Ministry of Education and Science of the Republic of Kazakhstan
Al-Farabi Kazakh National University
Faculty: “Mechanics and Mathematics”
Department: “Mathematical and computer modeling”

Project report

Theme: Solving the Navier-Stokes equation

Done by:

Kakibay A. K.

Checked by: Issakhov A.A.

.

Almaty 2020

1. **Construction of a mathematical model:**

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

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

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0\ (3)$$

With the initial conditions at $t = 0$:

$$u = 0$$

$$v = 0$$

$$p = 0$$

Here: (1) is equation of motion by x; (2) is equation of motion by y; (3) is the continuous equation.

U and V are the components of velocity by x and y, P is pressure, ρ is density and Re is dimensionless quantity characterizing the proportion of inertial forces to viscous friction forces in viscous liquids and gases.

2. **Statement of the problem with respect to boundary conditions.**

We'll solve the Navier-Stokes equations with the following boundary conditions.

It has to be mentioned, that our mesh has sizes:

$n = 101$ by X axis (indexes by i)

$m = 51$ by Y axis (indexes by j)

At left boundary:

$$p = 0$$

$$u\left(x = 0, 0 < y < \frac{m}{2}\right) = 0$$

$$v = 0$$

Since on the higher 2nd part of left boundary we have entrance:

$$u\left(x = 0, \frac{m}{2} < y < m\right) = u_0 = 1$$

At top boundary:

$$\frac{\partial p}{\partial n} = 0$$

$$u = v = 0$$

At bottom boundary:

$$\frac{\partial p}{\partial n} = 0$$

$$u = v = 0$$

At right boundary:

$$p\left(x = n, 0 < y < \frac{m}{2}\right) = 0$$

Since on the higher 2nd part of right boundary we have exit boundary conditions will be as following.

At $\frac{m}{2} \leq y < m$:

$$p_{n-1,j} = 0$$

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x}|_{x=n-1} = 0$$

$$u_{n-1,j} = u_{n-2,j}$$

$$\frac{\partial v}{\partial n} = \frac{\partial v}{\partial x}|_{x=n-1} = 0$$

$$v_{n-1,j} = v_{n-2,j}$$

Also, as we can notice on the graph above, there are barriers in the form of 2 houses.

Boundary conditions at the 1st house.

Inside the house:

$$p(23 \leq x \leq 41, 0 \leq y \leq 24) = 0$$
$$u(22 \leq x \leq 42, 0 \leq y \leq 25) = 0$$

$$v(22 \leq x \leq 42, 0 \leq y \leq 25) = 0$$

Only pressure will have Neuman boundary condition at the boundaries of house.

Left and right walls of the 1st house at $0 \leq y \leq 25$:

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial x}|_{(x=22)} = 0$$

$$p_{22,j} = p_{21,j}$$

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial x}|_{(x=42)} = 0$$

$$p_{42,j} = p_{43,j}$$

Our 1st house has roof. It wasn't hard to describe corresponding conditions for it.

At $22 \leq x \leq 42, 25 \leq y \leq 35$ and if at this interval $y - x \leq 3,\ x + y \leq 67$ holds:

$$u = 0$$

$$v = 0$$

$$\frac{\partial p}{\partial n} = 0$$

It means that right side:

$$p_{ij} = p_{i-1,j+1}$$

Left side:

$$p_{ij} = p_{i+1,j+1}$$

Boundary conditions at the 2nd house.

Inside the house:

$$p(71 \leq x \leq 84, 0 \leq y \leq 20) = 0$$
$$u(70 \leq x \leq 85, 0 \leq y \leq 20) = 0$$

$$v(70 \leq x \leq 85, 0 \leq y \leq 20) = 0$$

Only pressure will have Neuman boundary condition at the boundaries of house.

Left and right walls of the 2nd house at $0 \le y \le 20$:

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial x}|_{(x=70)} = 0$$

$$p_{70,j} = p_{69,j}$$

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial x}|_{(x=85)} = 0$$

$$p_{85,j} = p_{86,j}$$

The roof of 2nd house for pressure:

At $70 \le x \le 85$:

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial y}|_{(y=20)} = 0$$

$$p_{i,20} = p_{i,21}$$

3. Numerical algorithm:

Our algorithm will consist of 3 steps, and here is what we need to find:

1. u^*_{ij}, v^*_{ij} ;
2. p^{n+1}_{ij}
3. $u^{n+1}_{ij}, v^{n+1}_{ij}$;
4. Check for convergence, if it does not converge start from 1st step with new u and v

We will use the splitting in physical parameters method. First, we must approximate equation (1) as follows:

$$\frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta t} = \frac{u^{n+1}_{ij} - u^n_{ij} + u^*_{ij} - u^*_{ij}}{\Delta t}$$

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

$$\begin{cases} \dfrac{u^{n+1}_{ij} - u^*_{ij}}{\Delta t} = -\dfrac{1}{\rho} * \dfrac{\partial P}{\partial x} \\ \dfrac{u^*_{ij} - u^n_{ij}}{\Delta t} = -u\dfrac{\partial u}{\partial x} - v\dfrac{\partial u}{\partial y} + \dfrac{1}{Re} * \left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) \end{cases}$$

Now find the unknown u^*_{ij}:

$$u_{ij}^{*} = \Delta t * (-u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} + \frac{1}{Re} * \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)) + u_{ij}^{n}$$

$$u_{ij}^{*} = u_{ij}^{n} + \Delta t\left(-u_{ij}^{n} * \left(\frac{u_{i+1,j}^{n} - u_{i-1,j}^{n}}{2 * \Delta x}\right) - v_{ij}^{n} * \left(\frac{u_{i,j+1}^{n} - u_{i,j-1}^{n}}{2 * \Delta y}\right) + \frac{1}{Re}\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2}\right)\right) \quad (4)$$

(2) the equation is also approximate:

$$\frac{v_{ij}^{n+1} - v_{ij}^{n}}{\Delta t} = \frac{v_{ij}^{n+1} - v_{ij}^{n} + v_{ij}^{*} - v_{ij}^{*}}{\Delta t}$$

$$\begin{cases} \dfrac{v_{ij}^{n+1} - v_{ij}^{*}}{\Delta t} = -\dfrac{1}{\rho} * \dfrac{\partial P}{\partial x} \\ \dfrac{v_{ij}^{*} - v_{ij}^{n}}{\Delta t} = -u\dfrac{\partial v}{\partial x} - v\dfrac{\partial v}{\partial y} + \dfrac{1}{Re} * (\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2}) \end{cases}$$

And find the unknown v_{ij}^{*}:

$$v_{ij}^{*} = \Delta t * \left(-u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} + \frac{1}{Re} * \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right) + v_{ij}^{n}$$

$$v_{ij}^{*} = v_{ij}^{n} + \Delta t\left(-u_{ij}^{n} * \left(\frac{v_{i+1,j}^{n} - v_{i-1,j}^{n}}{2 * \Delta x}\right) - v_{ij}^{n} * \left(\frac{v_{i,j+1}^{n} - v_{i,j-1}^{n}}{2 * \Delta y}\right) + \frac{1}{Re}\left(\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{\Delta x^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}}{\Delta y^2}\right)\right) \quad (5)$$

(4) and (5) are the found 1-step.

Next step to find p_{ij}^{n+1}- pressure:

In order to find it, we must find the components u and v:

$$u_{ij}^{n+1} = -\frac{\Delta t}{\rho} * \frac{\partial P}{\partial x} + u_{ij}^{*} \quad (6)$$

$$v_{ij}^{n+1} = -\frac{\Delta t}{\rho} * \frac{\partial P}{\partial y} + v_{ij}^{*} \quad (7)$$

And insert it into equation (3):

$$\frac{\partial(-\frac{\Delta t}{\rho} * \frac{\partial P}{\partial x} + u_{ij}^*)}{\partial x} + \frac{\partial(-\frac{\Delta t}{\rho} * \frac{\partial P}{\partial x} + v_{ij}^*)}{\partial y} = 0$$

$$\frac{\partial u_{ij}^*}{\partial x} - \frac{\Delta t}{\rho} * \frac{\partial^2 P}{\partial x^2} + \frac{\partial u_{ij}^*}{\partial y} - \frac{\Delta t}{\rho} * \frac{\partial^2 P}{\partial y^2} = 0$$

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = \left(\frac{\partial u_{ij}^*}{\partial x} + \frac{\partial u_{ij}^*}{\partial y}\right) * \frac{\rho}{\Delta t}$$

$$P_{ij}^{n+1} = \frac{1}{4} * \left(P_{i+1,j}^n + P_{i-1,j}^n + P_{i,j+1}^n + P_{i,j-1}^n - \left(\frac{\rho * \left(u_{i+1,j}^* - u_{i-1,j}^*\right)}{2 * \Delta x * \Delta t} + \frac{\rho * \left(v_{i,j+1}^* - v_{i,j-1}^*\right)}{2 * \Delta y * \Delta t}\right) * \Delta x * \Delta y\right) (8)$$

We found the pressure.

