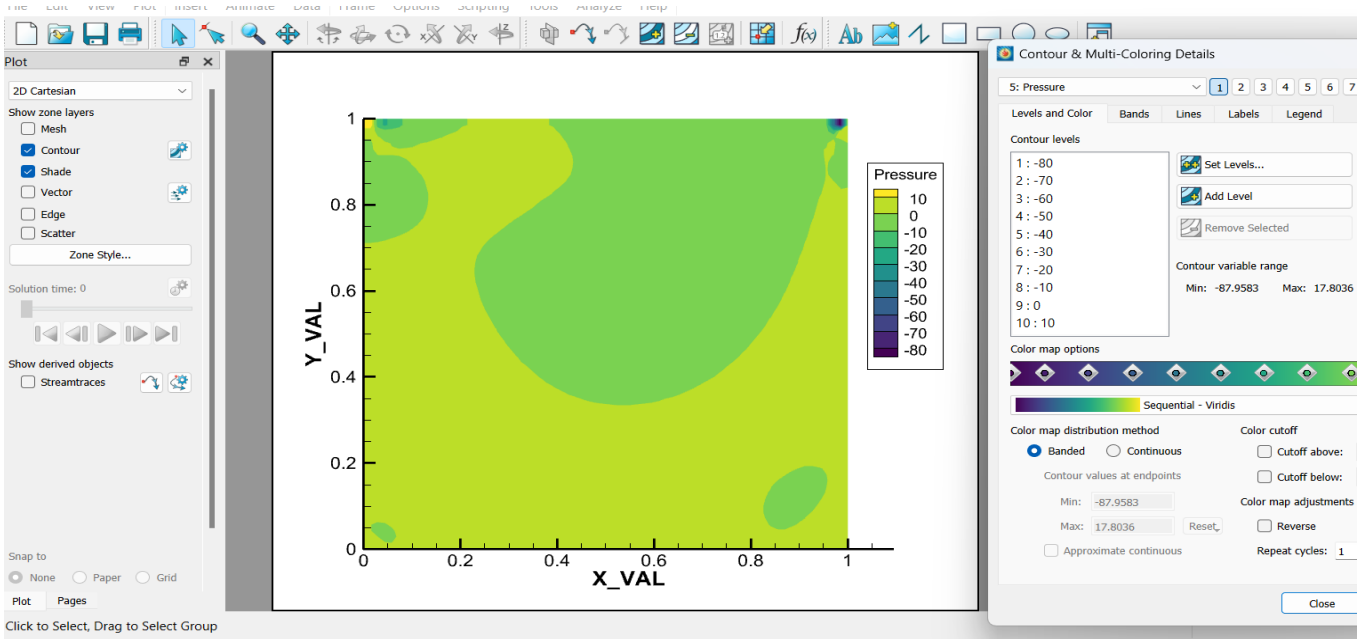


2)Reynolds's Number=300

→Grid Size=70x70

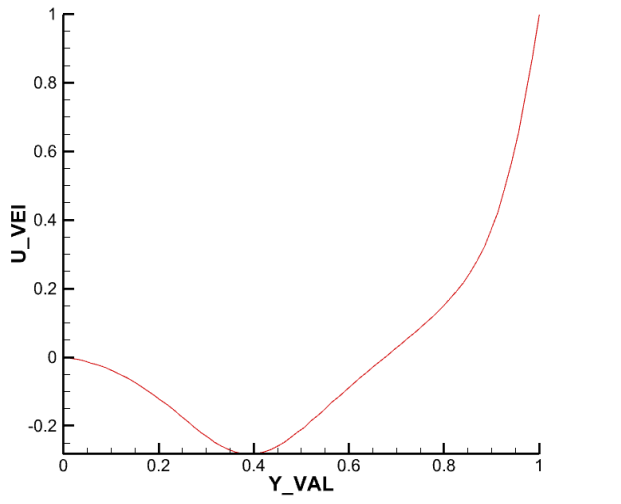


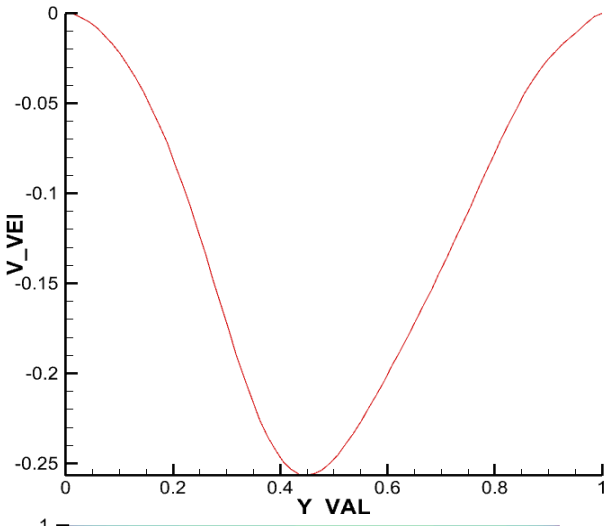


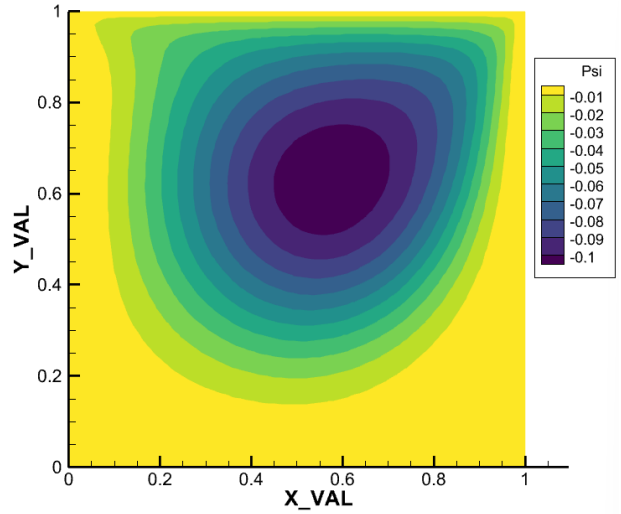


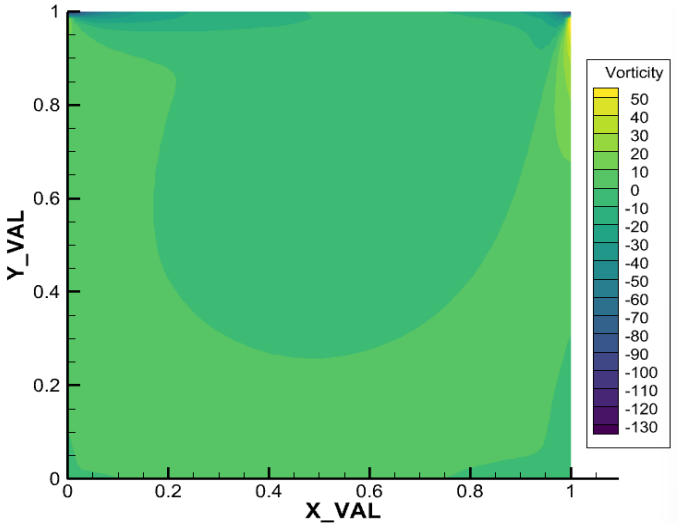
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -

