Institute of Fluid Mechanics and Heat Transfer
Graz University of Technology

Numerical Methods in Fluid Mechanics and Heat Transfer
Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Helfried Steiner

Lid Driven Cavity

Dalibor JAJCEVIC

Mat.Nr. : 0030015

Graz, October 2010

Contents

1 Introduction

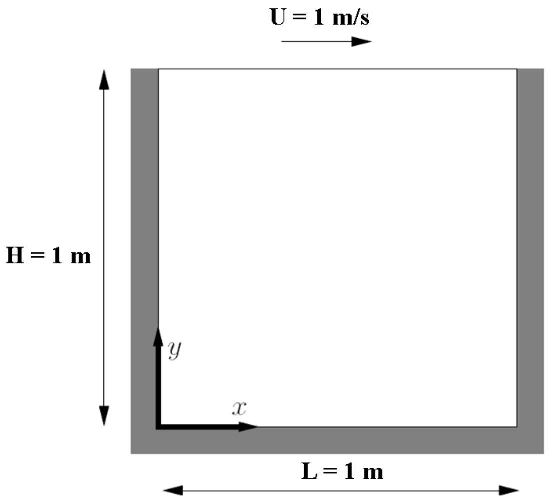


Figure 1: Geometry of the lid driven cavity

In this work a lid driven cavity flow is investigated applying numerical methods. The flow is considered as laminar, isothermal, and incompressible in a two-dimensional domain. The geometry is shown in Figure 1 in which three of the four boundaries are treated as walls. The top wall moves in the x-direction at a speed of 1 m/s while the other three are stationary, namely the right, the bottom, and the left wall. Taking into account the given Reynolds number of 1000, the kinematic viscosity of 0.001 m^2s^{-1} is calculated from the top wall velocity and given geometry data from Figure 1.

The numerical solution of this technical problem involves following points:

- The governing equations have to be solved in a dimensionless form.
- Finite difference discretisation scheme of 4th order in space and 2nd order in time is to apply.
- The calculation has to be carried out until a steady condition is reached.
- The solution of the pressure field is to obtain by applying Poisson's equation, in doing so two different methods have to be applied namely an iterative and a direct method.

2 Governing equations

In order to solve the above mentioned technical problem, following governing equations for incompressible fluids can be applied:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{2.1}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{2.2}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \tag{2.3}$$

Taking into account the characteristic values L, U_∞, and ρ_∞ the dimensionless values u^*, v^*, x^*, y^*, p^*, ρ^* and t^* can be calculated as follows:

$$u^*=\frac{u}{U_\infty},\ v^*=\frac{v}{U_\infty} \tag{2.4}$$

$$x^*=\frac{x}{L},\ y^*=\frac{y}{L} \tag{2.5}$$

$$\rho^*=\frac{\rho}{\rho_\infty},\ p^*=\frac{p}{\rho_\infty U_\infty^2},\ t^*=\frac{t\,U_\infty}{L} \tag{2.6}$$

Applying the dimensionless values the governing equations can be written in the following way:

$$\left(\frac{U_\infty}{L}\right)\frac{\partial u^*}{\partial x^*}+\left(\frac{U_\infty}{L}\right)\frac{\partial v^*}{\partial y^*}=0 \tag{2.7}$$

$$\left(\frac{U_\infty U_\infty}{L}\right)\frac{\partial u^*}{\partial t^*}+\left(\frac{U_\infty U_\infty}{L}\right)u^*\frac{\partial u^*}{\partial x^*}+\left(\frac{U_\infty U_\infty}{L}\right)v^*\frac{\partial u^*}{\partial y^*}=$$
$$-\left(\frac{\rho_\infty U_\infty U_\infty}{\rho_\infty L}\right)\frac{1}{\rho^*}\frac{\partial p^*}{\partial x^*}+\left(\frac{U_\infty}{L^2}\nu\right)\left(\frac{\partial^2 u^*}{\partial x^{*2}}+\frac{\partial^2 u^*}{\partial y^{*2}}\right) \tag{2.8}$$

$$\left(\frac{U_\infty U_\infty}{L}\right)\frac{\partial v^*}{\partial t^*}+\left(\frac{U_\infty U_\infty}{L}\right)u^*\frac{\partial v^*}{\partial x^*}+\left(\frac{U_\infty U_\infty}{L}\right)v^*\frac{\partial v^*}{\partial y^*}=$$
$$-\left(\frac{\rho_\infty U_\infty U_\infty}{\rho_\infty L}\right)\frac{1}{\rho^*}\frac{\partial p^*}{\partial y^*}+\left(\frac{U_\infty}{L^2}\nu\right)\left(\frac{\partial^2 v^*}{\partial x^{*2}}+\frac{\partial^2 v^*}{\partial y^{*2}}\right) \tag{2.9}$$

Equations (2.7), (2.8), and (2.9) can be further transformed to:

$$\frac{\partial u^*}{\partial x^*}+\frac{\partial v^*}{\partial y^*}=0 \tag{2.10}$$

$$\frac{\partial u^*}{\partial t^*}+u^*\frac{\partial u^*}{\partial x^*}+v^*\frac{\partial u^*}{\partial y^*}=-\frac{\partial p^*}{\partial x^*}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u^*}{\partial x^{*2}}+\frac{\partial^2 u^*}{\partial y^{*2}}\right) \tag{2.11}$$

$$\frac{\partial v^*}{\partial t^*}+u^*\frac{\partial v^*}{\partial x^*}+v^*\frac{\partial v^*}{\partial y^*}=-\frac{\partial p^*}{\partial y^*}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v^*}{\partial x^{*2}}+\frac{\partial^2 v^*}{\partial y^{*2}}\right) \tag{2.12}$$