And finally, the 3rd step is to find the velocity components u and v:

We can easily find it from this equation:

For u:

$$\frac{u_{ij}^{n+1} - u_{ij}^*}{\Delta t} = \frac{1}{\rho} * \frac{\partial P}{\partial x}$$

$$u_{ij}^{n+1} = \frac{\Delta t}{\rho} * \frac{\partial P}{\partial x} + u_{ij}^*$$

$$u_{ij}^{n+1} = u_{ij}^* - \Delta t * \left(\frac{P_{i+1,j}^n - P_{i-1,j}^n}{2 * \rho * \Delta x}\right) \ (9)$$

For v:

$$\frac{v_{ij}^{n+1} - v_{ij}^*}{\Delta t} = \frac{1}{\rho} * \frac{\partial P}{\partial x}$$

$$v_{ij}^{n+1} = \frac{\Delta t}{\rho} * \frac{\partial P}{\partial x} + v_{ij}^*$$

$$v_{ij}^{n+1} = v_{ij}^* - \Delta t * \left(\frac{P_{i,j+1}^n - P_{i,j-1}^n}{2 * \rho * \Delta y}\right) \ (10)$$

4. Program code

```
#include "pch.h"
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <ctime>

using namespace std;
```

```
int main() {
	int const xSize = 101; //x
	int const ySize = 51; //y
	int fullIterations = 0;
	int pressureIterations = 0;
	int endTime, startTime;
	double U[xSize][ySize],
		U_star[xSize][ySize],
		oldU[xSize][ySize],
		V[xSize][ySize],
		V_star[xSize][ySize],
		oldV[xSize][ySize],
		P[xSize][ySize],
		oldP[xSize][ySize];

	double dt, dx, dy, Re, errorP, errorSpeed, differenceU, differenceV, differenceP, rho;

	startTime = clock();
	Re = 10.0;
	dx = 1.0 * pow(xSize - 1, -1);
	dy = 1.0 * pow(xSize - 1, -1);
	dt = (Re*dx*dy)*0.25;
	errorP = pow(10, -5);
	errorSpeed = pow(10, -12);
	rho = 1.2;
	ofstream fout("Navie4.dat", ios::out);
	fout << "variables=\"X\" ,\"Y\" ,\"U\" ,\"V\" ,\"P\"" << endl;

	//fill by zeros
	for (int i = 0; i < xSize; i++) {
		for (int j = 0; j < ySize; j++) {
			U[i][j] = 0.0;
			V[i][j] = 0.0;
			oldU[i][j] = 0.0;
			oldV[i][j] = 0.0;
			U_star[i][j] = 0.0;
			V_star[i][j] = 0.0;
			oldP[i][j] = 0.0;
			P[i][j] = 0.0;
		}
	}

	for (int j = ySize / 2; j < ySize; j++) {
		oldU[0][j] = 0.05; //inlet
		oldU[xSize - 1][j] = oldU[xSize - 2][j]; //outlet
		oldV[xSize - 1][j] = oldV[xSize - 2][j];
	}

	//house 1 (without roof)
	for (int i = 22; i <= 42; i++) {
		for (int j = 0; j <= 25; j++) {
			oldU[i][j] = 0.0;
```

```
            oldV[i][j] = 0.0;
            oldP[i][j] = 0.0;
            P[i][j] = 0.0;
        }
}
// roof of house 1
for (int i = 22; i <= 42; i++) {
        for (int j = 25; j <= 35; j++) {
            if (j - i <= 3 && i + j <= 67) {
                oldU[i][j] = 0.0;
                oldV[i][j] = 0.0;
                oldP[i][j] = 0.0;
                P[i][j] = 0.0;
            }
        }
}

// house 2
for (int i = 70; i <= 85; i++) {
        for (int j = 0; j <= 20; j++) {
            oldU[i][j] = 0.0;
            oldV[i][j] = 0.0;
            oldP[i][j] = 0.0;
            P[i][j] = 0.0;
        }
}

do {
        for (int i = 0; i <= xSize - 1; i++) {
            // boundary conditions of Dirihlet, on walls
            for (int j = 0; j <= ySize - 1; j++) {
                U_star[xSize - 1][j] = 0.0;
                U_star[i][0] = 0.0;
                U_star[0][j] = 0.0;
                U_star[i][ySize - 1] = 0.0;

                V_star[i][ySize - 1] = 0.0;
                V_star[xSize - 1][j] = 0.0;
                V_star[i][0] = 0.0;
                V_star[0][j] = 0.0;

            }
        }
        //house 1 without roof
        for (int i = 22; i <= 42; i++) {
            for (int j = 0; j <= 25; j++) {
                U_star[i][j] = 0.0;
                V_star[i][j] = 0.0;
            }
        }
        //roof of house 1
        for (int i = 22; i <= 42; i++) {
            for (int j = 25; j <= 35; j++) {
```

```
				if (j - i <= 3 && i + j <= 67) {
					U_star[i][j] = 0.0;
					V_star[i][j] = 0.0;
				}
			}
		}

		//house 2
		for (int i = 70; i <= 85; i++) {
			for (int j = 0; j <= 20; j++) {
				U_star[i][j] = 0.0;
				V_star[i][j] = 0.0;
			}
		}

		for (int j = ySize / 2; j < ySize; j++) {
			U_star[0][j] = 0.05; //inlet
		}
		for (int j = ySize / 2; j < ySize; j++) {
			U_star[xSize - 1][j] = U_star[xSize - 2][j]; //outlet
			V_star[xSize - 1][j] = V_star[xSize - 2][j];
		}
		//Method of splitting by physical parameters
		for (int i = 1; i < xSize - 1; i++)
		{
			for (int j = 1; j < ySize - 1; j++) {
				if (i >= 22 && i <= 42 && j >= 0 && j <= 25) {
					U_star[i][j] = 0.0;
					V_star[i][j] = 0.0;
				}
				else if (i >= 22 && i <= 42 && j >= 25 && j <= 35 && j - i <= 3
&& i + j <= 67) {
					U_star[i][j] = 0.0;
					V_star[i][j] = 0.0;
				}
				else if (i >= 70 && i <= 85 && j >= 0 && j <= 20) {
					U_star[i][j] = 0.0;
					V_star[i][j] = 0.0;
				}
				else {
					U_star[i][j] = oldU[i][j] +
						dt * (
						(-oldU[i][j] *
							(oldU[i + 1][j] - oldU[i - 1][j]) / (2 * dx) -
							oldV[i][j] *
							(oldU[i][j + 1] - oldU[i][j - 1]) / (2 * dy))
							+ (
							(oldU[i + 1][j] - 2.0 * oldU[i][j] + oldU[i
- 1][j]) / (dx * dx)
								+ (oldU[i][j + 1] - 2.0 *
oldU[i][j] + oldU[i][j - 1]) / (dy* dy)
								) / Re
							);
```

```
                                        V_star[i][j] = oldV[i][j] + dt *
                                                (
                                                (-oldU[i][j] * (oldV[i + 1][j] - oldV[i - 1][j]) / (2
* dx)
                                                        - oldV[i][j] * (oldV[i][j + 1] - oldV[i][j -
1]) / (2 * dy))
                                                        + (
                                                        (oldV[i + 1][j] - 2.0 * oldV[i][j] + oldV[i
- 1][j]) / (dx* dx)
                                                                + (oldV[i][j + 1] - 2.0 *
oldV[i][j] + oldV[i][j - 1]) / (dy* dy)
                                                                ) / Re);
                                }
                        }
                }


                pressureIterations = 0;
                double prev_differenceP = 0;
                do {

                        for (int j = (ySize - 1) / 2; j < ySize - 1; j++) {
                                P[xSize - 1][j] = 0; //Outlet
                        }

                        //The left and right walls of 1st house
                        for (int j = 0; j <= 25; j++) {
                                P[22][j] = P[21][j];
                                P[42][j] = P[43][j];
                        }

                        //The left and right walls of 2nd house
                        for (int j = 0; j <= 20; j++) {
                                P[70][j] = P[69][j];
                                P[85][j] = P[86][j];
                        }

                        //Roof of 2nd house
                        for (int i = 70; i <= 85; i++) {
                                P[i][20] = P[i][21];
                        }
                        //Roof of 1st house
                        for (int i = 22; i <= 42; i++) {
                                for (int j = 25; j <= 35; j++) {
                                        if (j - i == 3) {
                                                P[i][j] = P[i - 1][j + 1];
                                        }
                                        else if (i + j == 67) {
                                                P[i][j] = P[i + 1][j + 1];
                                        }
                                }
                        }
```

```
			//Neumann conditions
			for (int i = 0; i <= xSize - 1; i++) {
				for (int j = 0; j <= (ySize - 1) / 2; j++) {
					P[0][j] = P[1][j];
					P[i][ySize - 1] = P[i][ySize - 2];
					P[i][0] = P[i][1];
					P[xSize - 1][j] = P[xSize - 2][j];
				}
			}


			//Poisson equation
			for (int i = 1; i < xSize - 1; i++) {
				for (int j = 1; j < ySize - 1; j++) {
					if (i >= 22 && i <= 42 && j >= 0 && j <= 25 && j - i
<= 3 && i + j <= 67) {
						P[i][j] = 0.0;
					}
					else if (i >= 22 && i <= 42 && j >= 25 && j <= 35 && j
- i <= 3 && i + j <= 67) {
						P[i][j] = 0.0;
					}
					else if (i >= 70 && i <= 85 && j >= 0 && j <= 20) {
						P[i][j] = 0.0;
					}
					else {
						P[i][j] = 0.25 * (oldP[i + 1][j] + oldP[i - 1][j] +
oldP[i][j + 1] + oldP[i][j - 1]
							- (rho*(U_star[i + 1][j] - U_star[i - 1][j])
/ (2 * dx*dt)
								+ rho * (V_star[i][j + 1] -
V_star[i][j - 1]) / (2 * dy*dt))*dy*dx);
					}
				}
			}

			//left and right walls of 1st house
			for (int j = 0; j <= 25; j++) {
				P[22][j] = P[21][j];
				P[42][j] = P[43][j];
			}

			//left and right walls of 2nd house
			for (int j = 0; j <= 20; j++) {
				P[70][j] = P[69][j];
				P[85][j] = P[86][j];
			}

			//roof of 2nd house
			for (int i = 70; i <= 85; i++) {
				P[i][20] = P[i][21];
			}
```