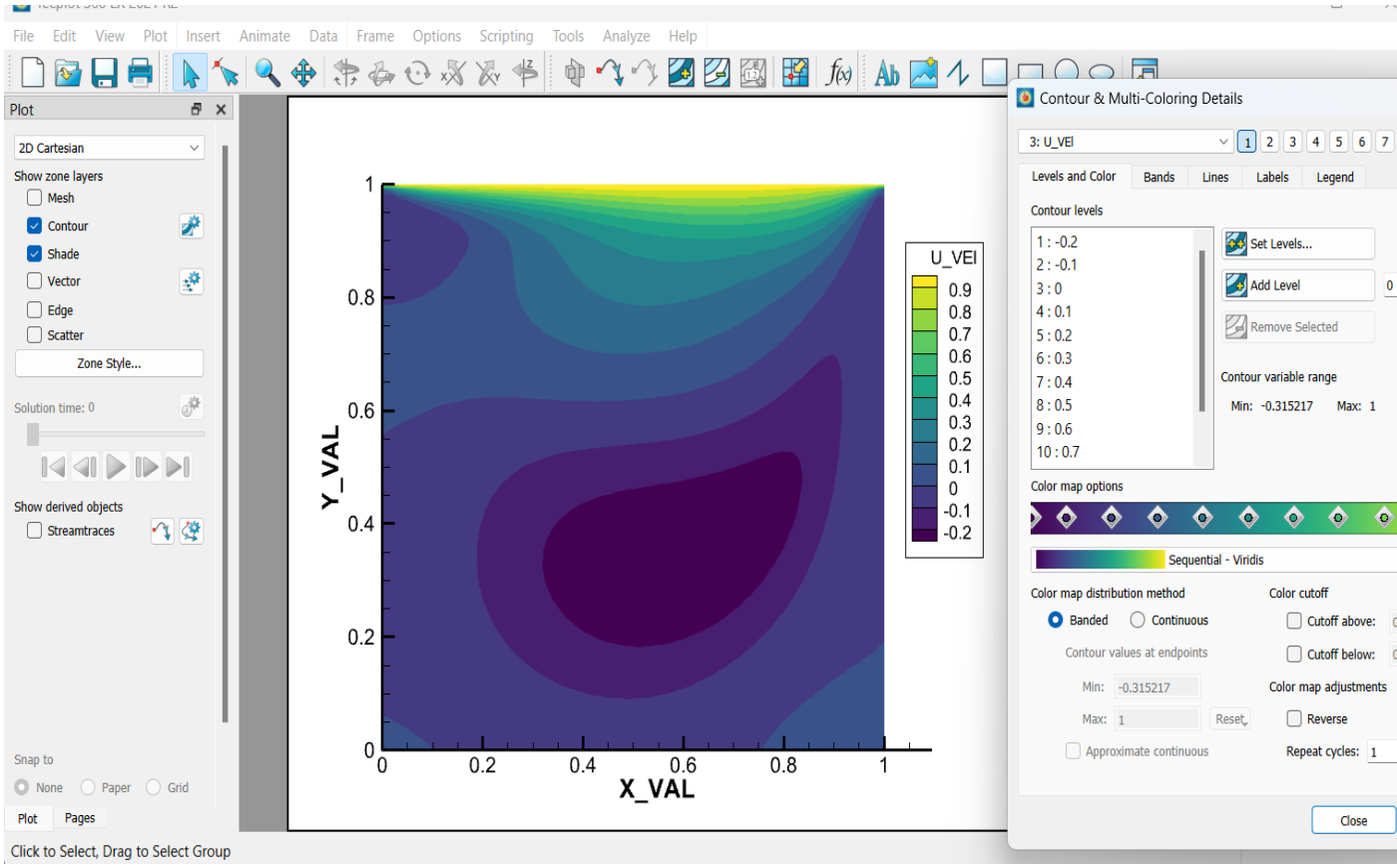


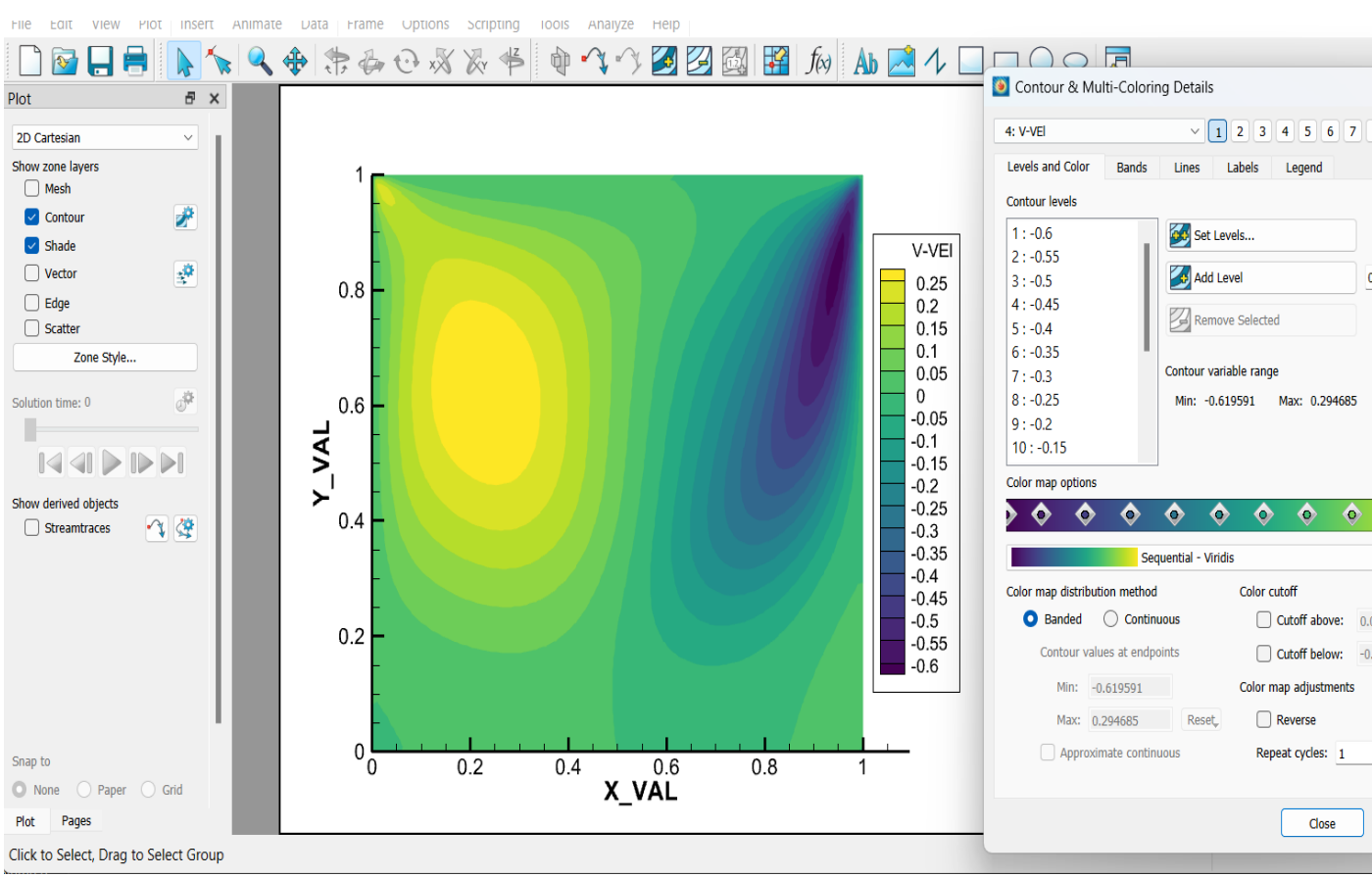


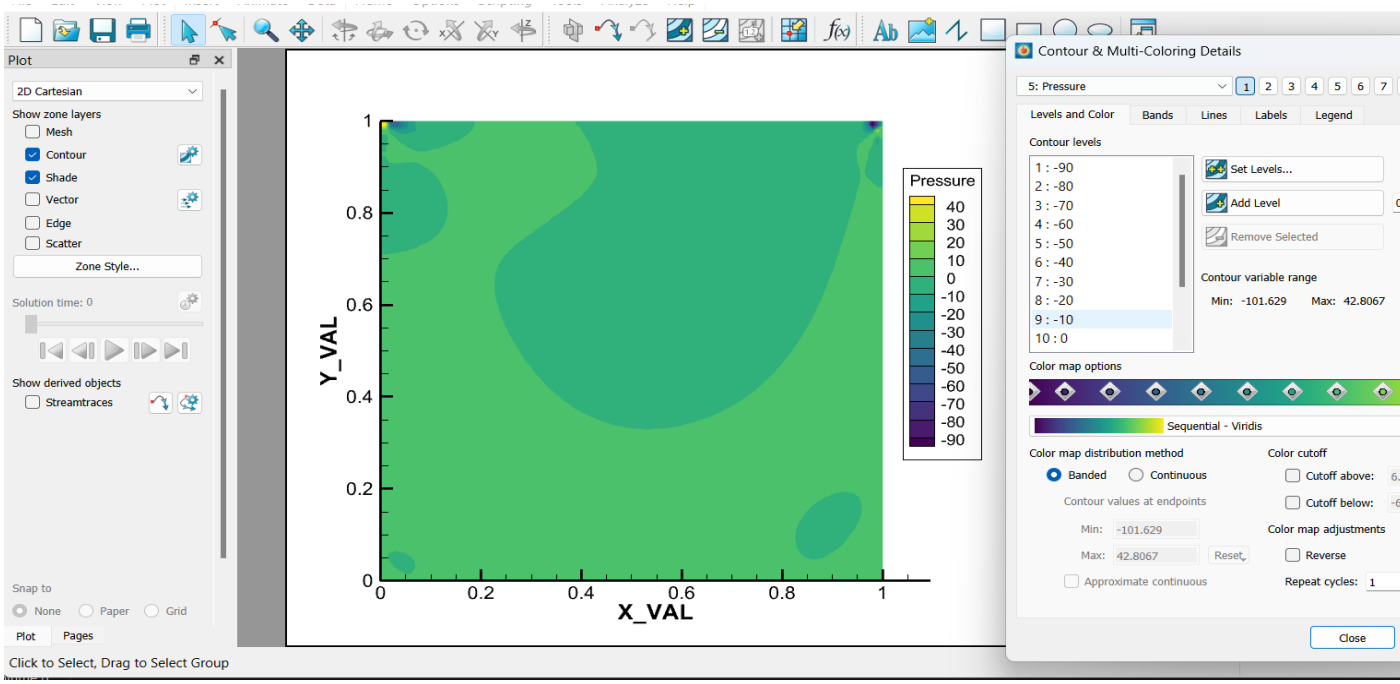




→Grid Size=100x100

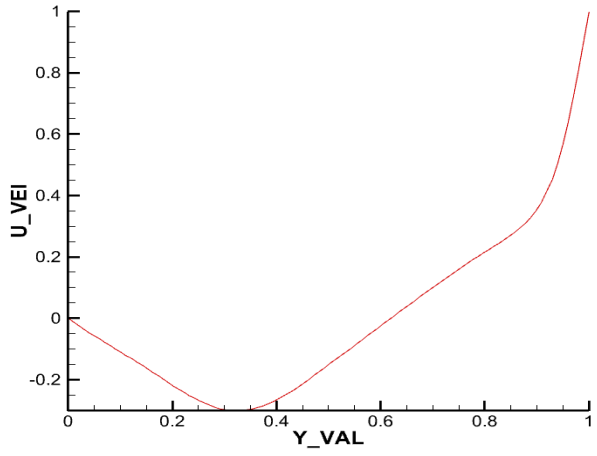


