**Introduction**

After experiencing 2D Navier-Stokes Equation, Navier-Stokes Equation in 3D has been implemented in our group project. C++ framework is replacing the traditional C programming. Then 3D calculation for velocity(u,v,w), force(F,G,H) ,pressure, residual and temperature has been reformed. Due to increase in the complexity of arbitrary geometry in 3D, a new algorithm was implemented. In order to parallelize the program, CUDA solver was built to decrease the runtime to calculate in 3D image. In the following sections, details will be explained and results will be displayed.

**3D incompressible Navier-Stokes equation**

Non-stationary incompressible viscous fluid flow is described in 3-dimensional Navier-Stokes Equation. The analysis was carried out in Cartesian coordinates. The quantities are computed as u,v,w and F,G,H as x,y,z directions. Then the 3D momentum ,continuity equations are:

3D Momentum equations. No:1,2,3

3D Continuity equation. No 4

1. Force F and velocity u calculation in x direction:

Velocity w is essentially considered into the calculation then, the discretization for derivatives of u,v and w has to be with respect to directions x,y and z respectively.

F calculation equation: No6

u calculation equation: No29

1. Discretization for F

The midpoints of 3 directions will be towards to evaluate the derivative of u,v and w:

Equations : No 9 -14

1. Energy equation and Discretization

The energy equation in 3D and its discretization are easy to compute

Energy equation: No 32

Discretization: No 33 and 36

1. SOR Solver

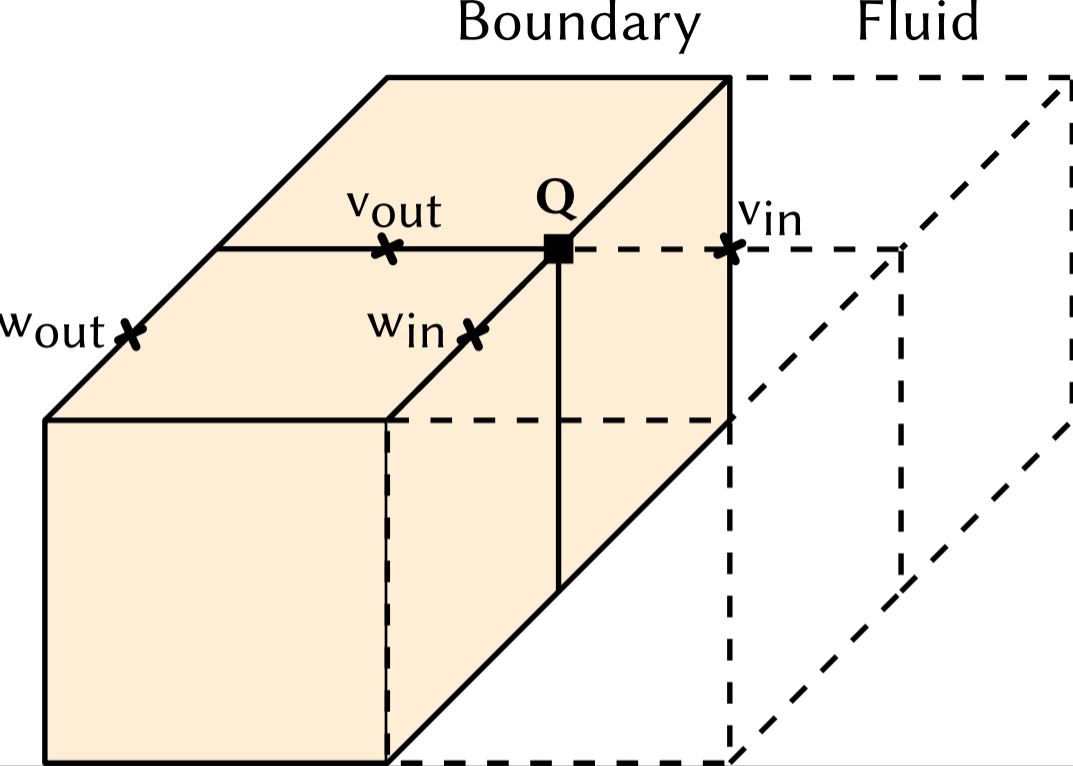
Pressure equation: No 27

Residual: No 28

**Boundary Conditions**

No Slip: Under the no-slip conditions, zero velocities at boundary. So we can set it to zero lying right on the boundary. Moreover, the boundary value zero is achieved by averaging the values on both sides of the boundary. Eg. Left wall case: U0jk = 0 ,V0jk = -V1jk, W0jk = -W1jk(latex);