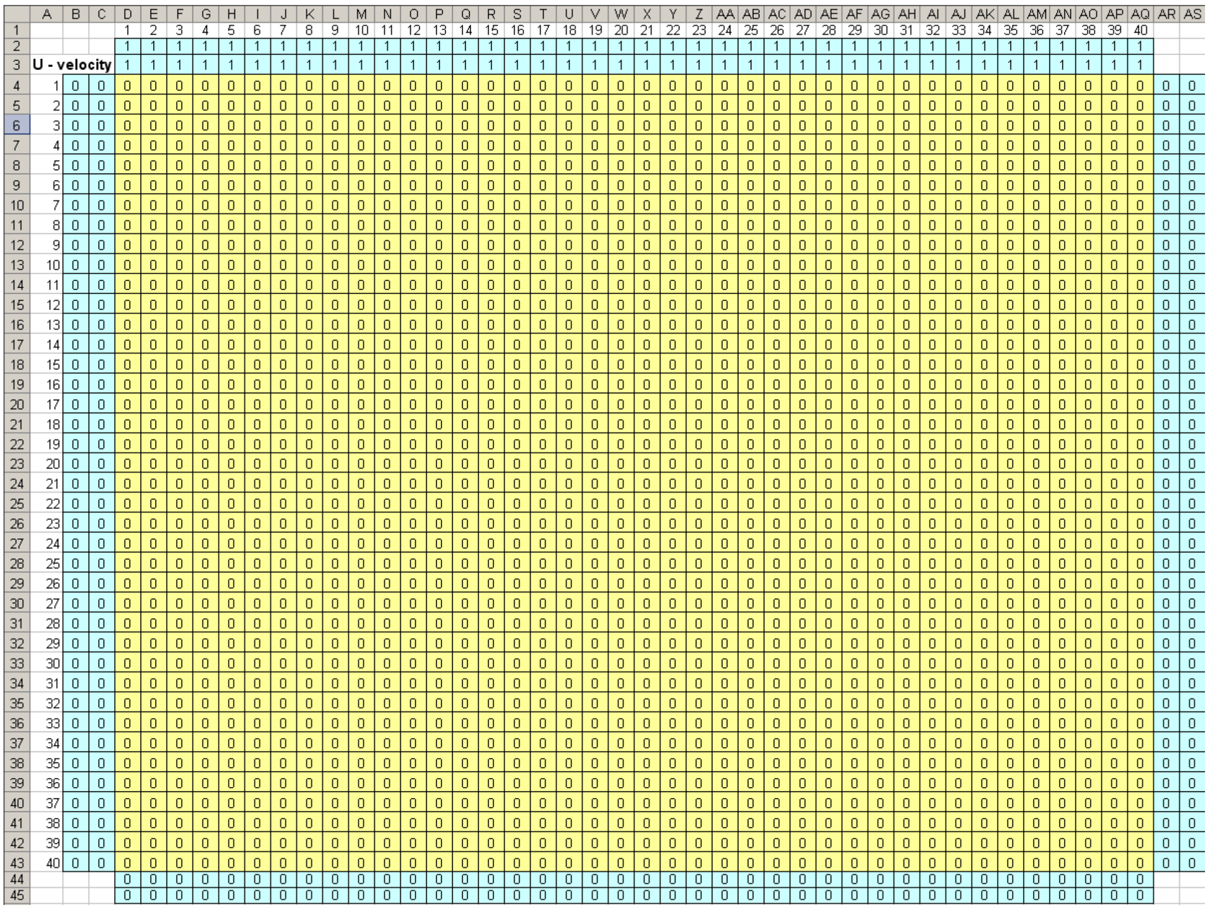
Re is the Reynolds number and is defined as follows:

$$\mathrm{Re}=\frac{U_\infty L}{\nu} \tag{2.13}$$

In the following chapters the dimensionless equations (2.10), (2.11), and (2.12) are written without “star” notation.

3 Numerical solution

The method which was applied for numerical solution of governing equations is the Finite Difference Method (FDM) employed by Euler already in 1768. In this method, the Taylor series expansion, named after the English mathematician Brook Taylor (1685-1731), is used for the discretisation of derivatives of variables. Essential advantages are its simplicity and the possibility to easily obtain both higher order approximation and accuracy. A drawback of this method is that it needs a structured grid, so that the application is strongly limited for industry-relevant simulations with complex geometries. Nevertheless, the generation of a structured 2D grid is a very simple task, so that the method can be applied without limitations. The code is written in Visual Basic programming language and it is implemented in Microsoft Excel.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS
1				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40		
2				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
3	U - velocity			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
45				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 2: Grid and field of u-velocity component (x-direction)

3.1 Grid generation

For both the grid generation process and variable storage the cells in Microsoft Excel program are used. The internal flow field consists of 40x40 cells, whereas for each wall additionally two cell layers (so-called ghost cells) are added in order to include the wall boundary condition. All together three cell fields are required, namely u-velocity (x-direction) v-velocity (y-direction) and the pressure field. Figure 2 shows the grid and the field of the u-velocity component The cells of the internal flow field are presented with yellow color and ghost boundary cells with blue color respectively. The v-velocity and the pressure field are generated in the same way. This means that the velocities and the pressure data are stored on the same places. This fact is taken into account by consideration of a displaced or a “staggered” grid for the velocity components.

3.2 Staggered grid

In order to avoid a decoupling of velocity and pressure a staggered grid is used, see Anderson [2] and Patankar [3]. In the case of a non-staggered grid the variables are stored at same grid points, see Figure 3 (a). The velocity components are presented with arrows (blue and green colors) and the pressure data with red dots. In a staggered grid, the velocity components are calculated for the points that lie on the faces of the control volume, see Figure 3 (b). This is realized in the way that the u-velocities are staggered $\Delta x/2$ in x-direction and the v-velocities $\Delta y/2$ in y-direction respectively.