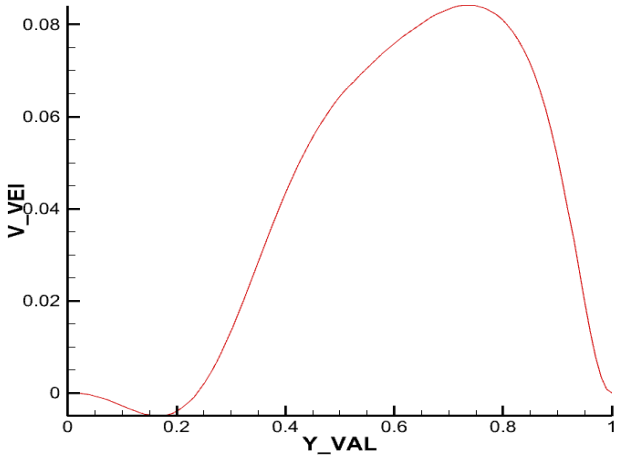


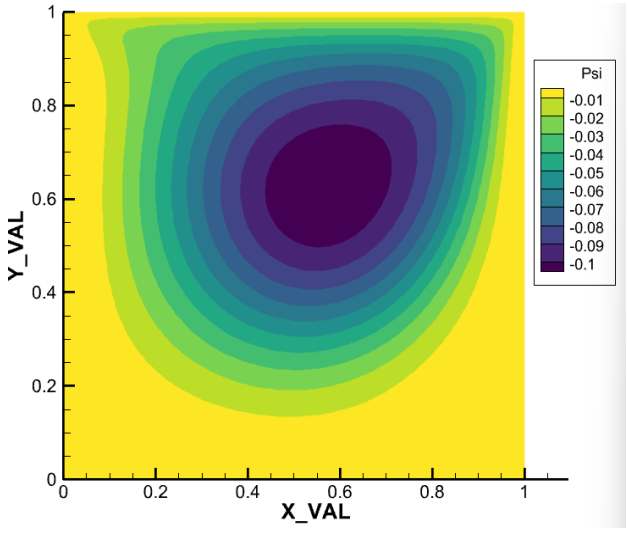


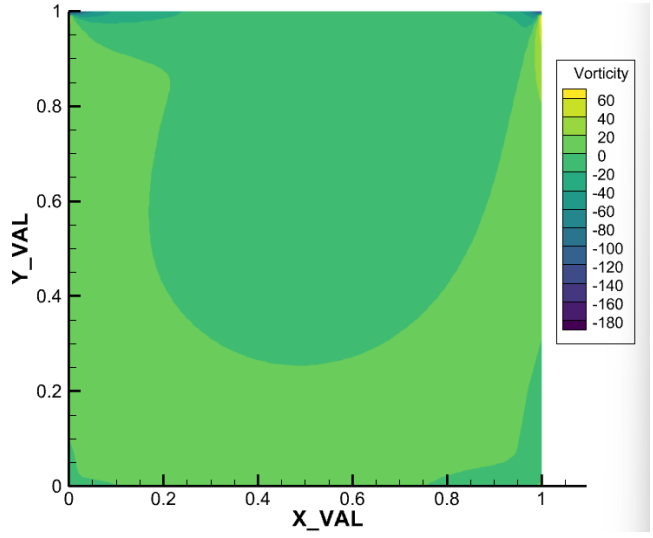
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -