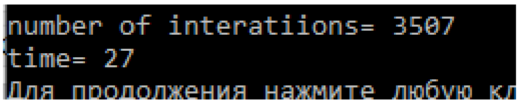
```
				//roof of 1st house
				for (int i = 22; i <= 42; i++) {
					for (int j = 25; j <= 35; j++) {
						if (j - i == 3) {
							P[i][j] = P[i - 1][j + 1];
						}
						else if (i + j == 67) {
							P[i][j] = P[i + 1][j + 1];
						}
					}
				}

				differenceP = 0.0;
				for (int i = 0; i < xSize; i++) {
					for (int j = 0; j < ySize; j++) {
						if (differenceP < fabs(P[i][j] - oldP[i][j])) {
							differenceP = fabs(P[i][j] - oldP[i][j]);
						}
					}
				}
				for (int i = 0; i < xSize; i++) {
					for (int j = 0; j < ySize; j++) {
						oldP[i][j] = P[i][j];
					}
				}
				pressureIterations++;
			} while (differenceP > errorP);

			//2nd part
			for (int i = 1; i < xSize - 1; i++)
			{
				for (int j = 1; j < ySize - 1; j++)
				{
					if (i >= 22 && i <= 42 && j >= 0 && j <= 25) {
						U[i][j] = 0.0;
						V[i][j] = 0.0;
					}
					else if (i >= 22 && i <= 42 && j >= 25 && j <= 35 && j - i <= 3
&& i + j <= 67) {
						U[i][j] = 0.0;
						V[i][j] = 0.0;
					}
					else if (i >= 70 && i <= 85 && j >= 0 && j <= 20) {
						U[i][j] = 0.0;
						V[i][j] = 0.0;
					}
					else {
						U[i][j] = U_star[i][j] - dt * (oldP[i + 1][j] - oldP[i - 1][j]) /
(2 * rho*dx);
						V[i][j] = V_star[i][j] - dt * (oldP[i][j + 1] - oldP[i][j - 1]) /
(2 * rho*dy);
					}
				}
```

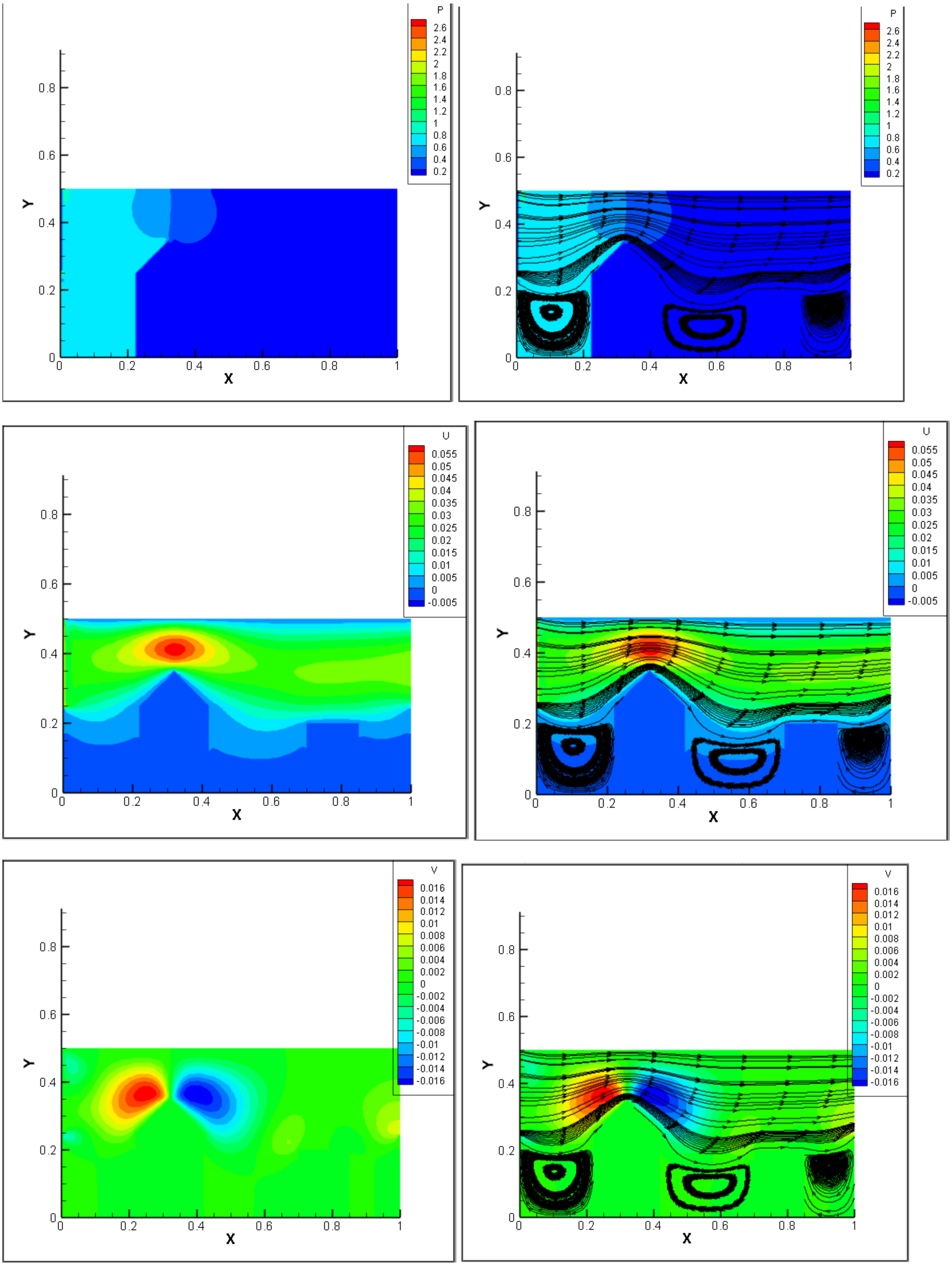
```
}
//Dirichlet
for (int i = 0; i < xSize; i++) {

	for (int j = 0; j < ySize; j++) {
		U[i][0] = 0;
		V[i][0] = 0;

		U[i][ySize - 1] = 0;
		V[i][ySize - 1] = 0;

		U[0][j] = 0;
		V[0][j] = 0;

		U[xSize - 1][j] = 0;
		V[xSize - 1][j] = 0;
	}
}
//House 1(without roof)
for (int i = 22; i <= 42; i++) {
	for (int j = 0; j <= 25; j++) {
		U[i][j] = 0.0;
		V[i][j] = 0.0;
	}
}
//roof of 1st house
for (int i = 22; i <= 42; i++) {
	for (int j = 25; j <= 35; j++) {
		if (j - i <= 3 && i + j <= 67) {
			U[i][j] = 0.0;
			V[i][j] = 0.0;
		}
	}
}

//house 2
for (int i = 70; i <= 85; i++) {
	for (int j = 0; j <= 20; j++) {
		U[i][j] = 0.0;
		V[i][j] = 0.0;
	}
}

for (int j = ySize / 2; j < ySize; j++) {
	U[0][j] = 0.05; //inlet
	U[xSize - 1][j] = oldU[xSize - 2][j]; //outlet
	V[xSize - 1][j] = V[xSize - 2][j];
}

differenceU = 0.0;
differenceV = 0.0;
for (int i = 0; i < xSize; i++) {
	for (int j = 0; j < ySize; j++) {
```

```
                    if (differenceU < fabs(U[i][j] - oldU[i][j]))
                    {
                        differenceU = fabs(U[i][j] - oldU[i][j]);
                    }
                    if (differenceV < fabs(V[i][j] - oldV[i][j]))
                    {
                        differenceV = fabs(V[i][j] - oldV[i][j]);
                    }
                }
            }
            for (int i = 0; i < xSize; i++) {
                for (int j = 0; j < ySize; j++) {
                    oldU[i][j] = U[i][j];
                    oldV[i][j] = V[i][j];
                }
            }
            if (fullIterations % 10 == 0) {
                fout << "zone T = \"" << fullIterations % 10 << "\", i=" << xSize << ",
j=" << ySize << ", F=point" << endl;
                for (int j = 0; j < ySize; j++) {
                    for (int i = 0; i < xSize; i++) {
                        fout << i * dx << "\t" << j * dy << "\t" << U[i][j] << "\t"
<< V[i][j] << "\t" << P[i][j] << "\t" << endl;
                    }
                }
            }
            fullIterations++;
            cout << fullIterations << endl;
        } while (differenceU > errorSpeed || differenceV > errorSpeed);

        fout << "zone T = \"" << fullIterations % 10 + 1 << "\", i=" << xSize << ", j=" << ySize
<< ", F=point" << endl;
        for (int j = 0; j < ySize; j++) {
            for (int i = 0; i < xSize; i++) {
                fout << i * dx << "\t" << j * dy << "\t" << U[i][j] << "\t" << V[i][j] <<
"\t" << P[i][j] << "\t" << endl;
            }
        }

        fout.close();
        endTime = clock();
        cout << "number of interations= " << fullIterations << endl;


        system("pause"); return 0; }
```