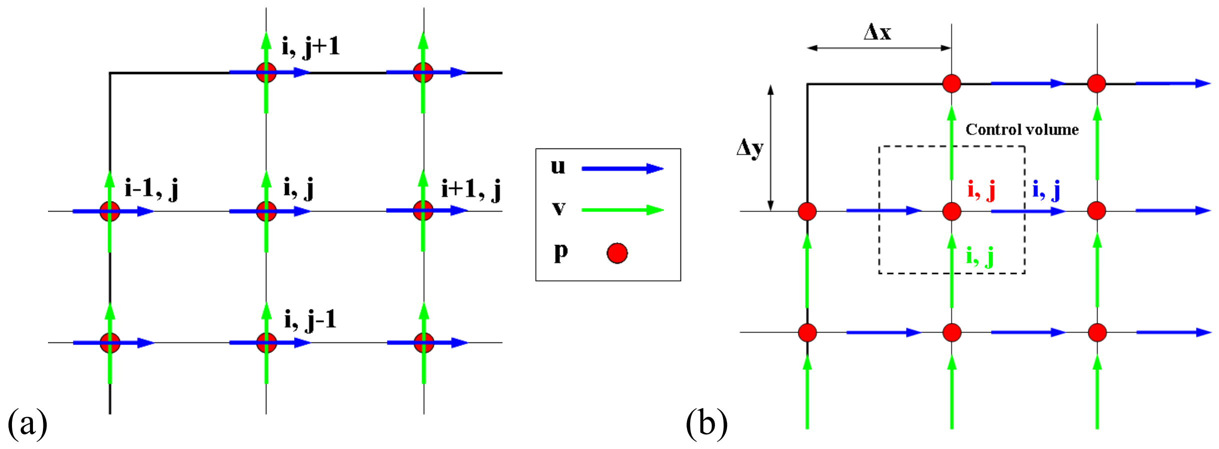


Figure 3 : Location of variables in a non-staggered grid (a) and in a staggered grid (b)

3.3 Initial and boundary conditions

After generation of the flow fields the initial and boundary condition can be defined. The internal field of the velocity components as well as of the pressure is set to zero. The ghost boundary cells for velocity components of three stationary walls are also set to zero, whereas the u-velocity component of the moving (top) wall is set to 1 m/s and the v-velocity component to zero respectively. The definition of the pressure boundary condition for ghost cells is not required due to definition of zero-gradient at walls, so that this condition is directly taken into account in the solving process of Poisson's equation. In order to achieve temporal accuracy and numerical stability a Courant number of less than 1 is needed. Courant number is defined as:

$$Co = \frac{|U| \cdot \Delta t}{\Delta x} \qquad \textbf{(3.1)}$$

Where, Δt is the time step, $|U|$ is the magnitude of the velocity and Δx is the cell size. The flow velocity varies across the domain and it must be defined that Courant number is less than 1 everywhere. Therefore, the Δt is calculated for the worst case. Due to the fact that the cell size is fixed the maximum Courant number will occur next to the lid where the velocity approaches 1 m/s. For this case Courant number can be calculated as follows:

$$\Delta t = \frac{Co \cdot \Delta x}{|U|} = \frac{1 \cdot 0.025}{1} = 0.025 \qquad \textbf{(3.2)}$$

Therefore, in this work the time step is defined less than 0.025 s.

3.4 Code algorithm

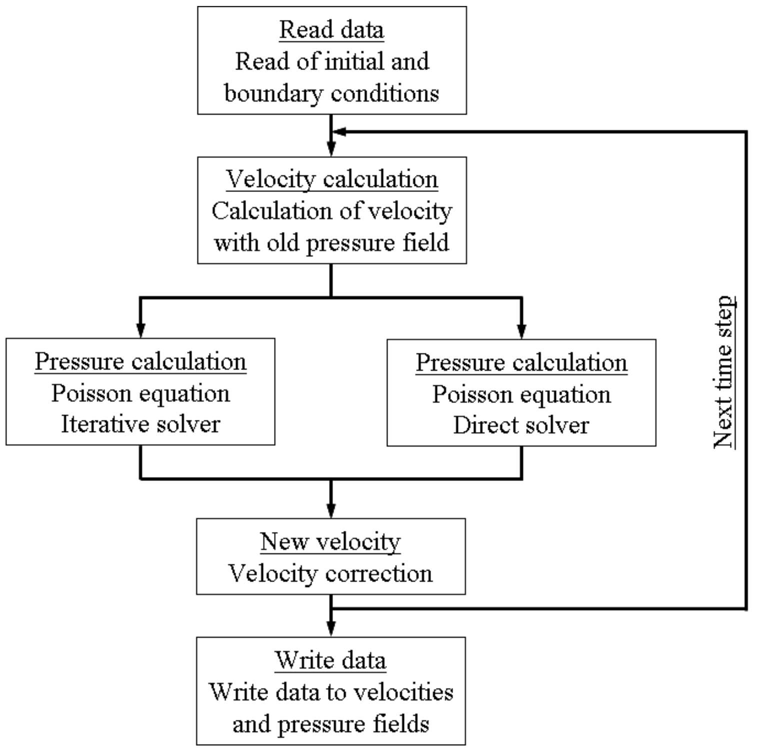


Figure 4: Algorithm

Figure 4 shows the algorithm of used code for the solution of above mentioned governing equations. First of all, the initial and the boundary condition data are read from the above described Excel tables. With these data predefined variables in Visual Basic are initialized and the time loop can be started. First step inside the time loop is the velocity calculation from momentum equations taking into account old pressure data. The new calculated velocity components do not fulfill the continuity equation, so that a velocity correction is needed. For the calculation of the pressure field two solvers are available, first an iterative and second a direct. In both cases the new pressure field is obtained from in the first step calculated velocities. Taking into account the new pressure field, the velocity correction is carried out and the calculation of the next step is possible. The simulation runs for a user defined period of time. Finally, the calculated velocities and the pressure fields are stored in Excel tables, so that data evaluation process can be carried out.

3.5 Spatial discretisation

As mentioned before, the method which was applied for numerical solution of governing equations is Finite Difference Method (FDM). In order to approximate derivatives the Taylor series expansion is applied. Equations (3.3) and (3.4) show the used first-derivative approximations on the grid for which $\Delta x = const.$ and in a general form for 2nd and 4th order accuracy. Further, equations (3.5) and (3.6) show the used second-derivative approximations. By a 4th order accuracy a four point discretisation for a first-derivative approximation is required and a five point discretisation for a second-derivative approximation respectively.

$$\left.\frac{\partial \phi}{\partial x}\right|_i = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} + O\left(\Delta x^2\right) \tag{3.3}$$

$$\left.\frac{\partial \phi}{\partial x}\right|_i = \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{12\Delta x} + O\left(\Delta x^4\right) \tag{3.4}$$

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} + O\left(\Delta x^2\right) \tag{3.5}$$

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i = \frac{-\phi_{i+2} + 16\phi_{i+1} - 30\phi_i + +16\phi_{i-1} - \phi_{i-2}}{12\Delta x^2} + O\left(\Delta x^4\right) \tag{3.6}$$