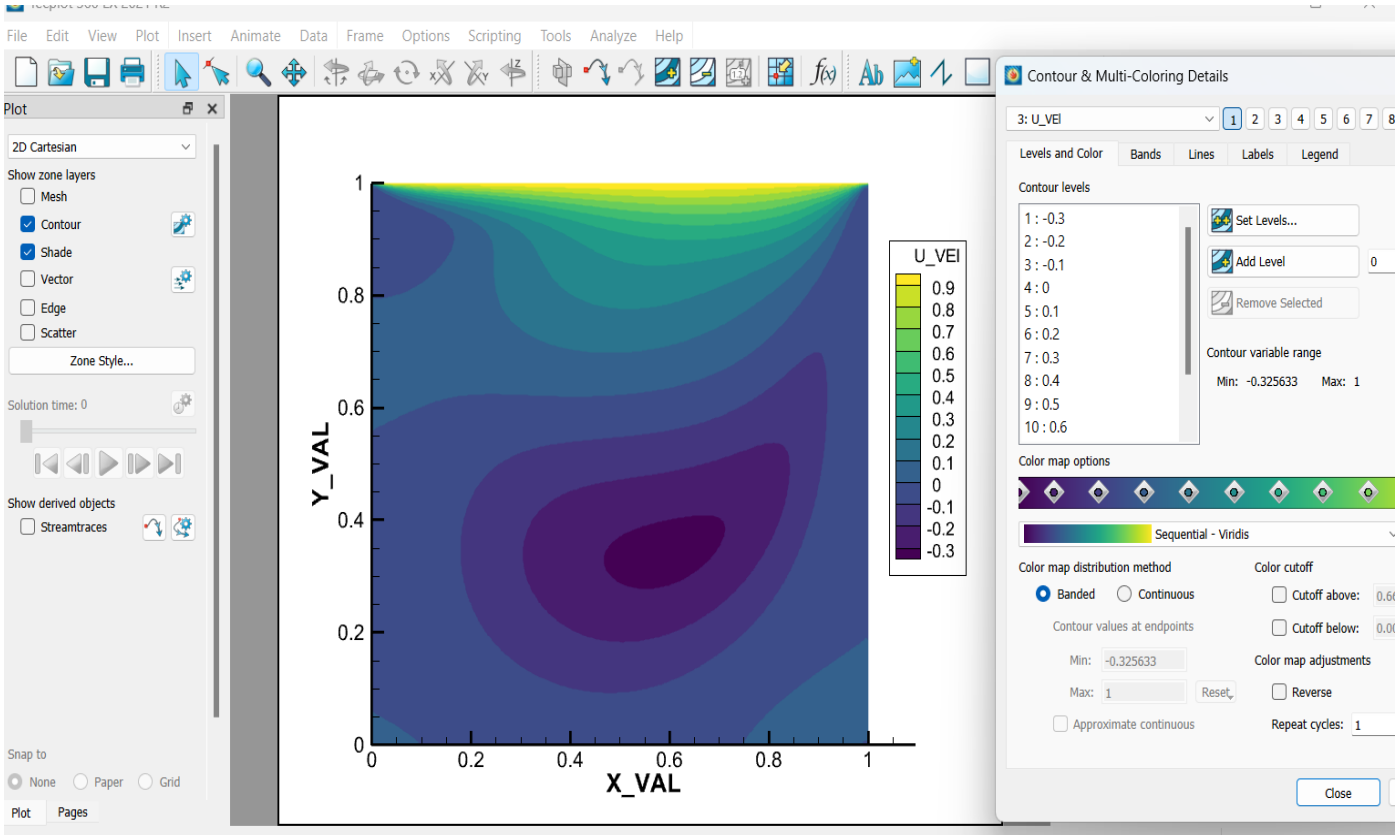


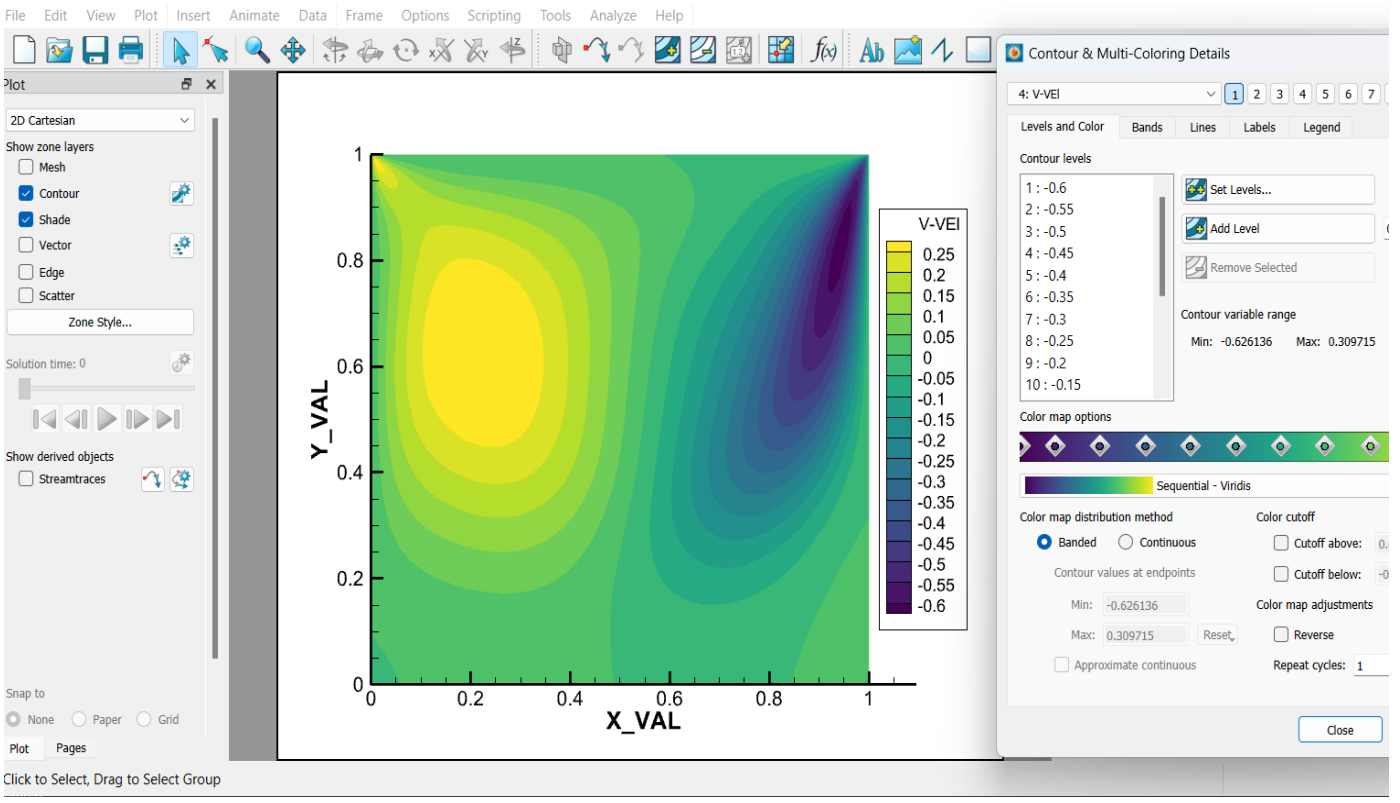


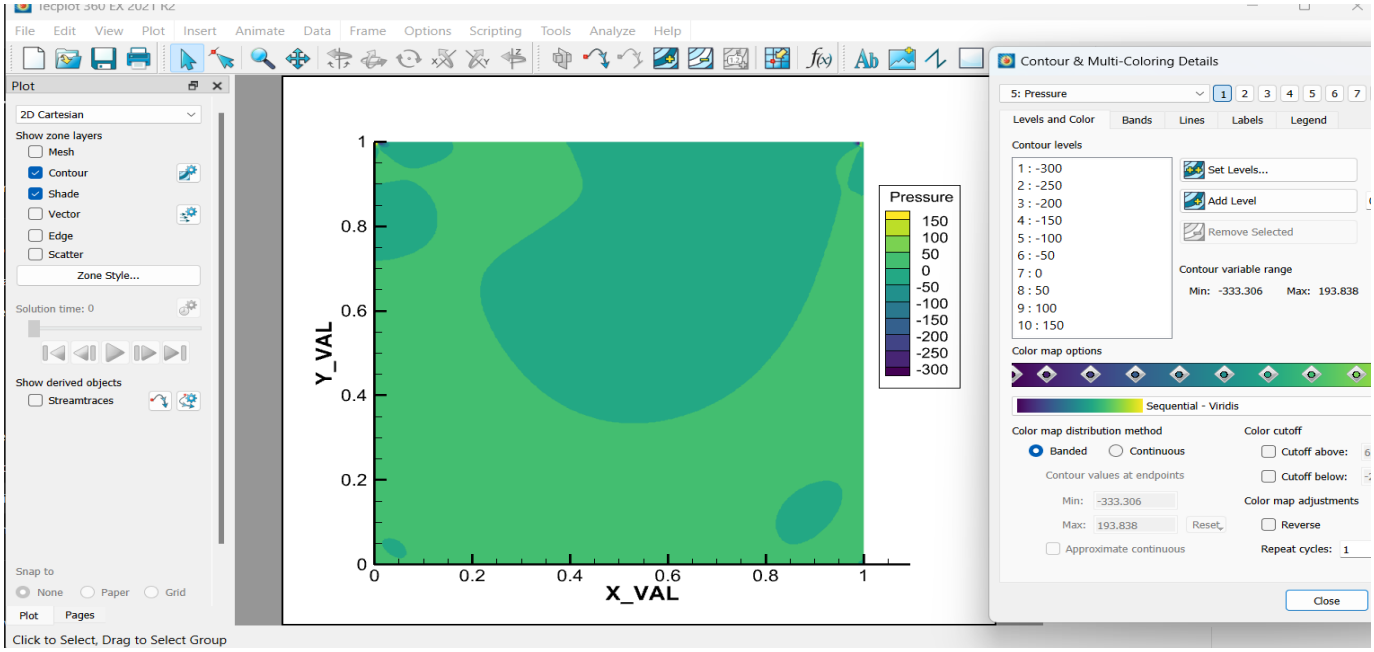




→Grid Size=200x200

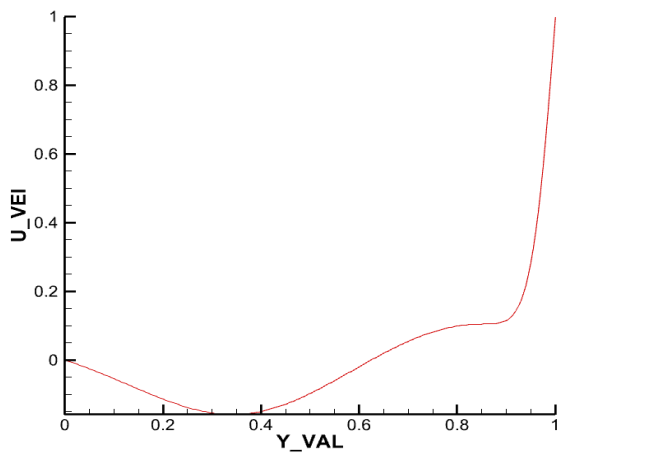


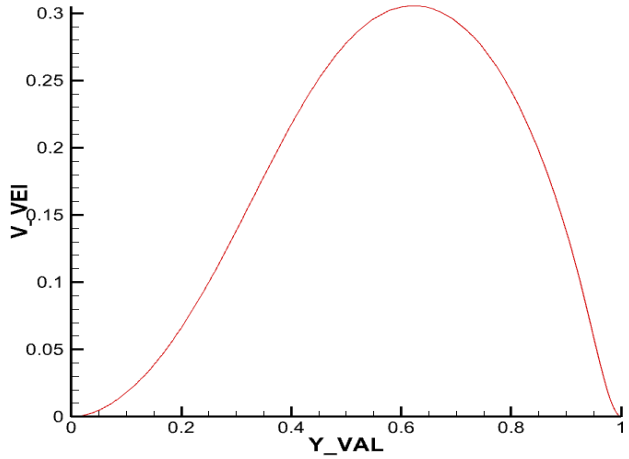


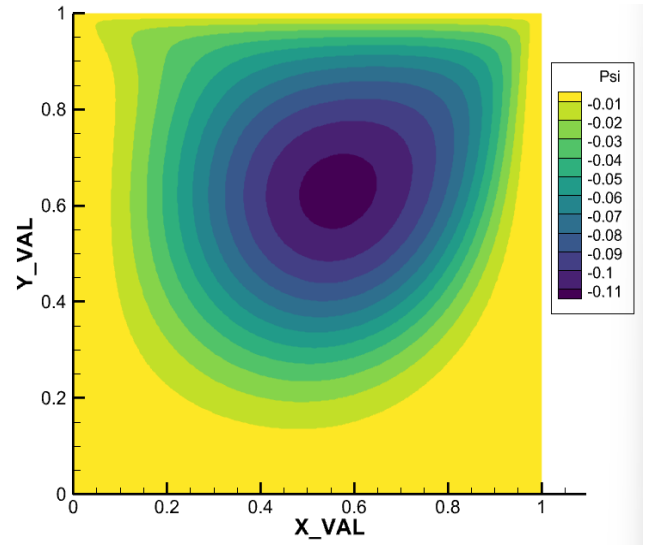


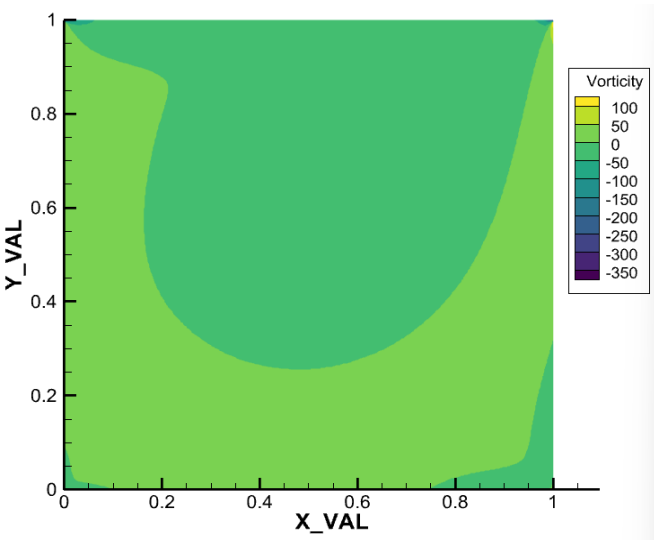
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -



