Praktikum Wissenschaftliches Rechnen (CFD, Final Project)

Group 9: Breznik, E., Cheng, Z., Ni, W., Schmidbartl, N.

1 The Topic

For the final project, our group chose to extend the Navier Stokes code to be used in 3D for arbitrary geometries, with the ability of handling the Free Surface flow scenarios. We put a lot of attention to designing a code that will work in 3D for as many different scenarios as possible. That is why we did not code a special set of boundaries for just a few more known and frequently simulated situations, but rather made them completely dependent on the input geometry file.

TODO:comment on not doing free surface

1.1 Input

The parameter file that was used as input in our previous worksheets remained, with just slight changes due to the additional dimension. Only notable difference is, that we now take as an input also a scalar $velIN$, and a vector $velMW$. First one represent the velocity at the inflow boundaries, and the second one the wall velocity for the moving wall boundaries.

To make our code able to handle truly arbitrary scenarios, we designed so that it allows for any sort of implemented boundary conditions to be employed in any domain cell - so even the obstacles inside the domain can have arbitrary boundaries, as opposed to only allowing that on the domain walls (as in workseet 3). The standard boundary conditions, that we implemented, are

- no slip,
- free slip,
- inflow,
- outflow,
- moving wall.

Cell type	Number code
water	0
air	1
no-slip	2
free-slip	3
inflow	4
outflow	5
moving wall	6

Table 1: Number representations of different possible cell types.

To specify, where which of them is applied, we used special numbering of cells when generating our input pgm files, which can be seen from table 1.1, so our geometries are represented by a grayscale image with 7 levels of brightness. Since we are working in three dimensions, this image consists of a sequence of 2D (xy plane) images. The orientation of coordinate system and an example picture for a small lid driven cavity case can be seen on figure 1.1. One geometry picture set thus needs to contain $(imax + 2) \times (jmax + 2) \times (kmax + 2)$ numbers in $(jmax + 2) \times (kmax + 2)$ rows, and is typically very big.

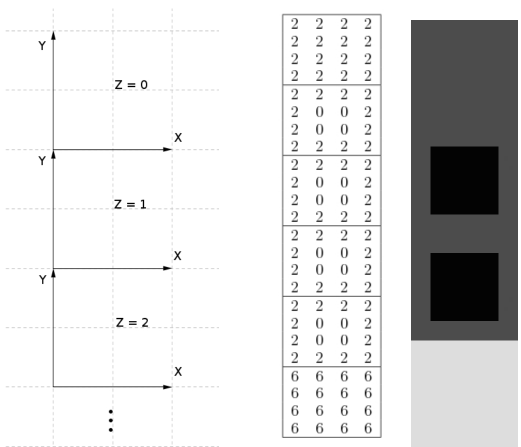

Figure 1: The structure of our input pgm files and a lid driven cavity example case.

2 Implementation

2.1 Boundary conditions for the velocities

We have 5 different boundary conditions, namely no-slip, free-slip, outflow, inflow, moving wall. We implement them for the 6 boundaries of our 3D domain, as well as for internal boundaries.

2.1.1 No-Slip boundary condition

For no-slip conditions, the fluid vanishes at the boundary. As a consequence, both the velocity component normal to and parallel to the boundary are zero. Using staggered grid, the discrete velocity components normal to and parallel to the boundary lie directly at the boundary, whilst the one parallel to the boundary is the average of the boundary cell's and its neighbouring fluid cell's velocity components.

Therefore, for the `B_O` case, we set the following boundary conditions:

$$\begin{array}{lll} u_{i,j,k} = 0, & v_{i,j-1,k} = -v_{i+1,j-1,k}, & v_{i,j,k} = -v_{i+1,j,k}, \\ w_{i,j,k-1} = -w_{i+1,j,k-1}, & w_{i,j,k} = -w_{i+1,j,k}, & \end{array} \tag{1}$$

and we analogously set the boundary conditions for the following 5 cases: `B_W`, `B_N`, `B_S`, `B_U` and `B_D`.

For the `B_NO` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = 0, & u_{i-1,j,k} = -u_{i-1,j+1,k}, \\ v_{i,j,k} = 0, & v_{i,j-1,k} = -v_{i+1,j-1,k}, \\ w_{i,j,k} = -\frac{1}{2}(w_{i+1,j,k} + w_{i,j+1,k}), & w_{i,j,k-1} = -\frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{2}$$

and we analogously set the boundary conditions for the following 11 cases: `B_NW`, `B_NU`, `B_ND`, `B_SO`, `B_SW`, `B_SU`, `B_SD`, `B_OU`, `B_WU`, `B_OD` and `B_WD`.

For the `B_NOU` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = 0, & u_{i-1,j,k} = -\frac{1}{2}(u_{i-1,j+1,k} + u_{i-1,j,k+1}), \\ v_{i,j,k} = 0, & v_{i,j-1,k} = -\frac{1}{2}(v_{i+1,j-1,k} + v_{i+1,j-1,k+1}), \\ w_{i,j,k} = 0, & w_{i,j,k-1} = -\frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{3}$$

and we analogously set the boundary conditions for the following 7 cases: `B_NWU`, `B_NOD`, `B_NWD`, `B_SOU`, `B_SWU`, `B_SOD`, and `B_SWD`.