3.6 Temporal discretisation

Taking into account equation (3.7) written in general form, the 1st order accurate temporal discretisation (Euler) of time the dependent term is given in equation (3.8) and the 2nd order (Leapfrog) in equation (3.9) respectively.

$$\frac{\partial f(x,t)}{\partial t} = F(x,t,f) \tag{3.7}$$

$$f^{n+1} = f^n + \Delta t \, F^n + O\left(\Delta t\right) \tag{3.8}$$

$$f^{n+1} = f^{n-1} + 2\Delta t \, F^n + O\left(\Delta t^2\right) \tag{3.9}$$

3.7 Discretisation of governing equations

Governing equations (2.11) and (2.12) can be transformed in the following form:

$$\frac{\partial u}{\partial t} = \underbrace{-u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y}}_{F_u} + \underbrace{\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)}_{D_u} - \frac{\partial p}{\partial x} = F_u + D_u - \frac{\partial p}{\partial x} \tag{3.10}$$

$$\frac{\partial v}{\partial t} = \underbrace{-u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y}}_{F_v} + \underbrace{\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)}_{D_v} - \frac{\partial p}{\partial y} = F_v + D_v - \frac{\partial p}{\partial y} \qquad \textbf{(3.11)}$$

The convective terms F_u and F_v as well as the diffusive terms D_u and D_v can be written in discretised form for 2nd order accuracy as follows:

$$F_u = -u_{i,j}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \qquad \textbf{(3.12)}$$

$$F_v = -u_{i,j}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - v_{i,j}\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \qquad \textbf{(3.13)}$$

$$D_u = \frac{1}{\mathrm{Re}}\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right) \qquad \textbf{(3.14)}$$

$$D_v = \frac{1}{\mathrm{Re}}\left(\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{\Delta x^2} + \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{\Delta y^2}\right) \qquad \textbf{(3.15)}$$

Pressure terms in a discretised form for 2nd order accuracy are given as follows:

$$\frac{\partial p}{\partial x} = \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x} \quad \text{and} \quad \frac{\partial p}{\partial y} = \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y} \qquad \textbf{(3.16)}$$

The 4th order accurate discretisations can be realized analog to 2nd applying equations (3.4) and (3.6) respectively.

3.8 Poisson's equation

In order to solve the pressure field at a new time step an additionally equation is needed, so-called Poisson's equation, which is derived from the governing equations, which can be also written in the following way:

$$\nabla \cdot U = 0 \qquad \textbf{(3.17)}$$

$$\frac{\partial U}{\partial t}+\underbrace{\nabla\cdot(UU)}_{F}=-\frac{1}{\rho}\nabla p+\underbrace{\nu\nabla^2 U}_{D}\text{, with } U=\begin{pmatrix}u\\ v\end{pmatrix} \qquad \textbf{(3.18)}$$

The time dependent term in the momentum equation (3.18) can be descretized applying equation (3.8), so that equation (3.18) can be written as:

$$\frac{U^{n+1}-U^n}{\Delta t}=(D-F)-\frac{1}{\rho}\nabla p \qquad \textbf{(3.19)}$$

Equation (3.19) can be further split into following two equations:

$$\frac{U^*-U^n}{\Delta t}=(D-F) \quad \Rightarrow \quad U^*=U^n+\nabla t\cdot(D-F) \qquad \textbf{(3.20)}$$

$$\frac{U^{n+1}-U^*}{\Delta t}=-\frac{1}{\rho}\nabla p \quad \Bigg|\,\nabla\cdot \qquad \textbf{(3.21)}$$

At the beginning of the time step loop the velocity U^* from equation (3.20) is firstly calculated. This velocity does not fulfill the continuity equation (3.17), so a correction is required. First of all the pressure field at the new time step has to be calculated. Applying the ∇-operator in equation (3.21), it can be transformed into:

$$\frac{\nabla\cdot U^{n+1}-\nabla\cdot U^*}{\Delta t}=-\frac{1}{\rho}\nabla\cdot(\nabla p) \qquad \textbf{(3.22)}$$

Taking into account the condition at the new time step $n+1$, the velocity U^{n+1} must fulfill the continuity equation, so that equation (3.22) can be reduced to the following equation (so-called Poisson's equation):

$$\nabla^2 p=\frac{\rho}{\nabla t}\nabla\cdot U^* \qquad \textbf{(3.23)}$$

The new pressure field is calculated applying equation (3.23) and the velocity U^*. The new velocity U^{n+1} at time step $n+1$ is calculated from equation (3.21) and the new pressure data obtained from Poisson's equation, namely:

$$U^{n+1} = U^* - \frac{\Delta t}{\rho} \nabla p \tag{3.24}$$

3.9 Solver

For the solving process of Poisson's equation, two methods are implemented, namely an iterative and a direct. Both methods are used to solve a linear system of equations, which can be written in following form:

$$Ax = b \tag{3.25}$$

Where

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{3.26}$$

The iterative solvers are Jacobi, Gauss-Seidel, Successive Overrelaxation (SOR), Line Successive Overrelaxation (LSOR), Alternating Direction Implicit (ADI), Conjugate Gradient Methode (CG), Biconjugate Gradient Method (BCG), etc. The direct solvers are Gaussian elimination, LU-decomposition, Thomas algorithm etc.

In this work iterative Gauss-Seidel solver and direct LU-decomposition are implemented and used.

3.9.1 Gauss-Seidel

The Gauss–Seidel method is an iterative and very similar method to the Jacobi method. In contrast to the latter, the Gauss-Seidel uses during the iteration process the elements at $n+1$ time level that have already been computed. With this method the number of iterations is decreased, so that compared to the Jacobi method iterations can be reduced up to 50%, see Steiner [1].

The values at $n+1$ time level can be computed sequentially using forward substitution:

$$x_i^{n+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} \phi_j^{n+1} - \sum_{j=i+1}^{N} a_{ij} \phi_j^{n} \right) \tag{3.27}$$

The iterative process runs until a convergent condition of the pressure at time level $n+1$ is reached. Convergence is reached when the following relation is fulfilled:

$$MAX \left(x_i^{n+1}\Big|_{k+1} - x_i^{n+1}\Big|_{k} \right) \leq r \tag{3.28}$$

Where index k describes the number of iteration and r is the break criteria.