5. Illustration and descriptions of numerical results

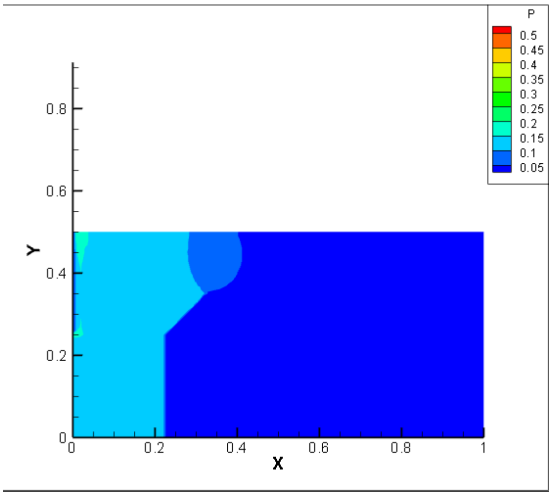
U = 0.05, Iter = 1000, Re = 10

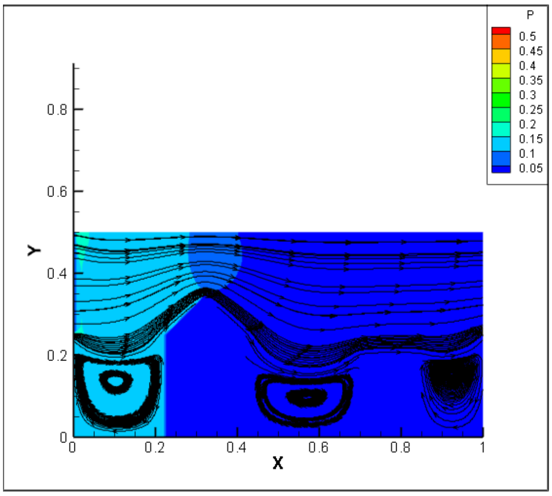


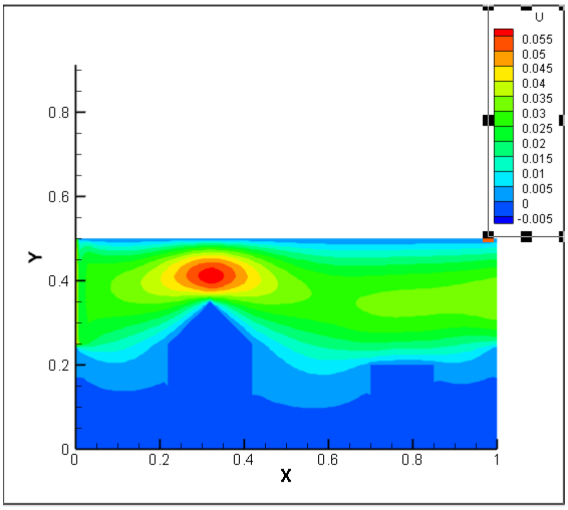


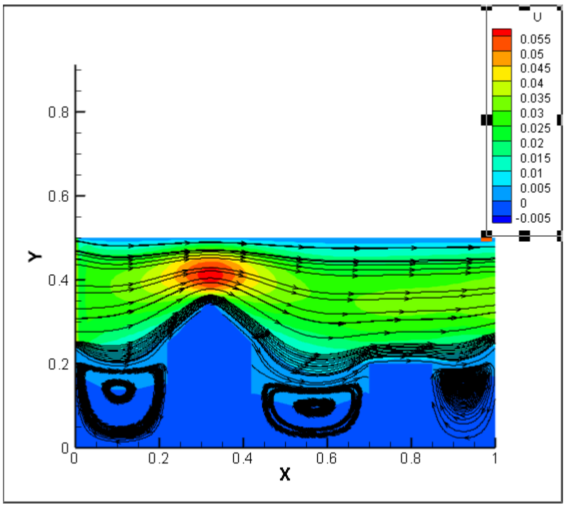
U = 0.05, Iter = 1000, Re = 50

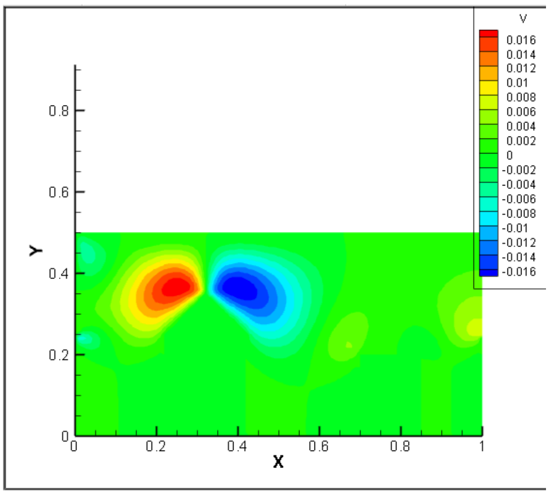
```
number of interatiions= 3768
time= 16
Для продолжения нажмите любую
```

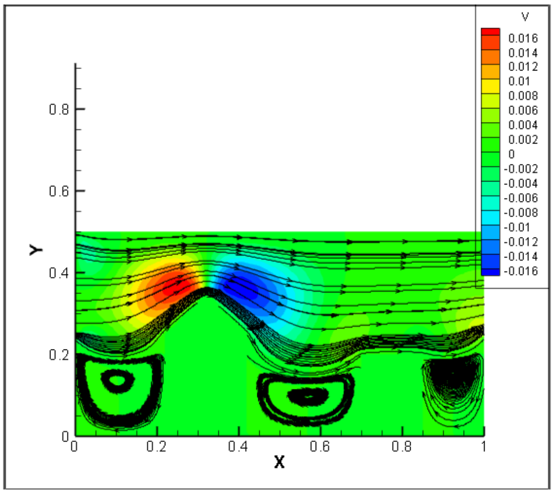






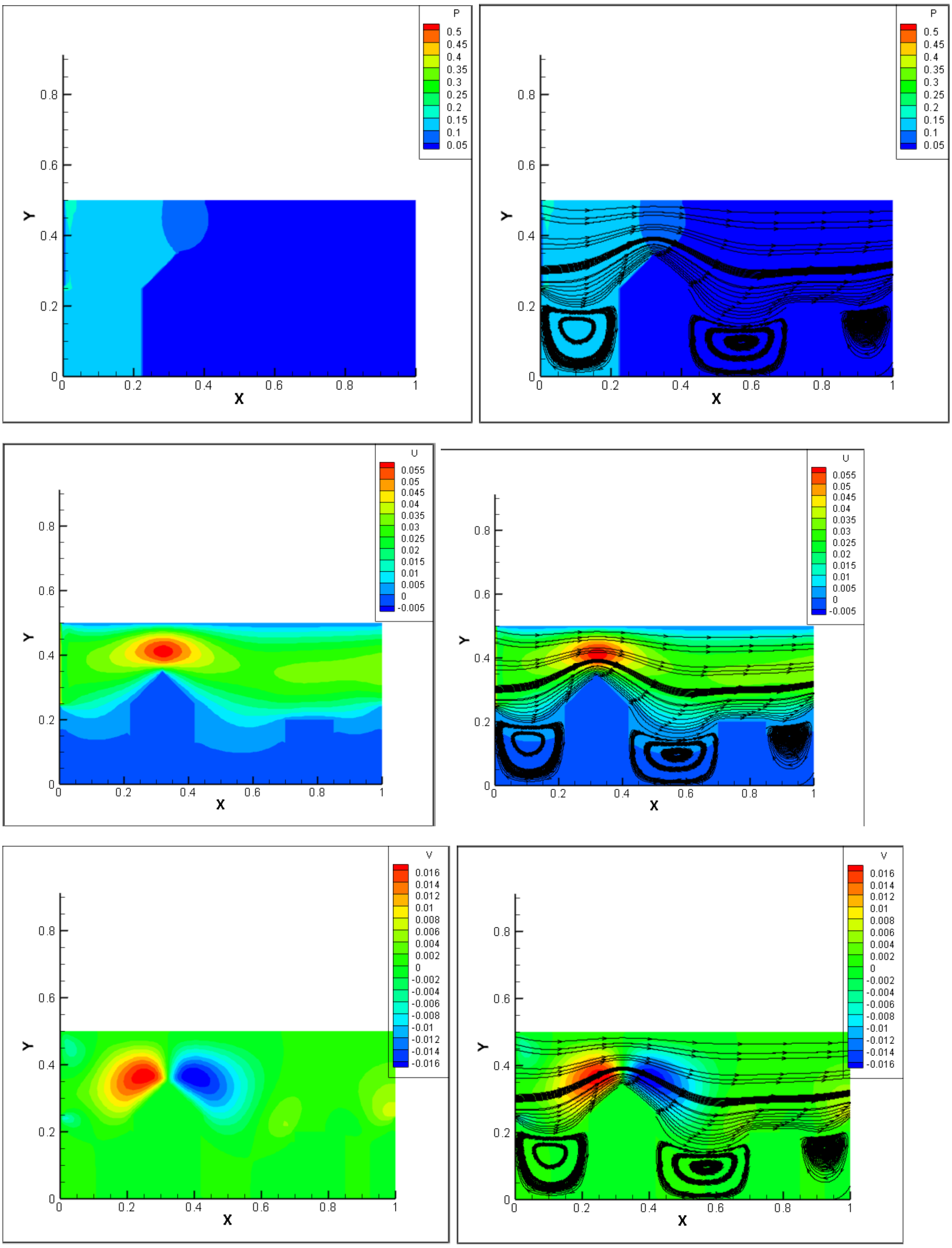






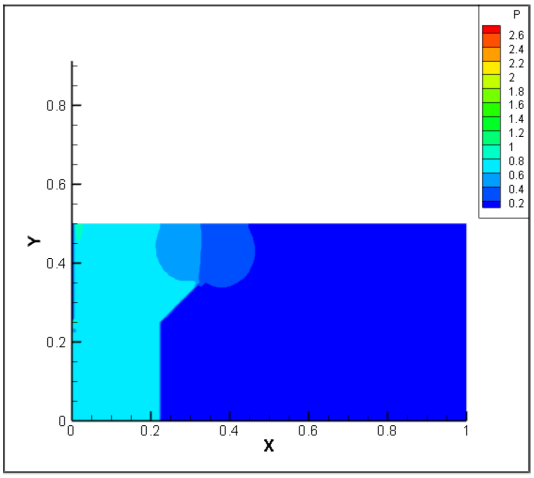
U = 0.05, Iter = 2000, Re = 50

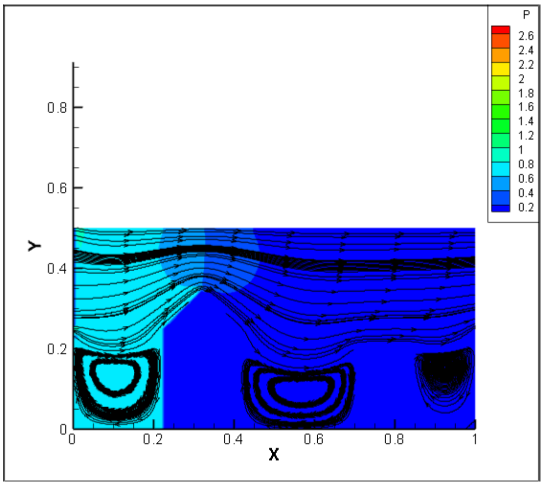
```
number of interatiions= 3768
time= 12
Для продолжения нажмите любую
```