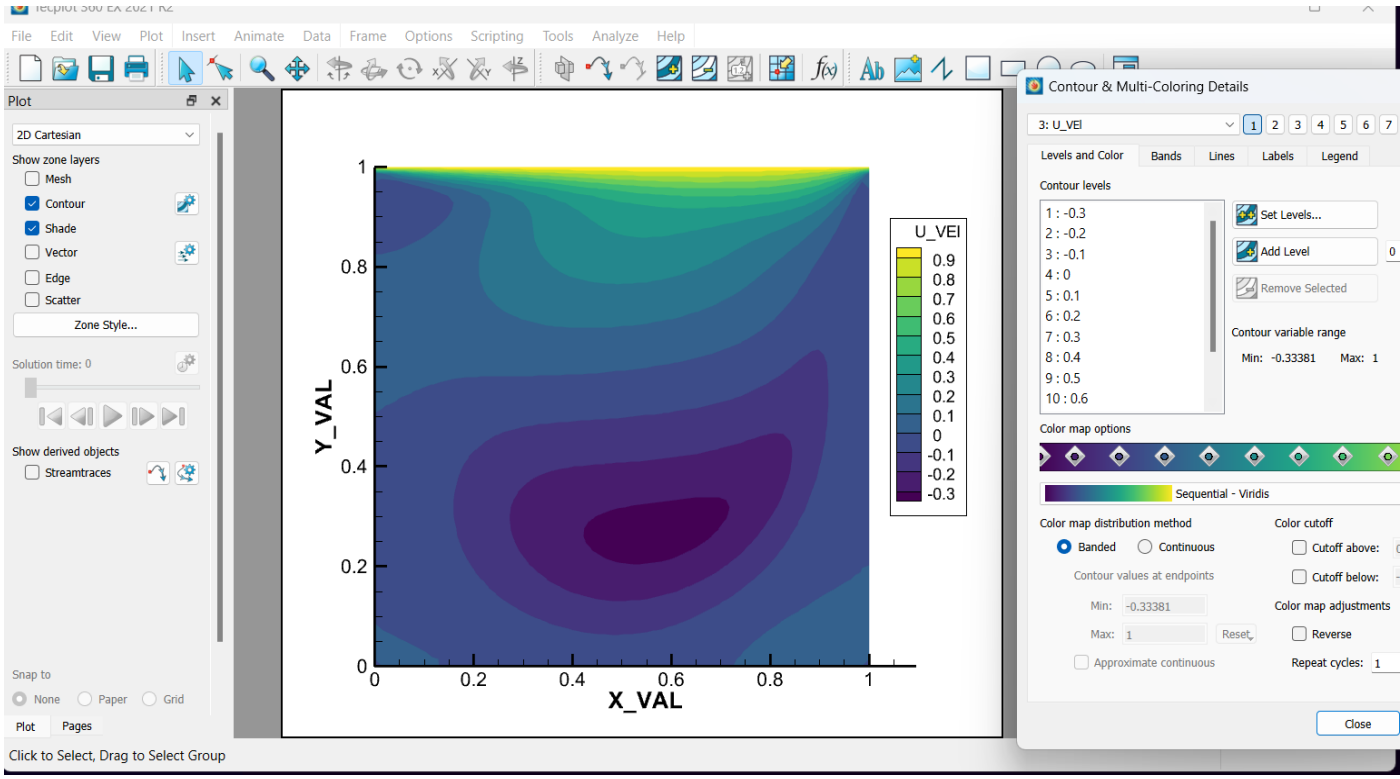


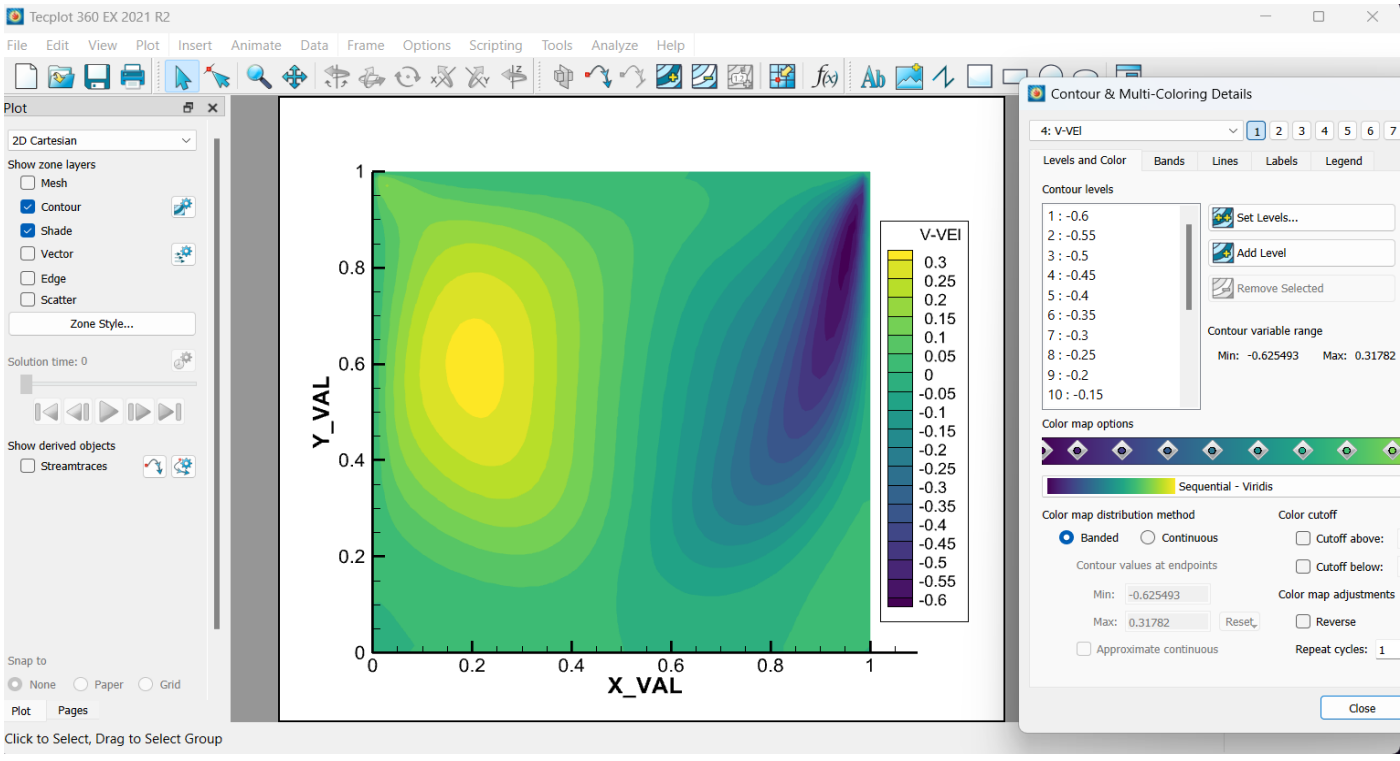


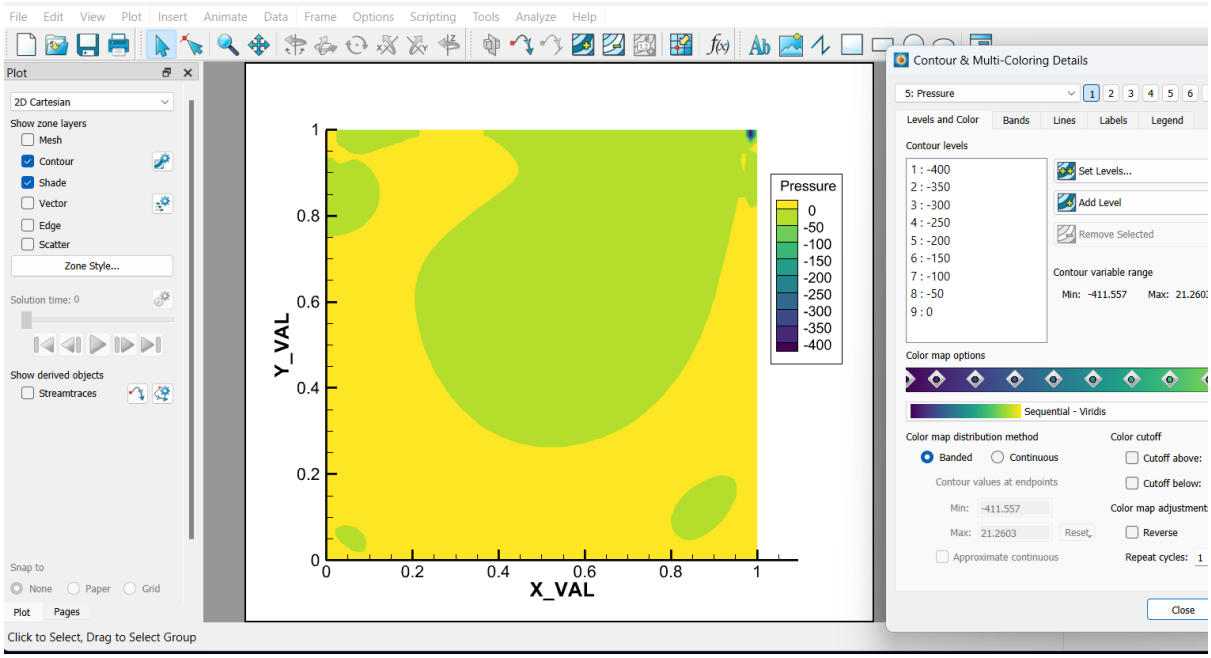


3)Reynolds's Number=500

→Grid Size=70x70