3.9.2 LU-decomposition

The LU-decomposition is a Gaussian elimination method which includes decomposition of a matrix A. Generally, the process of Gaussian elimination consists of two steps. The first step reduces a given system to a triangular matrix applying elementary row operations. The second step uses back-substitution to find the solution of the system. The first step is very time consuming and it has to be carried out for every time step. The advantage of LU-decomposition is that the decomposition of matrix A has to be carried out only once at the beginning of the calculation (only for the cases of a time independent matrix A). In contrast to Gaussian elimination, the LU-decomposition consists of three steps. In the first step the LU-decomposition of matrix A occurs resulting in the following form:

$$A = LU$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \tag{3.29}$$

As mentioned before, this step is only carried out once. Taking into account matrices L and U, the system of linear equations can be written:

$$Ax = b \quad \Rightarrow \quad LUx = b \tag{3.30}$$

In order to solve the equations for given matrices A and b, two additional steps are required. First, for a matrix y following equation is solved:

$$Ly = b \tag{3.31}$$

Whereas the component y is calculated using forward substitution applying following equation:

$$y_i = \frac{1}{l_{ii}}\left(b_i - \sum_{k=1}^{i-1} l_{ik} \cdot y_k\right) \tag{3.32}$$

Finally, in the last step for the needed solution matrix x following equation is solved:

$$Ux = y \tag{3.33}$$

The component x is calculated using backward substitution applying following equation:

$$x_i = \frac{1}{u_{ii}}\left(y_i - \sum_{k=i+1}^{n} u_{ik} \cdot x_k\right) \tag{3.34}$$

Summarized, the LU-decomposition is computationally efficient only when multiple-time-solutions of linear equations for different matrices b are required. This means that the LU-decomposition of matrix A is carried out only once at the start of the calculation, in contrast to Gaussian elimination which needs elimination for every time step.

4 Results

The calculation is carried out until a steady condition, namely, until a non-changeing solution of all calculated variables (velocity components and pressure) is reached. This condition is achieved after about 30 second. Figure 5 shows velocities and pressure trend at point x=0.975 and y=0.925 over 30 seconds. It can be seen that the u-velocity achieves a steady condition after about 15 second and the v-velocity after about 5 second respectively. In contrast to that the pressure achieves a converged condition after 25 second.