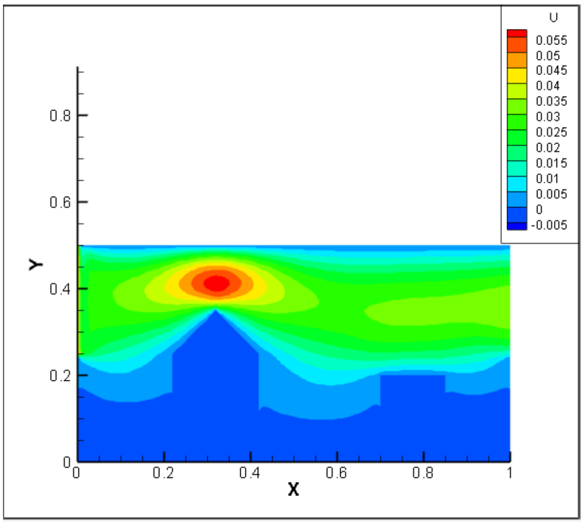


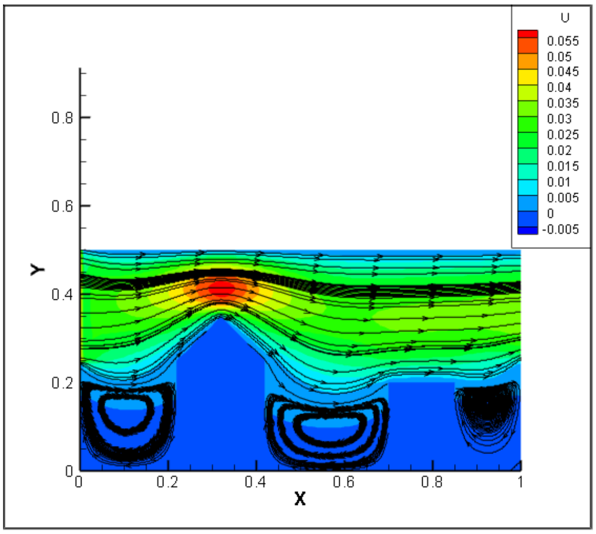
U = 0.05, Iter = 2000, Re = 10

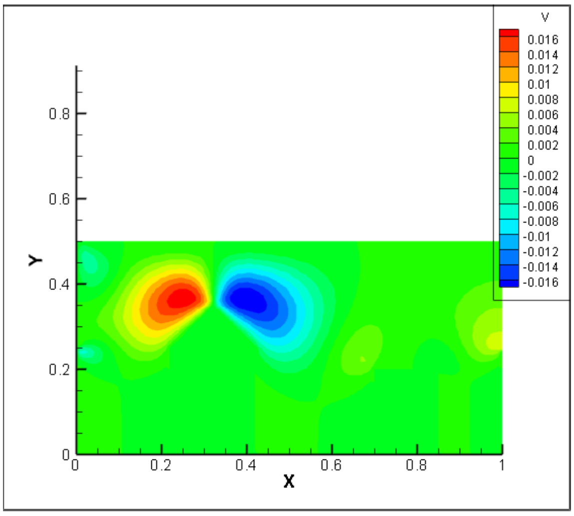
```
number of interatiions= 3507
time= 32
Для продолжения нажмите любую кл
```

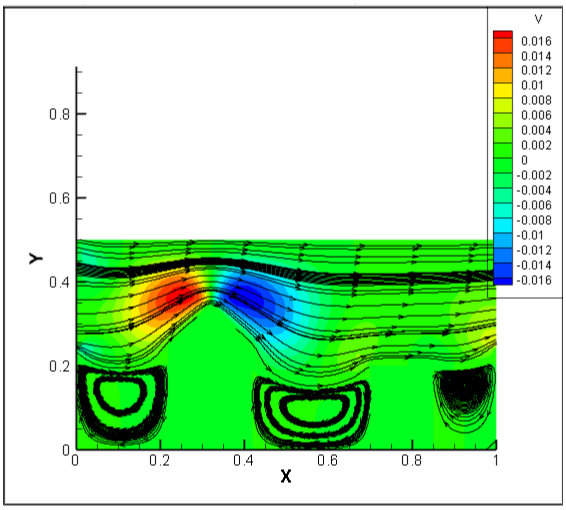






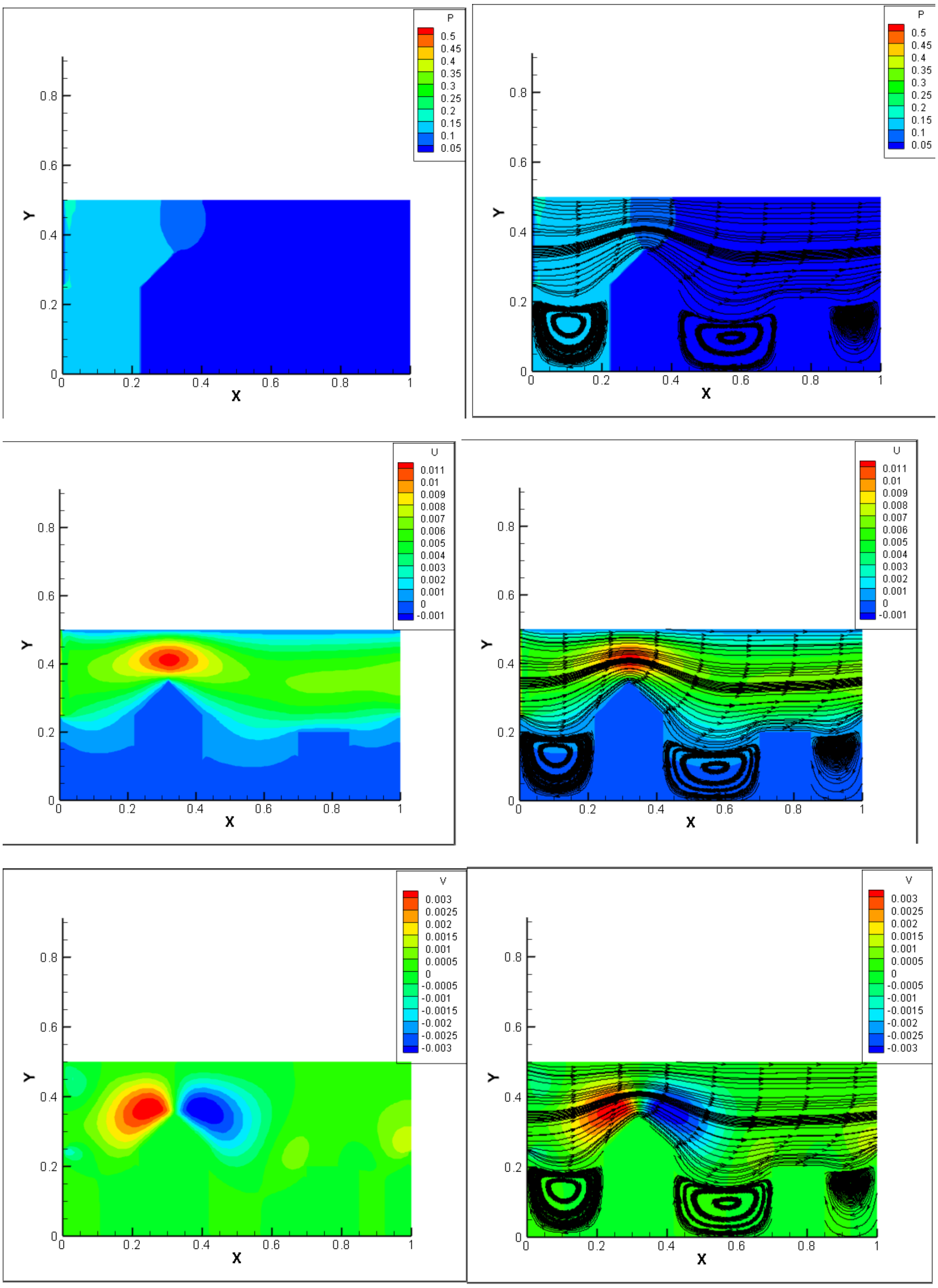






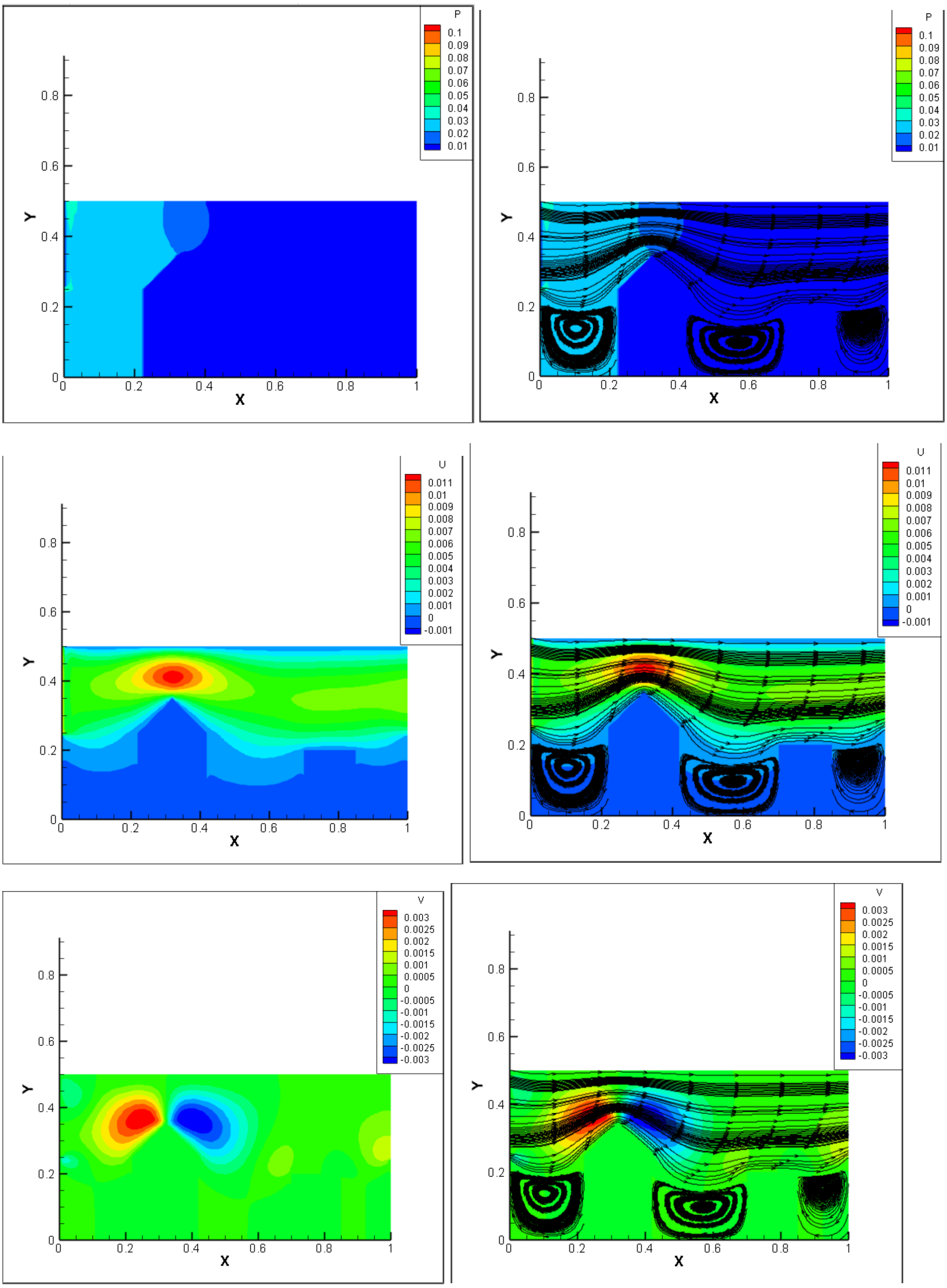
U = 0.01, Iter = 1000, Re = 10

```
number of interatiions= 3303
time= 12
Для продолжения нажмите любую клав
```