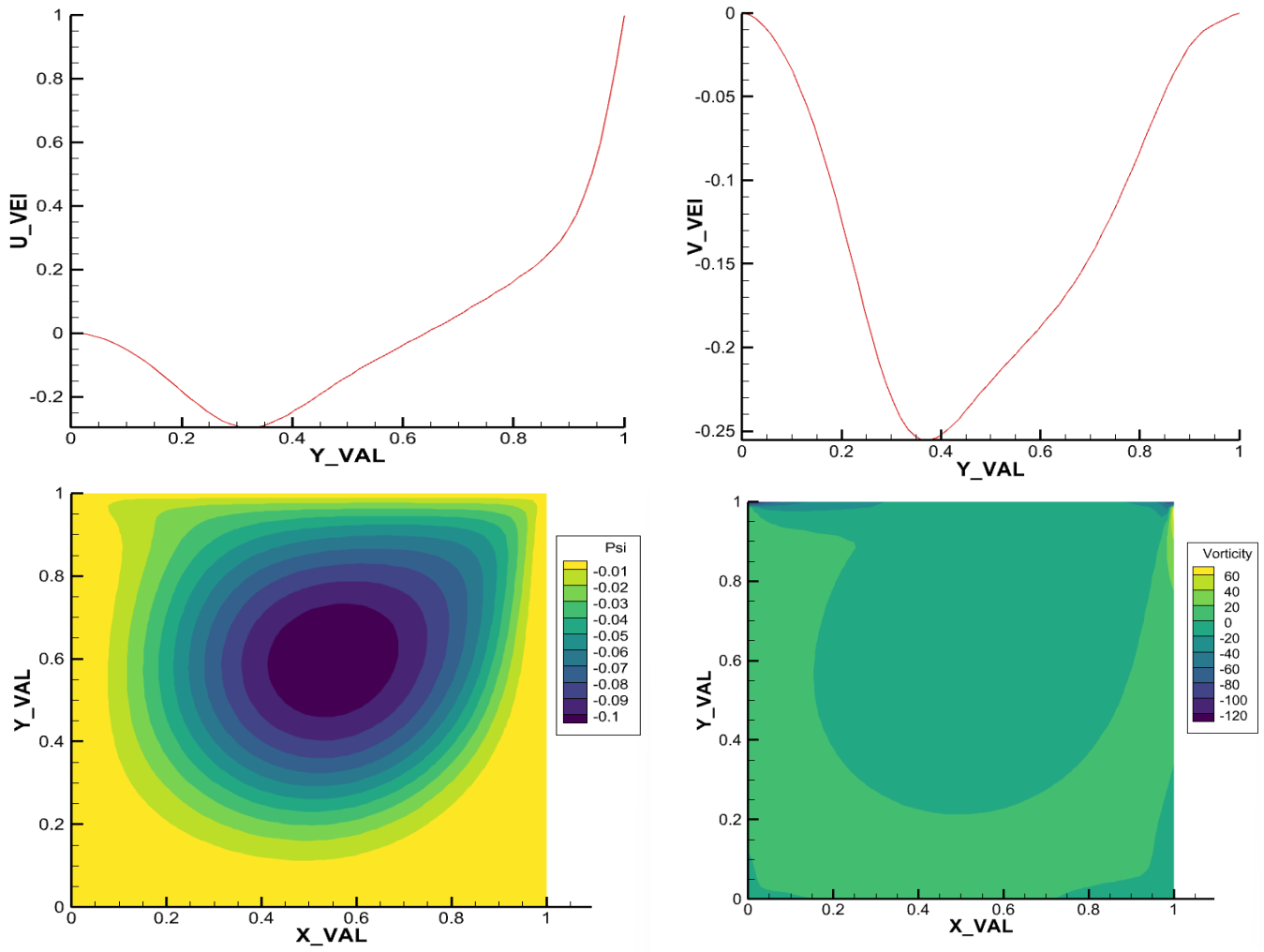




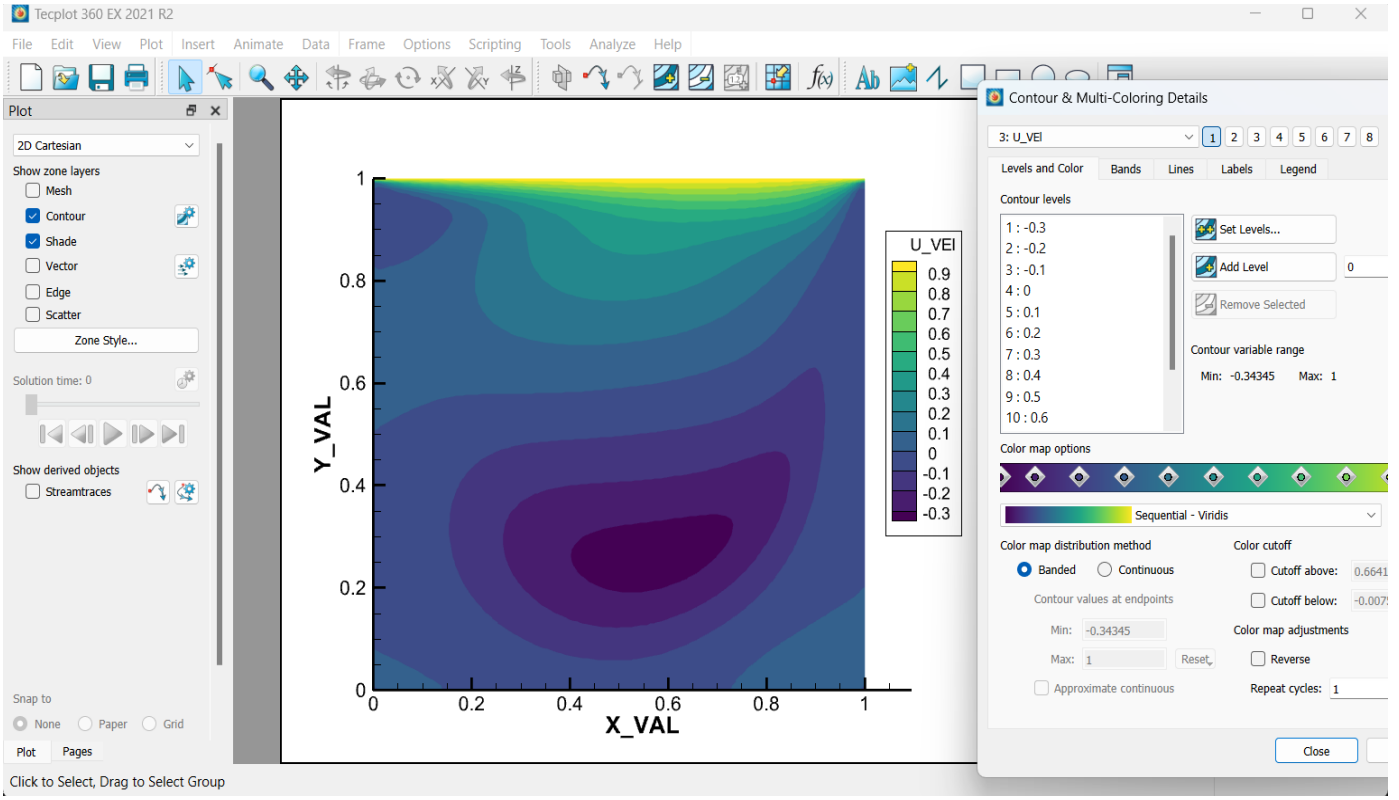


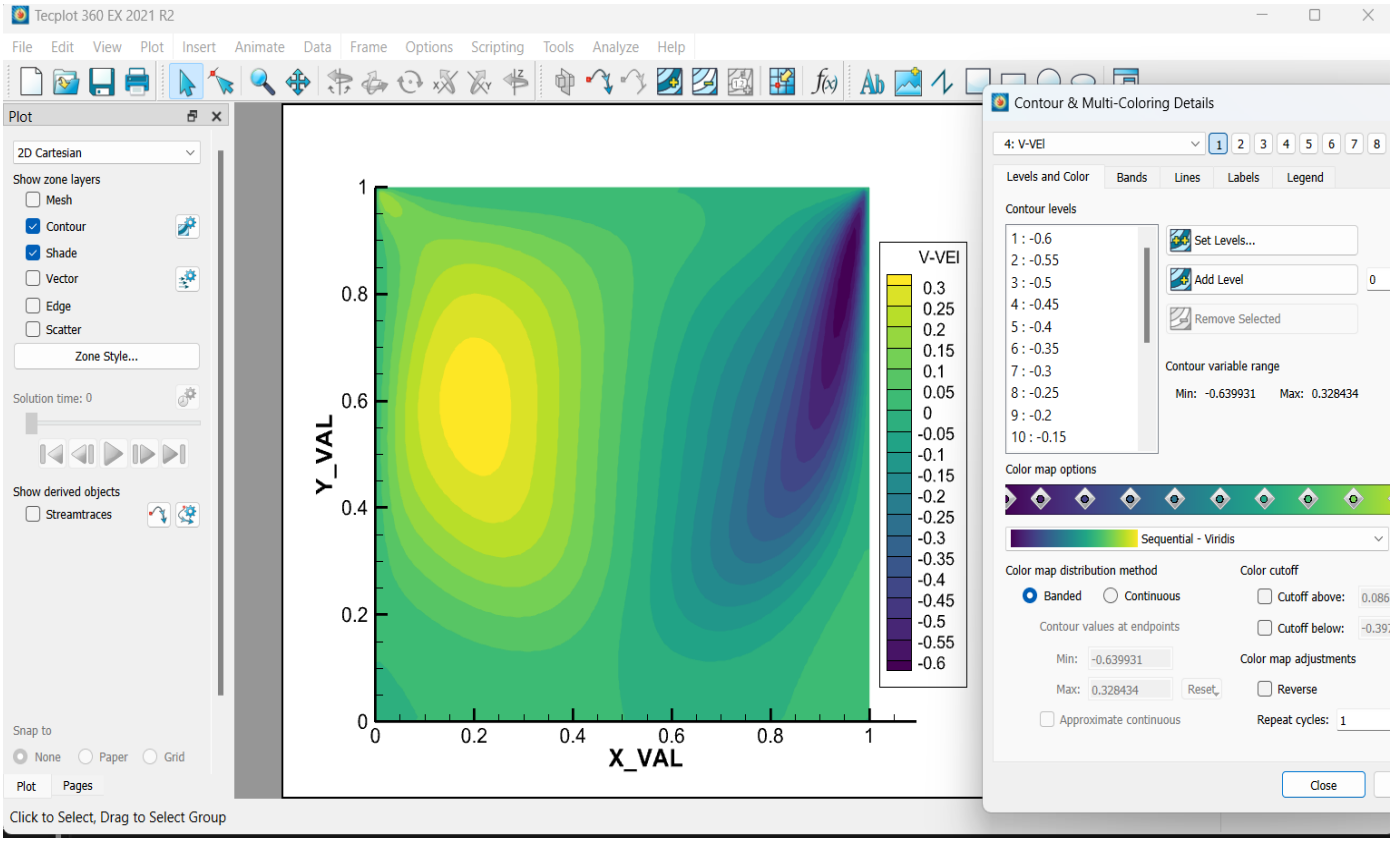
(U and V) vs (Mid plane in Y-direction) plot