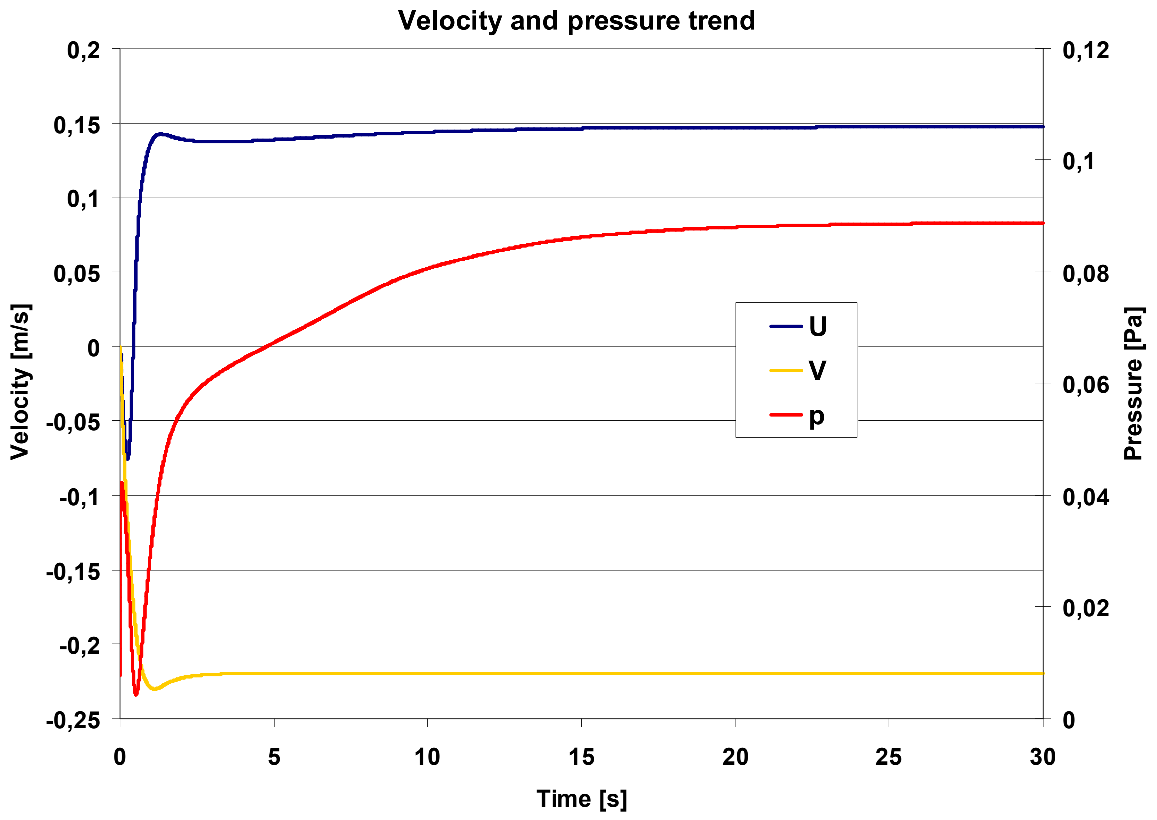


Figure 5 : Velocity and pressure trend - point x=0.975m and y=0.925m

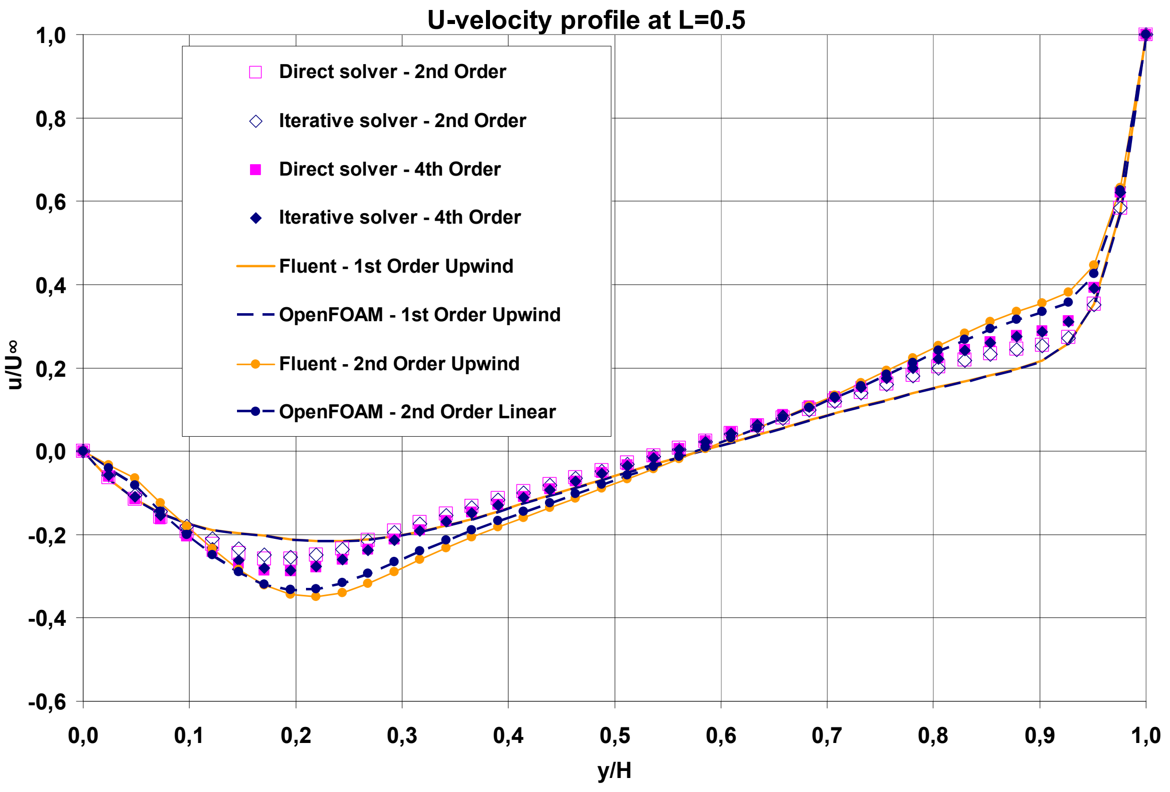


Figure 6: Comparison of U-velocity profiles at L=0.5 m

In order to validate the code the simulation results are compared with the calculations carried out applying the commercial CFD code Fluent as well as the open source CFD code OpenFOAM.

Figure 6 shows a comparison of u-velocity profiles at $L = 0.5$ m between direct and iterative solvers used in this work as well as a comparison with results which are obtained with the commercial CFD code Fluent and open source code OpenFOAM. Generally, an acceptable agreement between all curves is visible. Nevertheless, the biggest deviation is visible at $y/H = 0.2$ as well as at $y/H = 0.9$. The blue and yellow line show the 1st and 2nd order accurate calculations carried out with conventional CFD code Fluent and OpenFOAM. In above mentioned points the deviation between these results can be also seen. Blue and pink points present results calculated with used code in this work applying the 2nd and 4th order accurate scheme. Generally, a very small deviation between these schemes is visible. However, the presented results show an acceptable correlation with data which are calculated with Fluent and OpenFOAM, which further confirms tolerable accurateness of used 2D CFD code in this work.