2.1.2 Free-Slip boundary condition

For free-slip conditions, the fluid flows freely parallel to the boundary, but does not cross the boundary. As a consequence, the velocity component normal to the boundary is zero, as well as the normal derivative of the velocity component parallel to the wall. Using staggered grid, the discrete velocity components normal to the boundary lie directly at the boundary, whilst the one parallel to the boundary is the average of the boundary cell's and its neighbouring fluid cell's velocity components.

Therefore, for the `B_O` case, we set the following boundary conditions:

$$\begin{array}{lll} u_{i,j,k} = 0, & v_{i,j-1,k} = v_{i+1,j-1,k}, & v_{i,j,k} = v_{i+1,j,k}, \\ w_{i,j,k-1} = w_{i+1,j,k-1}, & w_{i,j,k} = w_{i+1,j,k}, & \end{array} \tag{4}$$

and we analogously set the boundary conditions for the following 5 cases: `B_W`, `B_N`, `B_S`, `B_U` and `B_D`.

For the `B_NO` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = 0, & u_{i-1,j,k} = u_{i-1,j+1,k}, \\ v_{i,j,k} = 0, & v_{i,j-1,k} = v_{i+1,j-1,k}, \\ w_{i,j,k} = \frac{1}{2}(w_{i+1,j,k} + w_{i,j+1,k}), & w_{i,j,k-1} = \frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{5}$$

and we analogously set the boundary conditions for the following 11 cases: `B_NW`, `B_NU`, `B_ND`, `B_SO`, `B_SW`, `B_SU`, `B_SD`, `B_OU`, `B_WU`, `B_OD` and `B_WD`.

For the `B_NOU` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = 0, & u_{i-1,j,k} = \frac{1}{2}(u_{i-1,j+1,k} + u_{i-1,j,k+1}), \\ v_{i,j,k} = 0, & v_{i,j-1,k} = \frac{1}{2}(v_{i+1,j-1,k} + v_{i+1,j-1,k+1}), \\ w_{i,j,k} = 0, & w_{i,j,k-1} = \frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{6}$$

and we analogously set the boundary conditions for the following 7 cases: `B_NWU`, `B_NOD`, `B_NWD`, `B_SOU`, `B_SWU`, `B_SOD`, and `B_SWD`.

2.1.3 Outflow boundary condition

For outflow conditions, the normal derivatives of both the velocity components are zero. Using staggered grid, the discrete velocity component normal to the boundary lie directly at the boundary, whilst the one parallel to the boundary is the average of the boundary cell's and its neighbouring fluid cell's velocity components.

Therefore, for the `B_O` case, we set the following boundary conditions:

$$\begin{array}{lll} u_{i,j,k} = u_{i+1,j,k}, & v_{i,j-1,k} = v_{i+1,j-1,k}, & v_{i,j,k} = v_{i+1,j,k}, \\ w_{i,j,k-1} = w_{i+1,j,k-1}, & w_{i,j,k} = w_{i+1,j,k}, & \end{array} \tag{7}$$

and we analogously set the boundary conditions for the following 5 cases: `B_W`, `B_N`, `B_S`, `B_U` and `B_D`.

For the `B_NO` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = u_{i+1,j,k}, & u_{i-1,j,k} = u_{i-1,j+1,k}, \\ v_{i,j,k} = v_{i,j+1,k}, & v_{i,j-1,k} = v_{i+1,j-1,k}, \\ w_{i,j,k} = \frac{1}{2}(w_{i+1,j,k} + w_{i,j+1,k}), & w_{i,j,k-1} = \frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{8}$$

and we analogously set the boundary conditions for the following 11 cases: `B_NW`, `B_NU`, `B_ND`, `B_SO`, `B_SW`, `B_SU`, `B_SD`, `B_OU`, `B_WU`, `B_OD` and `B_WD`.

For the `B_NOU` case, we set the following boundary conditions:

$$\begin{array}{ll} u_{i,j,k} = u_{i+1,j,k}, & u_{i-1,j,k} = \frac{1}{2}(u_{i-1,j+1,k} + u_{i-1,j,k+1}), \\ v_{i,j,k} = v_{i,j+1,k}, & v_{i,j-1,k} = \frac{1}{2}(v_{i+1,j-1,k} + v_{i+1,j-1,k+1}), \\ w_{i,j,k} = w_{i,j,k+1}, & w_{i,j,k-1} = \frac{1}{2}(w_{i+1,j,k-1} + w_{i,j+1,k-1}), \end{array} \tag{9}$$

and we analogously set the boundary conditions for the following 7 cases: `B_NWU`, `B_NOD`, `B_NWD`, `B_SOU`, `B_SWU`, `B_SOD`, and `B_SWD`.

2.1.4 Inflow boundary condition

For inflow conditions, we assume the inflow velocity is perpendicular to the inflow boundary. We implement the following 6 cases, `B_O`, `B_W`, `B_N`, `B_S`, `B_U` and `B_D`, and forbid the other cases.