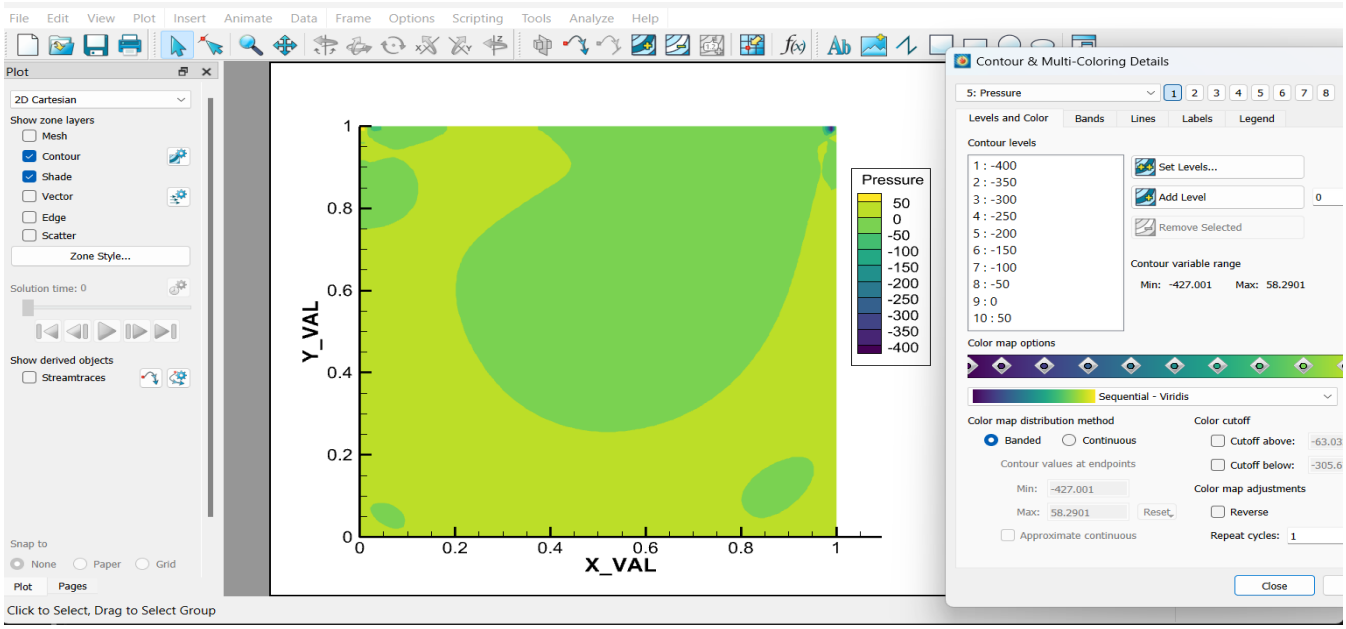
Stream Function and Vorticity Plot: -



→Grid Size=100x100

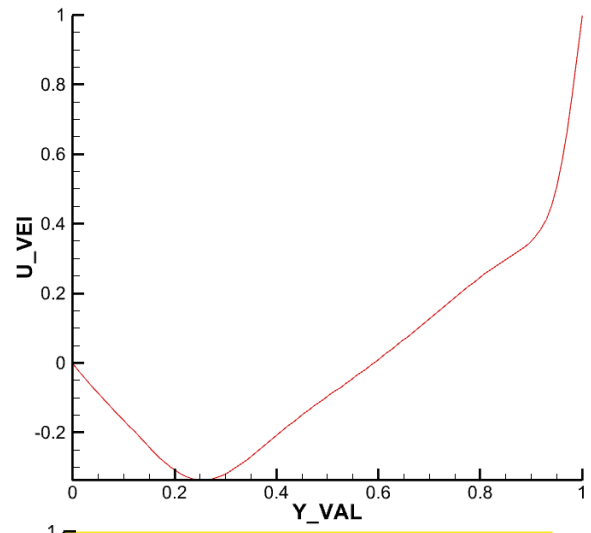


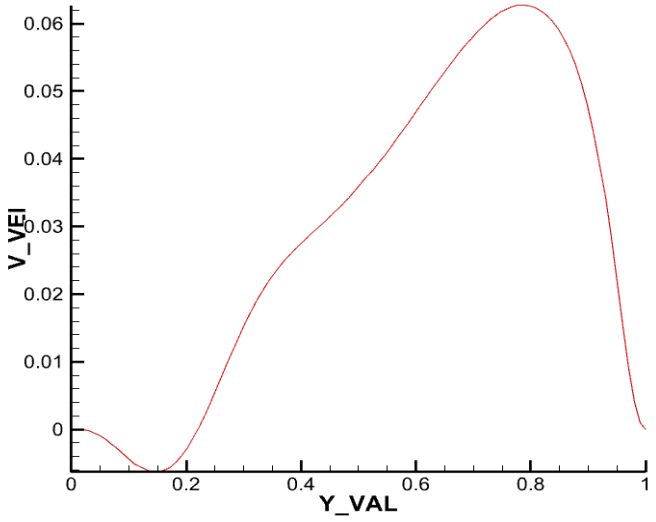


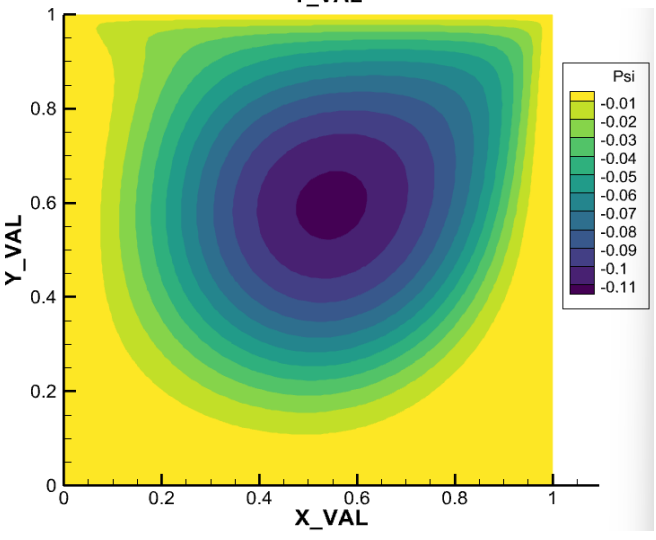


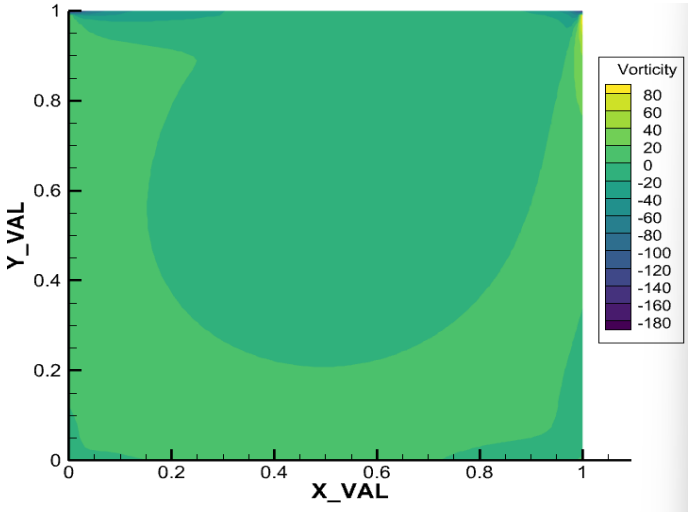
(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -