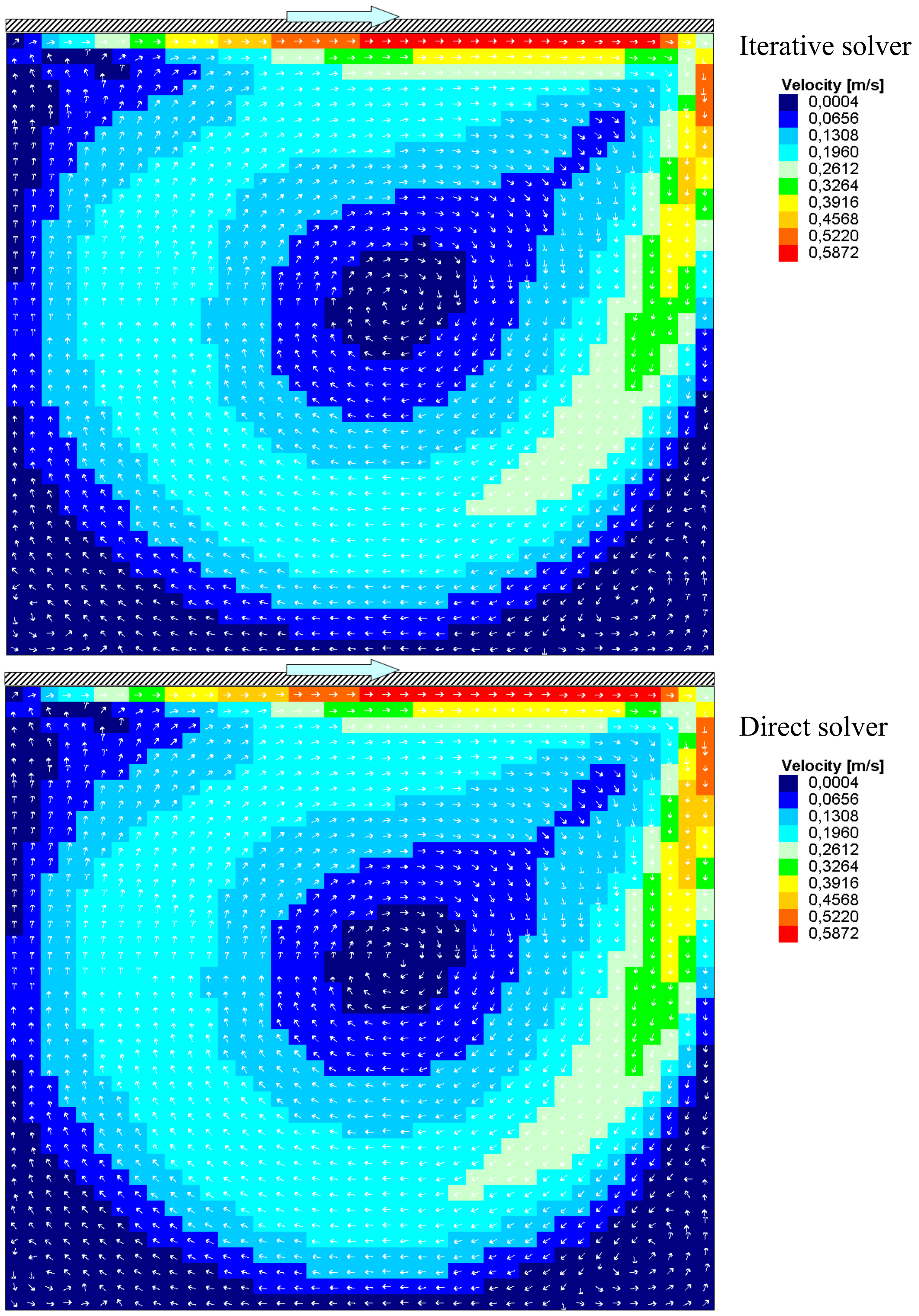
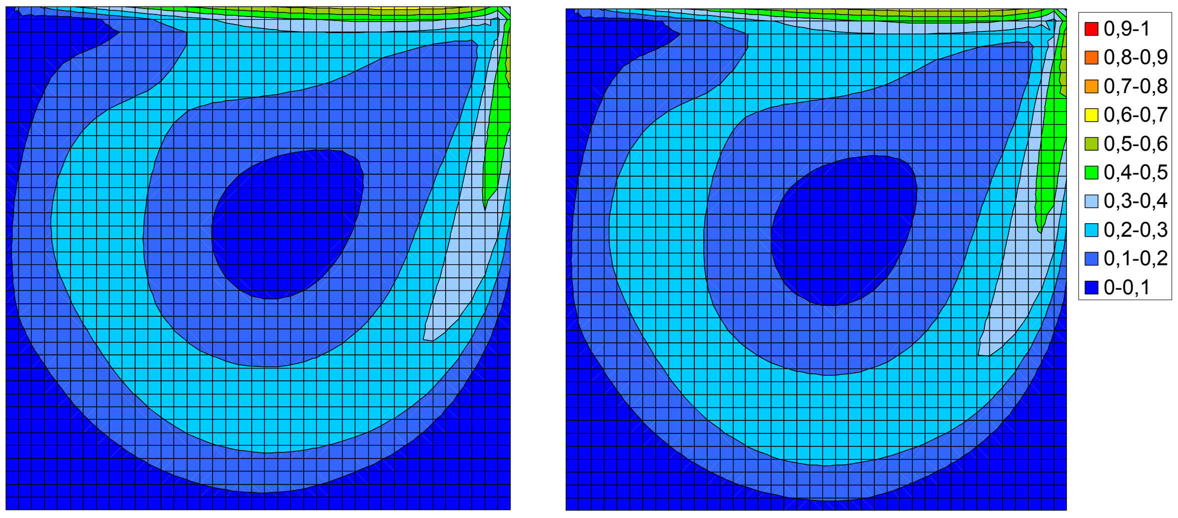


Figure 7: **Vector field of magnitude velocity given in m/s**

Figure 7 shows a comparison of velocity vector fields between results obtained with iterative and direct solvers. An identical flow situation with three vortices can be seen. The biggest vortex is positioned almost in the middle of the geometry. Two small vortices are visible in the corners, which are bordered by bottom wall and left and right walls respectively. The highest velocity is visible next to the lid where the velocity approaches approximately 0.58 m/s.



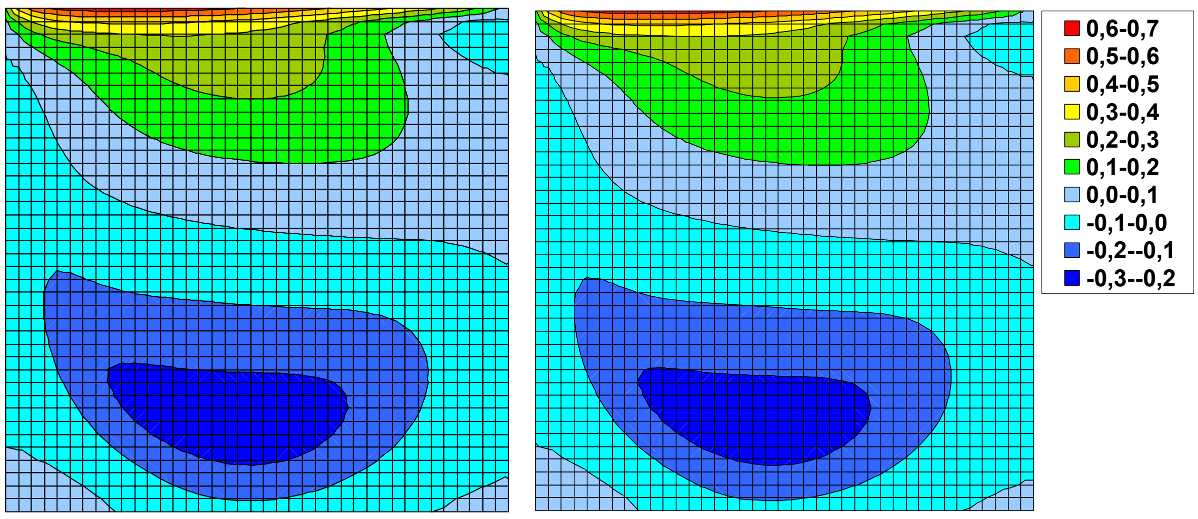
(a) Iterative solver (b) Direct solver

Figure 8: Contours of magnitude velocity given in m/s

Figure 8 presents contours of magnitude velocity and a comparison between iterative and direct solvers. In this case a good correlation between iterative and direct solvers is also visible. In this contours only the biggest vortex is visible, due to limitation caused by post-processing in Microsoft Excel.