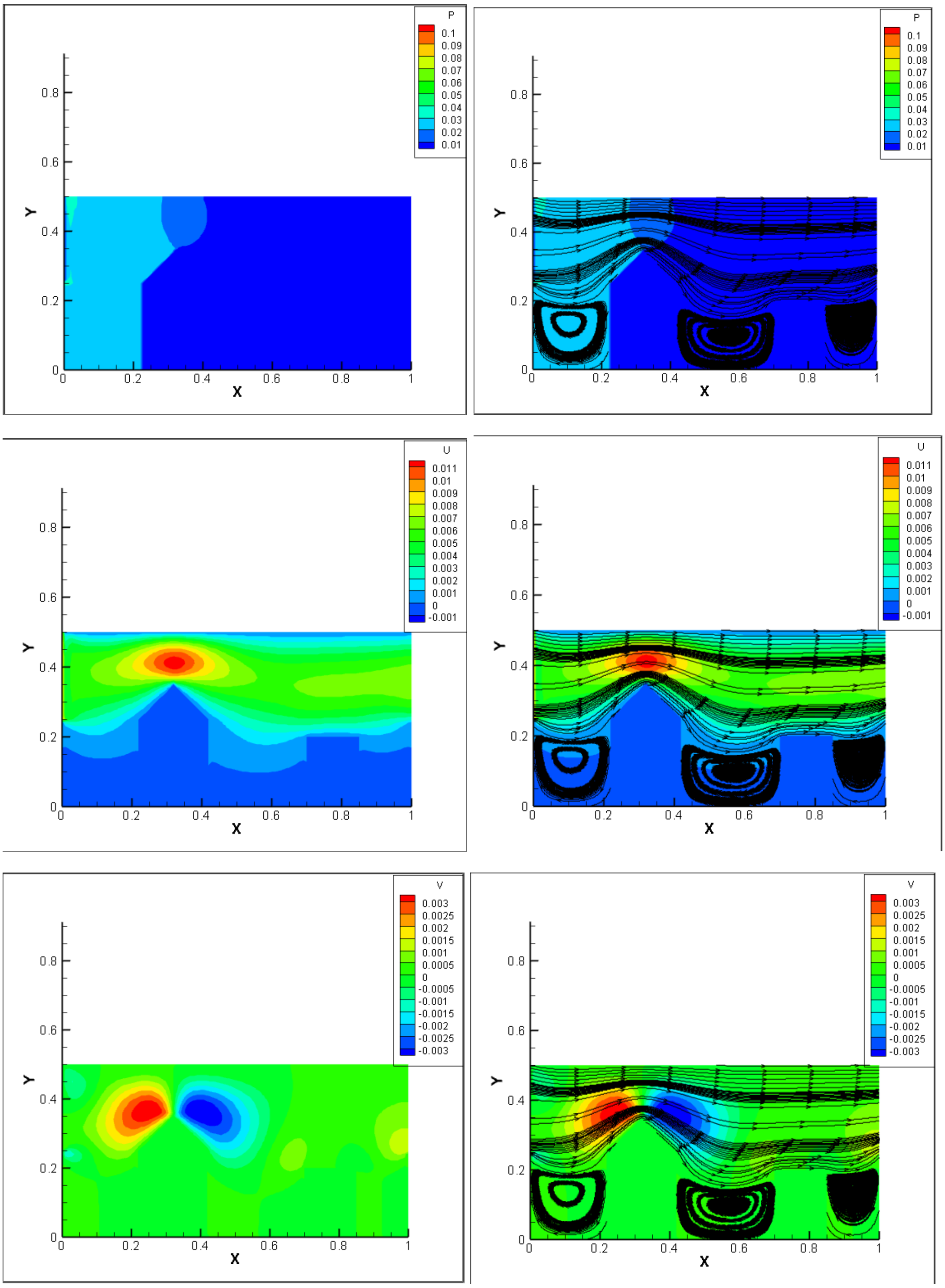
U = 0.01, Iter = 1000, Re = 50

```
number of interatiions= 3421
time= 8
```



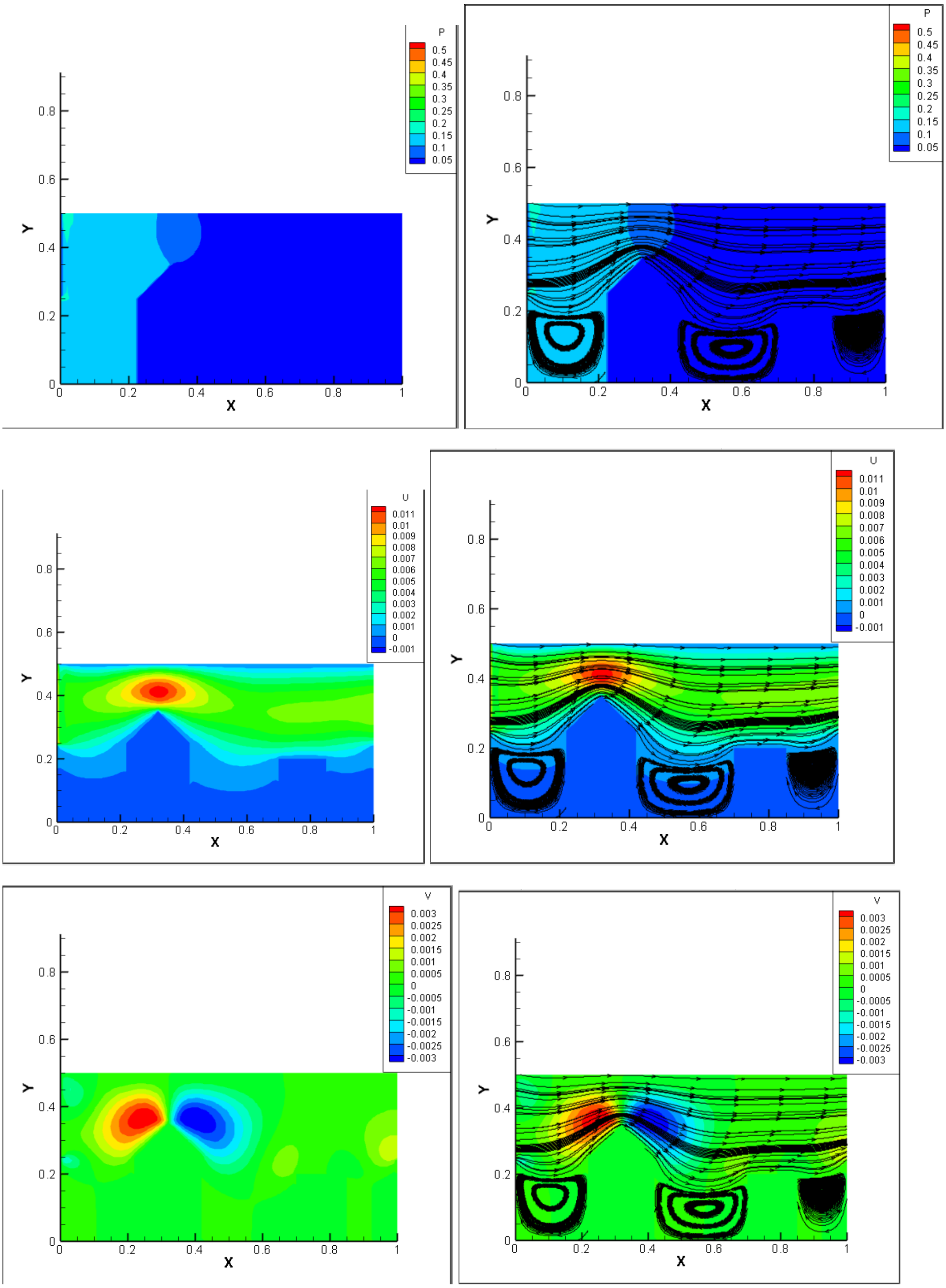
U = 0.01, Iter = 2000, Re = 50

```
number of interatiions= 3421
time= 7
Для продолжения нажмите любую
```