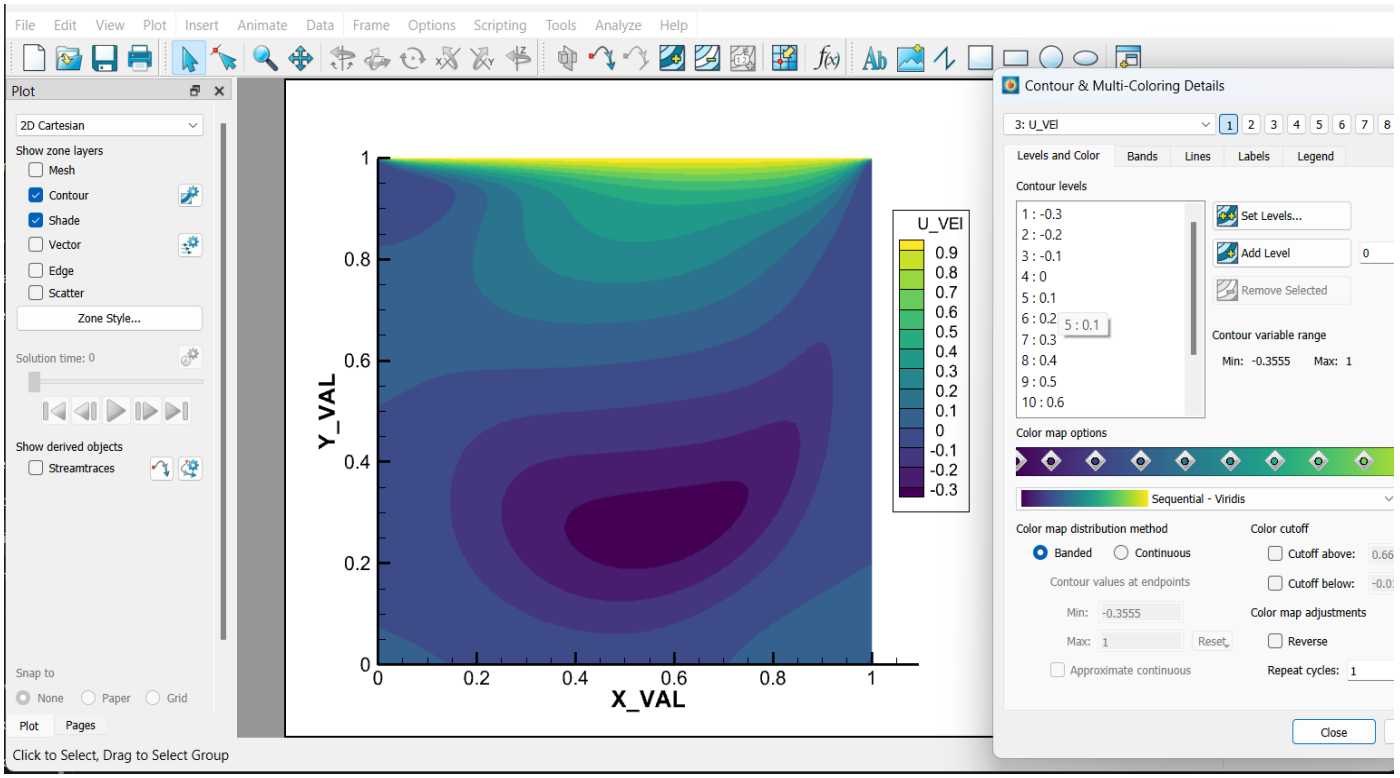


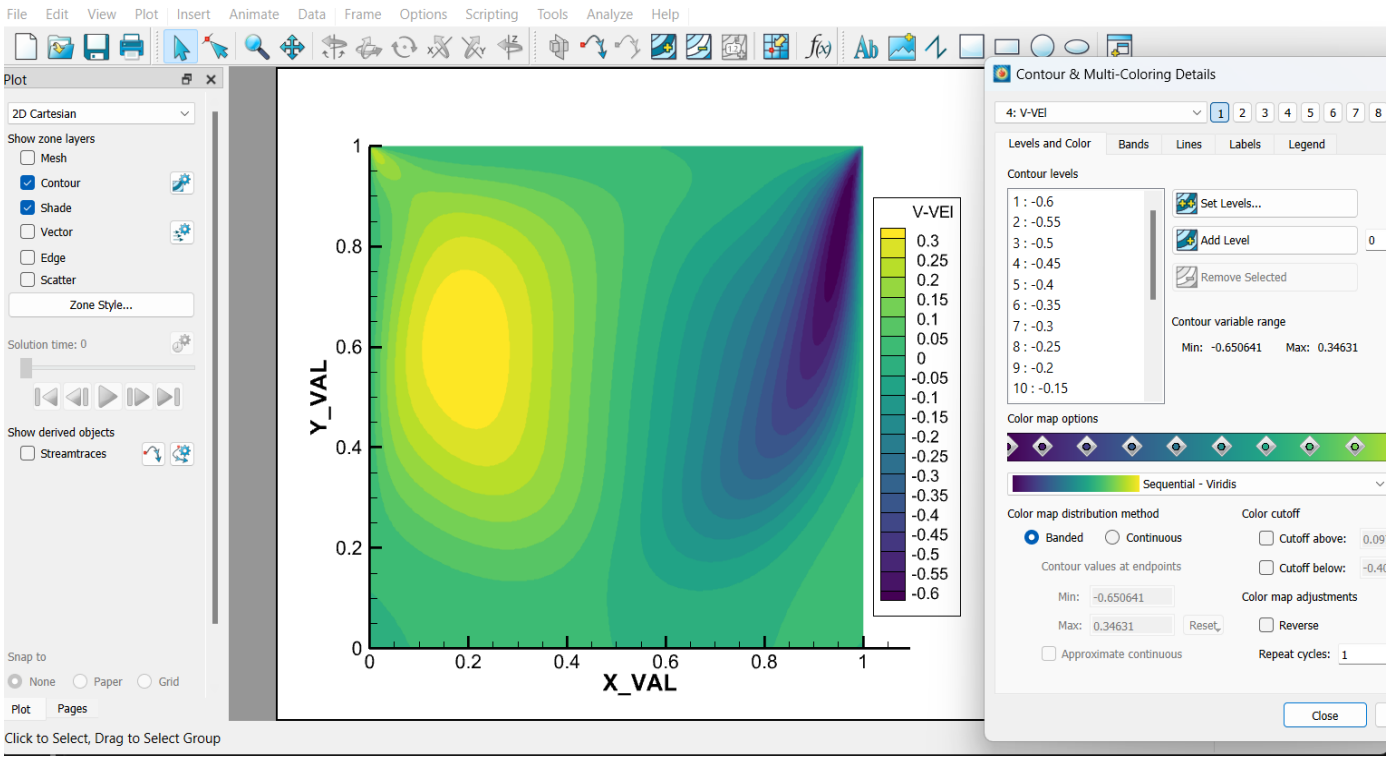


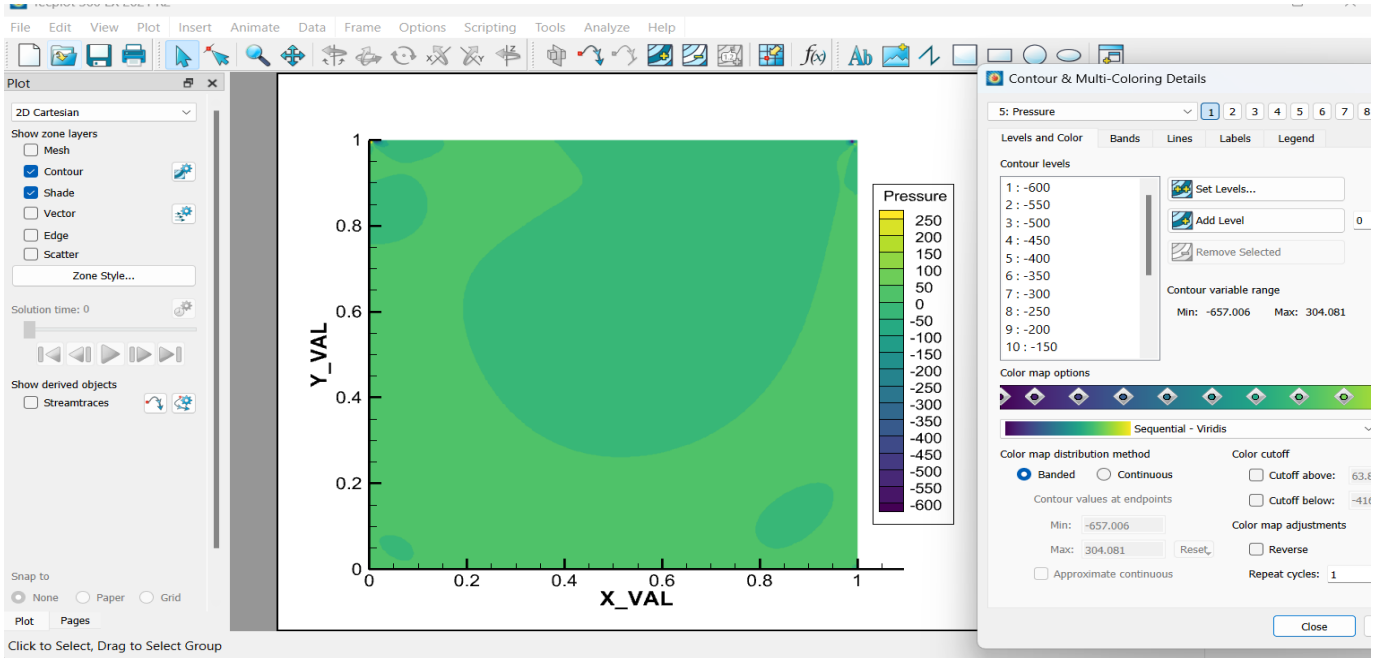




→Grid Size=200x200







(U and V) vs (Mid plane in Y-direction) plot

Stream Function and Vorticity Plot: -