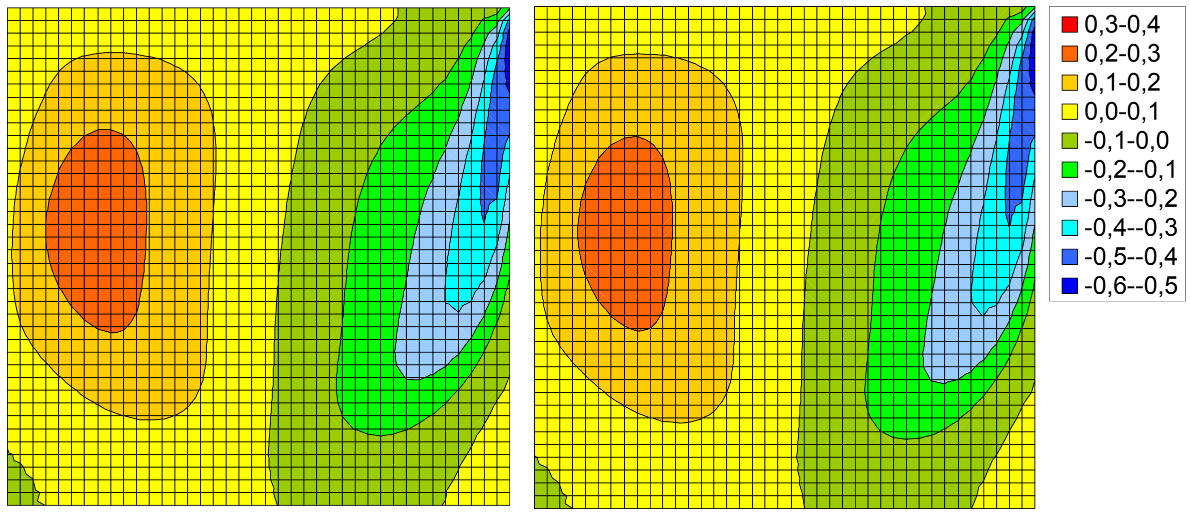
Figure 9 and Figure 10 show contours of u-velocity and v-velocity as well as a comparison between iterative and direct solvers. In these cases a good agreement between iterative and direct solver can be also seen. The highest u-velocity level is visible next to the lid where the velocity approaches approximately 0.7 m/s. In contrast to that, the v-velocity reaches its highest level of -0.6 m/s next to the wall on the right side, see Figure 10.

Finally, Figure 11 presents pressure contours as well as a comparison of iterative and direct solvers. The highest pressure level of approximately 0.24 is visible in the top corner on the right side. A good correlation between both solvers is also achieved in this case.



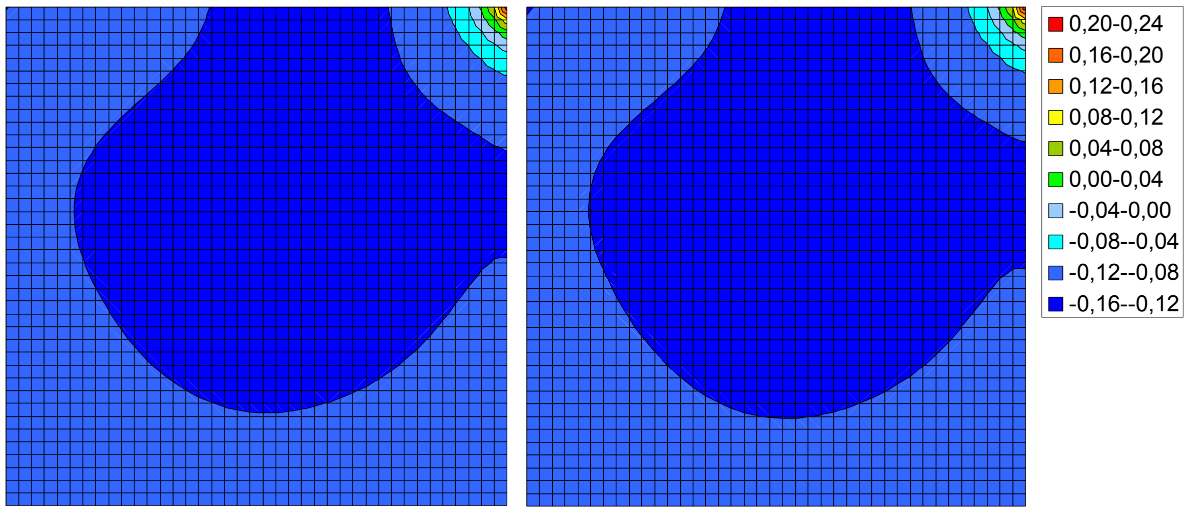
(a) Iterative solver (b) Direct solver

Figure 9: Contours of U-velocity given in m/s



(a) Iterative solver (b) Direct solver

Figure 10: Contours of V-velocity given in m/s



(a) Iterative solver (b) Direct solver

Figure 11: Pressure contours

5 Summary and conclusion

In this work a lid driven cavity flow is investigated applying numerical methods. The flow is considered as laminar, isothermal, and incompressible in a two-dimensional domain. In order to solve governing equations Finite Difference Method is applied. The code is written in Visual Basic programming language and it is implemented in commercial Microsoft Excel program. The solution of the pressure field is obtained applying Poisson's equation, whereas two different methods are used, namely an iterative and a direct solver. For this purpose iterative Gauss-Seidel and direct LU-decomposition solvers are implemented. It can be concluded that Gauss-Seidel solver is more appropriate, primary due to less computational effort. The LU-decomposition solver is very time consuming and causes a tenfold increase of computational effort. This increase of calculation time is mainly caused by the decomposition of matrix A, whereas an increase of computational effort during solving process is not dramatically. From accuracy point of view, both solvers show a very good correlation and almost identical results are obtained. In order to validate used code the calculation results are compared with simulations carried out using commercial CFD code Fluent as well as open source CFD code Open-FOAM. A comparison of u-velocity profile at L=0.5m shows an acceptable agreement with conventional CFD codes which further confirms the accuracy of used 2D CFD code in this work.

6 References

[1] Steiner H., “Numerische Methoden in Strömungslehre und Wärmeübertragung”, LV-Nr.: 321.023, Institut für Strömungslehre und Wärmeübertragung, Technische Universität Graz, 2009

[2] Anderson A. D., Tannehill C. J. and Pletcher H. R., “Computational Fluid Mechanics and Heat Transfer”, Series in comp. methods in mech. and therm. sciences, ISBN 0-07-050328-1, 1984

[3] Patankar V. S., “Numerical heat transfer and fluid flow”, Series in comp. methods in mech. and therm. sciences, ISBN 0-07-048740-5, 1980