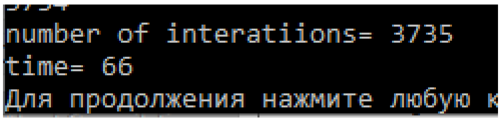


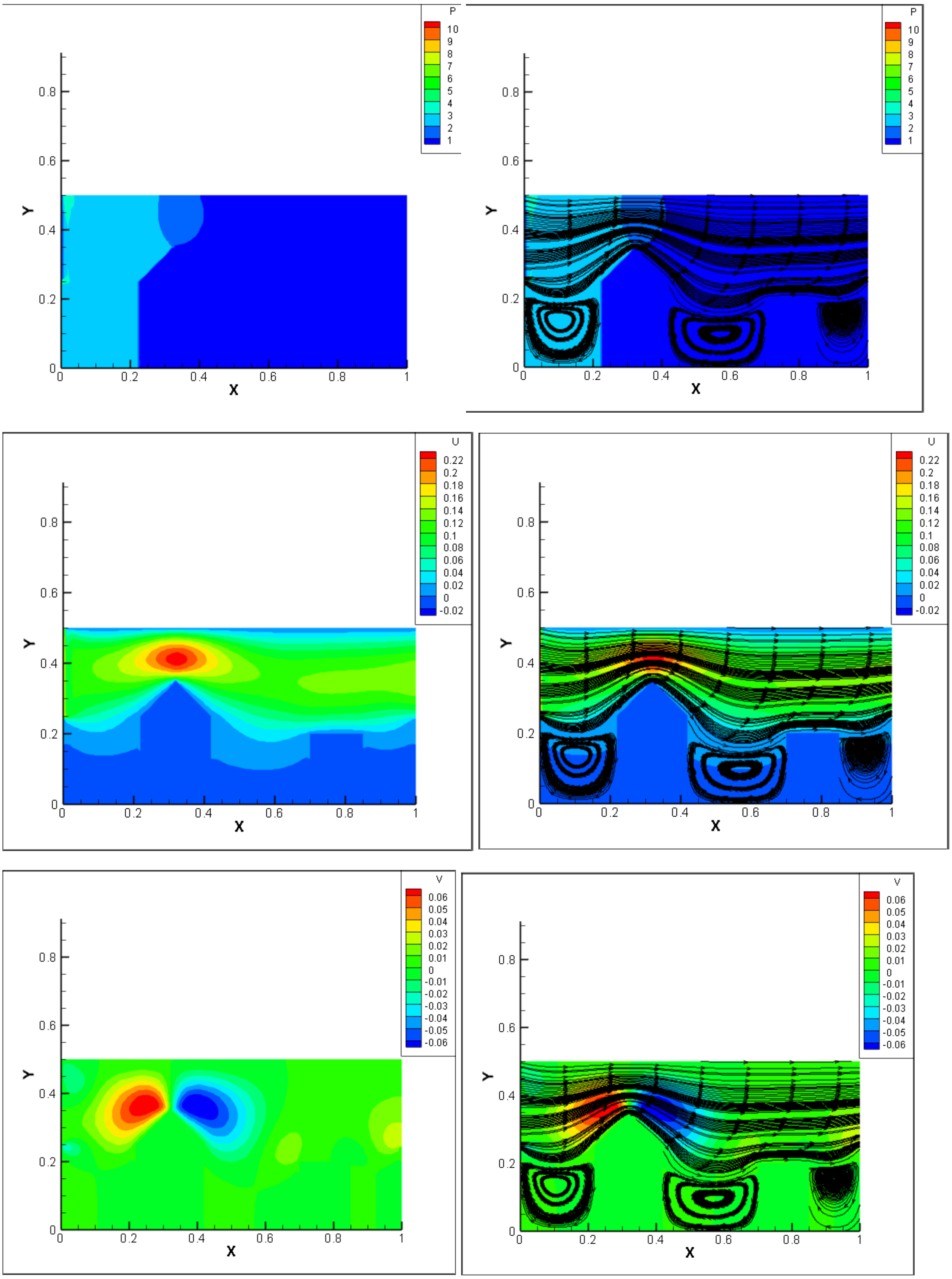
U = 0.01, Iter = 2000, Re = 10

```
number of interatiions= 3303
time= 14
Для продолжения нажмите любую
```



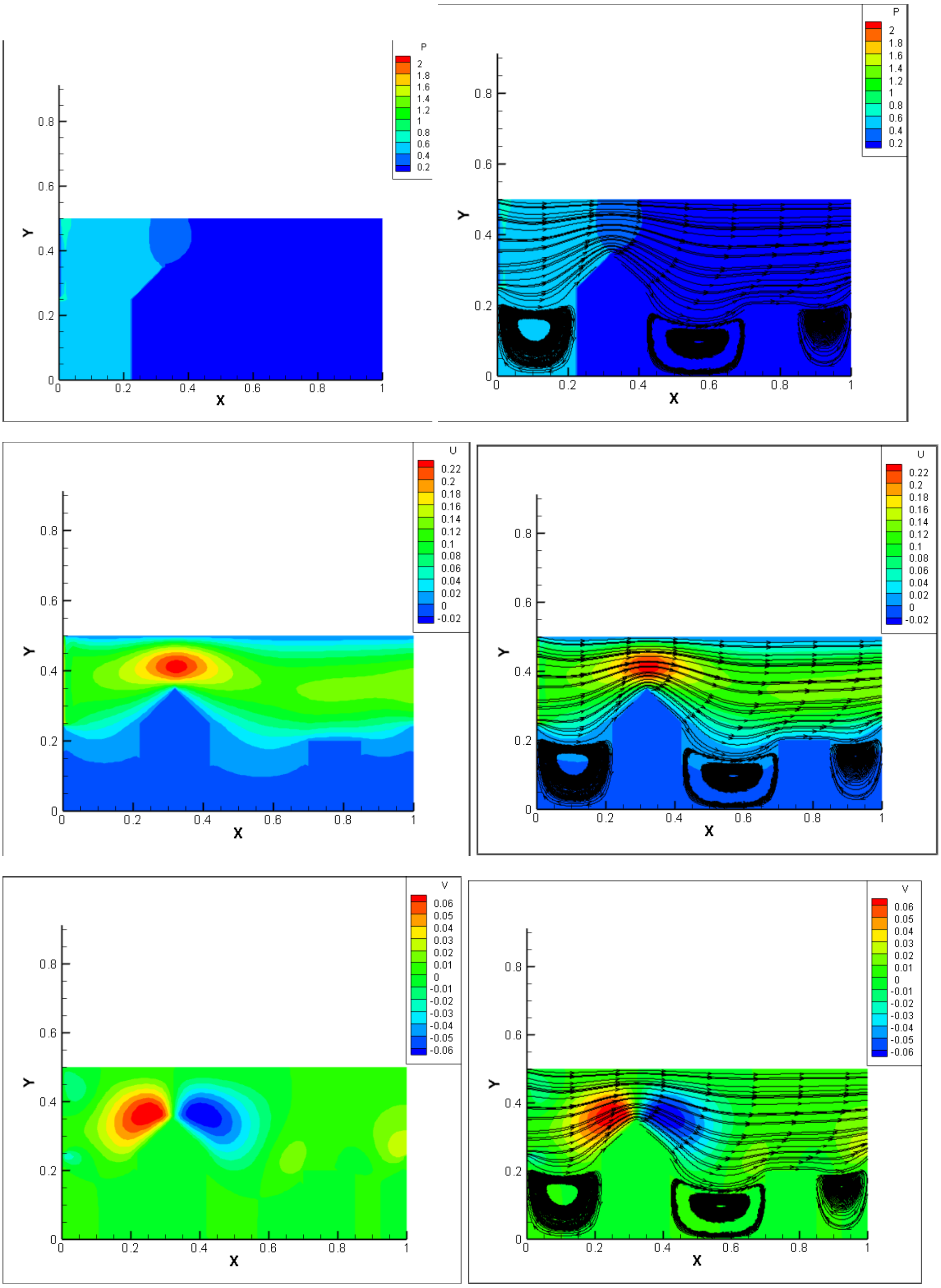
U = 0.2, Iter = 1000, Re = 10


number of interatiions= 3735
time= 66
Для продолжения нажмите любую к



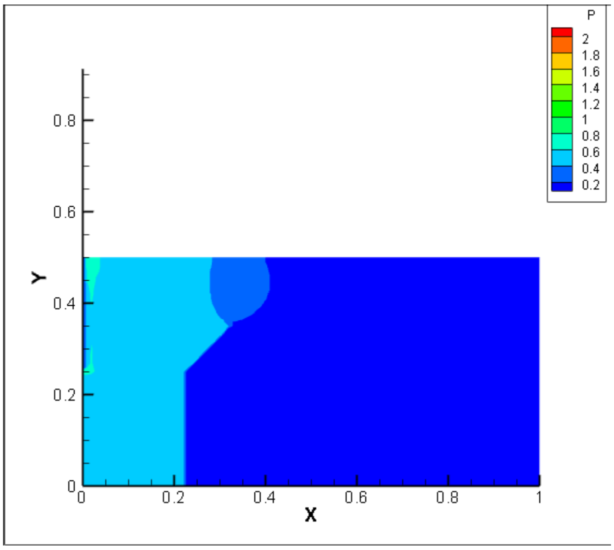
U = 0.2, Iter = 1000, Re = 50

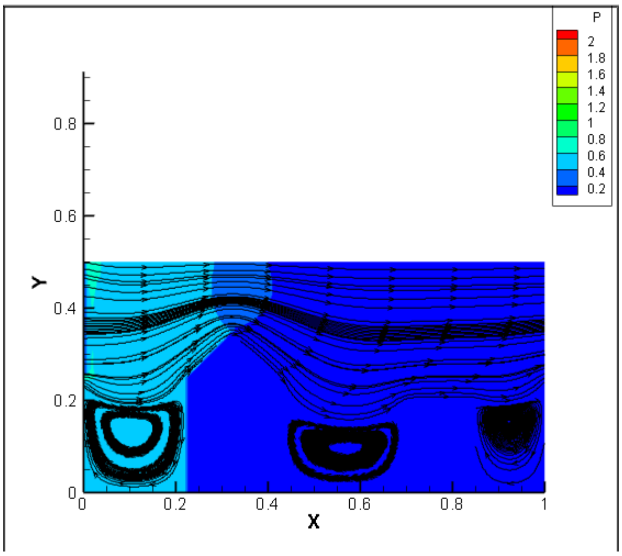
```
number of interatiions= 3727
time= 29
Для продолжения нажмите любую клав
```