For the `B_O` case, the boundary velocities are same with those the no-slip condition, except that $u_{i,j,k} = \mathtt{velIN}$, where the inflow velocity `velIN` is larger than zero.

Similarly, for the `B_W` case, the boundary velocities are same with those of the no-slip condition, except that $u_{i,j,k} = -\mathtt{velIN}$, where `velIN` is larger than zero, and we analogously set the boundary conditions for the following 4 cases: `B_N`, `B_S`, `B_U` and `B_D`.

2.1.5 Moving wall boundary condition

For moving wall conditions, again we need to restrict the moving wall directions.

When the boundary cell is like `B_O`, we set the moving wall direction to the direction of next (first nonfixed) coordinate, i.e., if $x/y/z$ is fixed, the wall moving in $y/z/x$. For example, if the flag is `B_O`, the moving wall direction is $+y$ or $-y$. Consequently, the velocities are same with those of the no-slip condition, except for the ones along this moving wall direction and parallel to the boundary. Therefore, for the `B_O` case, the following boundary conditions differ from those of the no-slip condition:

$$v_{i,j-1,k} = 2 * \mathtt{velMW}_y - v_{i+1,j-1,k}, \quad v_{i,j,k} = 2 * \mathtt{velMW}_y - v_{i+1,j,k} \tag{10}$$

, where velMW $= [\mathtt{velMW}_x, \mathtt{velMW}_y, \mathtt{velMW}_z]$ is the moving wall velocity vector, and $\mathtt{velMW}_x$ is its component along the x axis. We analogously set the boundary conditions for the following 5 cases: B_W, B_N, B_S, B_U and B_D.

When the boundary cell is like B_NO, we set the directions of moving walls N and O in a "circular" way. For example, if N moves in the $+x$ direction, then O moves in the $-y$ direction. Alternatively, if N moves in the $-x$ direction, then O moves in the $+y$ direction. Therefore, for the B_NO case, the following boundary conditions differ from those of the no-slip condition:

$$\begin{aligned} u_{i,j,k} &= 2 * \mathtt{velMW}_x - u_{i,j+1,k}, \quad u_{i-1,j,k} = 2 * \mathtt{velMW}_x - u_{i-1,j+1,k}, \\ v_{i,j,k} &= 2 * \mathtt{velMW}_y - v_{i+1,j,k}, \quad v_{i,j-1,k} = 2 * \mathtt{velMW}_y - v_{i+1,j-1,k} \end{aligned} \tag{11}$$

and we analogously set the boundary conditions for the following 11 cases: B_NW, B_NU, B_ND, B_SO, B_SW, B_SU, B_SD, B_OU, B_WU, B_OD and B_WD.

However, for the moving wall condition we forbid the boundary conditions for the following 8 cases: B_NOU, B_NWU, B_NOD, B_NWD, B_SOU, B_SWU, B_SOD, and B_SWD.

2.2 Boundary condition for the pressure

On the other hand, for all the 5 aforementioned boundary conditions, the boundary values for the pressure are derived from the discretized momentum equation and result in discrete *Neumann* conditions.

Therefore, for the B_O case, we set the following boundary conditions:

$$F_{i,j,k} = u_{i,j,k}, \quad p_{i,j,k} = p_{i+1,j,k} \tag{12}$$

and we analogously set the boundary conditions for the following 5 cases: B_W, B_N, B_S, B_U and B_D.

For the B_NO case, we set the following boundary conditions

$$F_{i,j,k} = u_{i,j,k}, \quad G_{i,j,k} = v_{i,j,k}, \quad p_{i,j,k} = \tfrac{1}{2}(p_{i+1,j,k} + p_{i,j+1,k})) \tag{13}$$

and we analogously set the boundary conditions for the following 11 cases: B_NW, B_NU, B_ND, B_SO, B_SW, B_SU, B_SD, B_OU, B_WU, B_OD and B_WD.

For the B_NOU case, we set the following boundary conditions

$$\begin{aligned} F_{i,j,k} = u_{i,j,k}, \quad G_{i,j,k} &= v_{i,j,k}, \quad H_{i,j,k} = w_{i,j,k}, \\ p_{i,j,k} &= \tfrac{1}{3}(p_{i+1,j,k} + p_{i,j+1,k} + p_{i,j,k+1}) \end{aligned} \tag{14}$$

and we analogously set the boundary conditions for the following 7 cases: B_NWU, B_NOD, B_NWD, B_SOU, B_SWU, B_SOD, and B_SWD.

3 Problems, current state and future work

4 Results

References

[1] Griebel, M., Dornsheifer, T., Neunhoeffer, T.: *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM, **1998**.

[2] Hirt, C. W., Nichols, B. D.: *Volume of Fluid Method for the Dynamics of Free Boundaries*. Journal of Computational Physics **39** (1981).

[3] Hirt, C. W., Nichols, B. D., Hotchkiss, R. S.: *SOLA-VOF: A solution Algorithm for Transient Fluid Flow with Multiple Free Boudaries*. LASL, **1980**.