Free Slip: it is the case when fluid flow freely parallel to the boundary but cannot across the boundary. In 3D cases, the fluid freely flow along two dimension but cannot across remaining direction. For example, the velocity u of left wall which is normal(perpendicular) to the wall is 0. Its boundary condition is to set as U0jk = 0; (latex). And other boundary conditions to set V0jk = V1jk, W0jk = W1jk(latex);



Picture 1: the free slip boundary condition of left wall. (will scan)

Outflow: the normal velocity derivatives are set to 0 at boundary. So the total velocity does not change in the direction normal to the boundary. The values of velocities at boundary can be set equal to the neighboring velocities inside the region. Eg. Left wall case: U0jk = U1jk ,V0jk = V1jk, W0jk = W1jk(latex);

Boundary conditions for pressure and temperature: the boundary conditions for the pressure derived from momentum equation, resulting in discrete Neumann conditions. The boundary conditions for temperature derived from energy equation, resulting in discrete Dirichlet Condition for cold wall and hot wall separately. Eg. Left wall case: P0jk = P1jk. T0jk = 2\*T\_c/T\_h – T1jk (latex)

**Arbitrary Geometries**

u,v,w:

Unlike the 2D cases which is necessarily to set the arbitrary geometries of velocities for 8 directions(4 lines + 4 corners of a square), there are total 26 directions (12 lines, 8 corners and 6 faces of a cube) to be set. Then setting it in 3D brought difficulties if we use the method from ws2. Therefore, we design a new algorithm to set arbitrary geometries.

We observed that we could set velocity to zero lying right on the boundary. Moreover, the boundary value zero is achieved by averaging the values on both sides of the boundary. It is the similar case to set no-slip boundary condition on walls. We will set u,v and w separately . The flag of direction is checked if the velocity is lying on. If yes, the velocity is set to zero. Then an indicator here is inserted that this velocity is not changed anymore. Otherwise, the velocity is set to equal to the negative of the neighboring velocity. For example, here is sudo code of the arbitrary geometry setting of u on left wall:

for (int i = 1; i <= imax-1; i++) {

for (int j = 1; j <= jmax; j++) {

for (int k = 1; k <= kmax; k++) {

int L\_check = 0, L1\_check = 0;

if (Flag is not fluid)) {

if (Left flag is obstacle) {

Ui-1,j,k = 0;

L\_check = 1;

}

if (Up flag is obstacle) {

if (L\_check == 0) {

Ui-1,j,k = -Ui-1,j+1,k;

L1\_check = 1;

}

if (Down flag is obstacle) {

if (L\_check == 0) {

Ui-1,j,k = -Ui-1,j-1,k;

L1\_check = 1;

}

}

if (back flag is obstacle) {

if (L\_check == 0 && L1\_check == 0) {

Ui-1,j,k = -Ui-1,j,k-1;

}

}

if (front flag is obstacle) {

if (L\_check == 0 && L1\_check == 0) {

Ui-1,j,k = -Ui-1,j,k+1;

}

}

}

}

}

}

Pressure and Temperature:

Setting the arbitrary geometry of pressure and temperature are using the same methodology. As the pressure and temperature at the direction is the average of the 6 faces involved, for example, B\_URF(up, right and front)’s pressure Pijk = (Pi+1jk + Pij+1k + Pijk+1)/3. So each flag of direction is checked and accumulated . To sum up the pressure/temperature and take the average of them. Here is the sudo code:

for (int i = 1; i <= imax; i++) {

for (int j = 1; j <= jmax; j++) {

for (int k = 1; k <= kmax; k++) {

int numDirectFlag = 0;

Real P\_temp = Real(0);

if (flag is not fluid) {

if (right flag is obstacle) {

P\_temp +=Pi+1,j,k;

numDirectFlag++;

}

if (left flag is obstacle) {

P\_temp += Pi-1,j,k;

numDirectFlag++;

}

...

if (numDirectFlag == 0) {

Pi,j,k = 0;

} else {

Pi,j,k = P\_temp / Real(numDirectFlag);

}

}

}

}

}