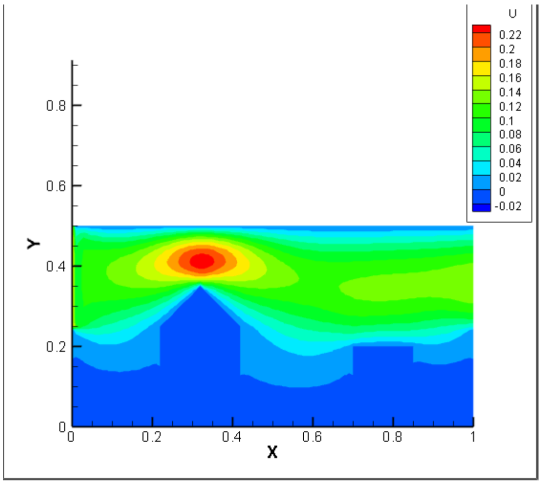


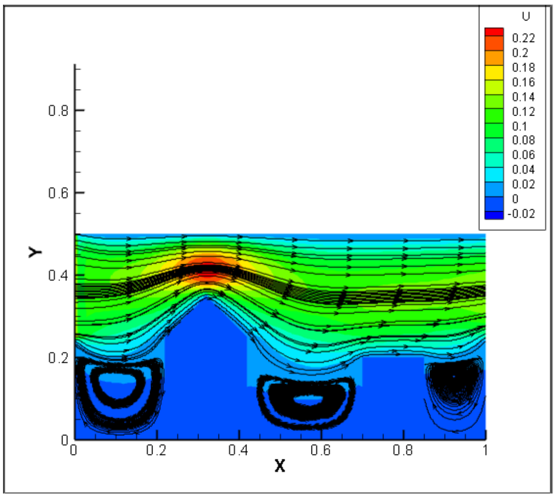
U = 0.2, Iter = 2000, Re = 50

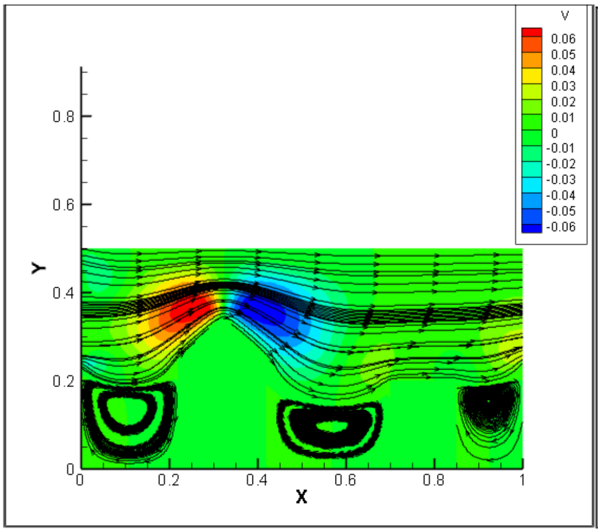
```
number of interatiions= 3727
time= 32
Для продолжения нажмите любук
```

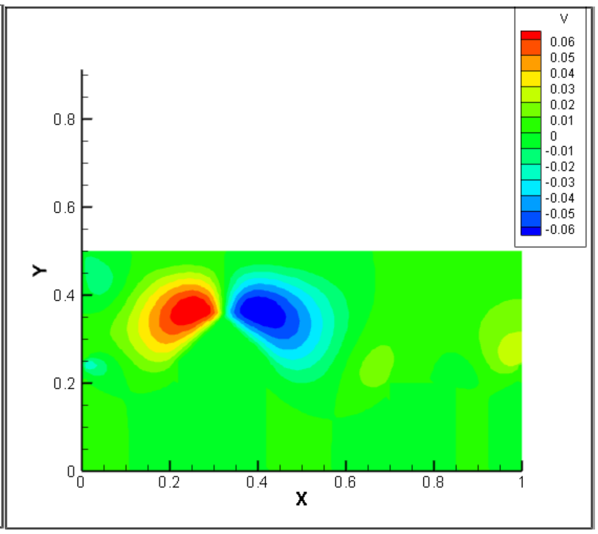






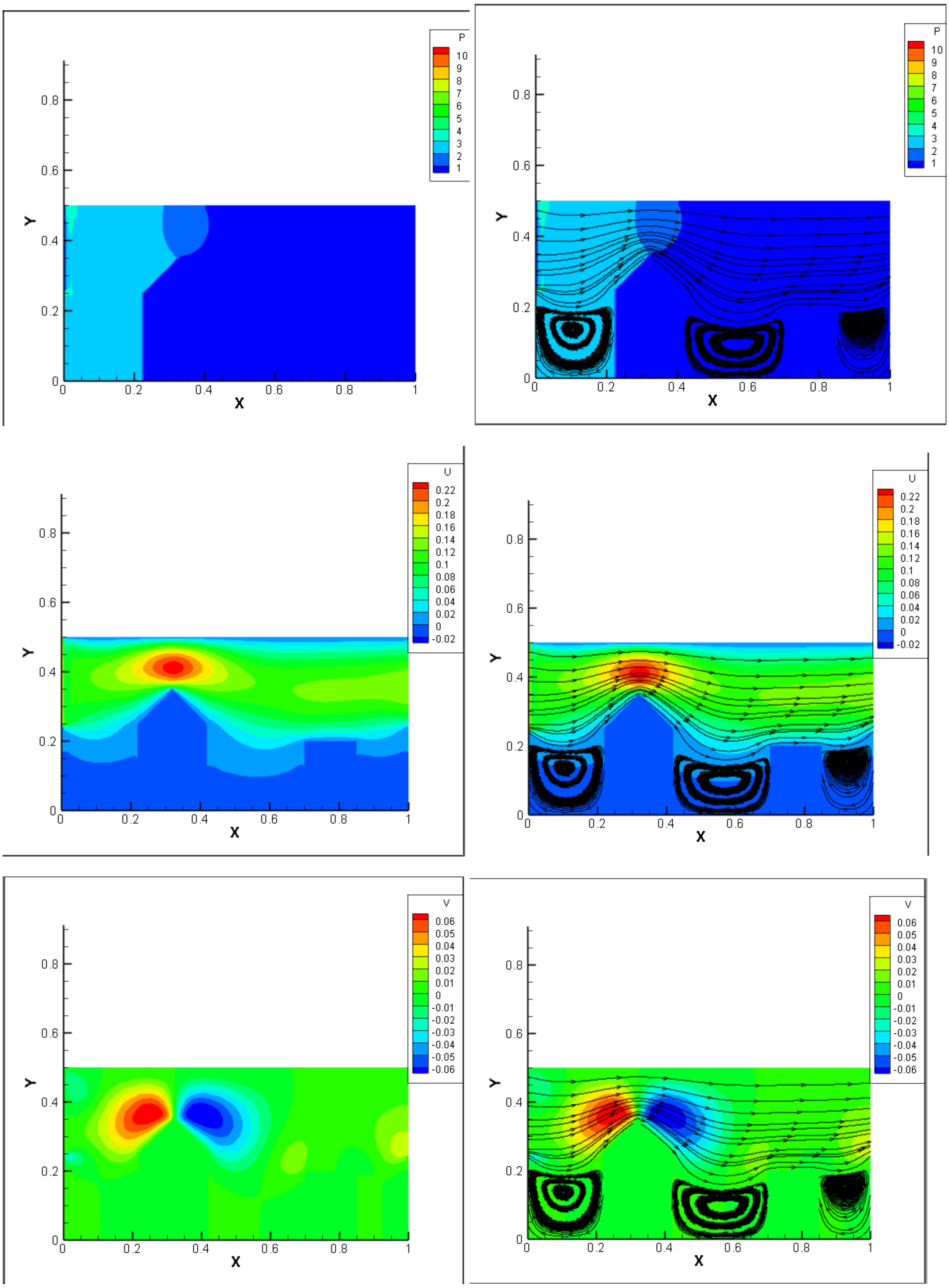






U = 0.2, Iter = 2000, Re = 50

```
number of interatiions= 3735
time= 78
Для продолжения нажмите любую
```



6. Analysis of results

We looked at different speeds and the Reynolds number also showed dynamics at different times. As you can see, the larger the Reynolds number, the more vortices there are, you can also see that between the houses and on the sides of the houses there are vortices, which prove that the yards are not blown, which is very bad for the environment and people. As already noted, vortices happen, or simply called turbulence, can happen in places where the laminar flow cannot flow through the parting areas. Turbulence can be on the roof and on the corners of the obstacles. We can notice it when the airflow flows around two houses, however if the building is higher then further airflow is required to flow it, which means that flow rate is becoming higher around of the buildings. When the air flow streams around a croud of buildings, the process is becoming more complecated, but the main patterns must be conserved. In order to make the yards draughtier, it is necessary to know that the distance between the houses and the height of the houses, but in our case, there is no yard draughtiness until the close distance of the two houses. As you can see, the highest speed U can be seen above the roof of the first house, since it is located at a high enough point. Also, the speed in V shows that the highest speed is in the left part of the roof of the first house, since this part meets the direct flow of the wind, and the other part of the roof has the lowest speed, since the wind flows around it. If you look at the pressure, you can see that at the inlet, the pressure has the maximum value and the further in the direction of the becomes less and the output has a zero value.

7. **Conclusions**

Summing up, we solved the problem of mathematical and computer simulation of air flow around buildings. To build a mathematical model, two-dimensional Navier-Stokes equations were used, where there are no such parameters as diffusion coefficients because the height (of buildings) in this case is not as large as in the previous problems that were considered earlier in this course (i.e., the environment is relatively homogeneous). This model is used, since these equations completely describe the flow movement, and in this task, the space is two-dimensional, the problem is more local and accurate for calculation. Locality is proved by the fact that we have considered the case of two houses located not far from each other. To solve this problem, the method of splitting by physical parameters was used (the full numerical algorithm is described in paragraph 3). The grid dimension is 101*51 for more accurate results and detailed research. As stated in the analysis of the data obtained (paragraph 6), it turned out that in the case under consideration, the buildings are too close to each other and the overlap between them is low. This can lead to environmental damage and air pollution. And the presence of a road between them can significantly worsen the whole situation. Therefore, appropriate conclusions were made that the first building should be built either lower or at a sufficiently large distance for good ventilation between the houses.