# **Progress Report**

Syed Ahmad Raza 2018.05.09

# 1 2D Finite Volume solver Navier-Stokes using two layers of ghost cells

#### 1.1 Ghost cells

Two layers of ghost cells were implemented in C++ using Finite Volume Method to solve the Navier-Stokes equations, as shown in the following figure for a reduced grid size.

	$y_n$											
_Dv	Dy <sub>n-1</sub> s <sub>n-1</sub> y <sub>n-1</sub>		1,n <sub>y-1</sub>	2,n <sub>y-1</sub>	3,n <sub>y-1</sub>	4,n <sub>y-1</sub>	5,n <sub>y-1</sub>	6,n <sub>y-1</sub>	7,n <sub>y-1</sub>	$n_{x-2}$ , $n_{y-1}$	n <sub>x-1</sub> , n <sub>y-1</sub>	
	Dy <sub>n-2</sub> S <sub>n-2</sub>	0,n <sub>y-2</sub>	1,n <sub>y-2</sub>	2,n <sub>y-2</sub>	3,n <sub>y-2</sub>	4,n <sub>y-2</sub>	5,n <sub>y-2</sub>	6,n <sub>y-2</sub>	7,n <sub>y-2</sub>	n <sub>x-2</sub> , n <sub>y-2</sub>	n <sub>x-1</sub> , n <sub>y-2</sub>	
	Dy <sub>7</sub>	0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	n <sub>x-2</sub> ,7	n <sub>x-1</sub> ,7	
	$Dy_6$ $y_6$	0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	n <sub>x-2</sub> ,6	n <sub>x-1</sub> ,6	
	Dy <sub>5</sub>	0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	n <sub>x-2</sub> ,5	n <sub>x-1</sub> ,5	
	Dy <sub>4</sub>	0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	n <sub>x-2</sub> ,4	n <sub>x-1</sub> ,4	
	Dy <sub>3</sub>	0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	n <sub>x-2</sub> ,3	n <sub>x-1</sub> ,3	
Di	Dy <sub>2</sub>	0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	n <sub>x-2</sub> ,2	n <sub>x-1</sub> ,2	
	$Dy_1$ $y_1$	0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	n <sub>x-2</sub> ,1	n <sub>x-1</sub> ,1	
	$Dy_0$ $y_0$ $y_0$	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	n <sub>x-2</sub> ,0	n <sub>x-1</sub> ,0	
D <sub>3</sub>		$Dx_0$	$x_1$ $Dx_1$ $s_1 \rightarrow \leftarrow Dx$	$Dx_2$	X <sub>3</sub> Dx <sub>3</sub>	X <sub>4</sub> DX <sub>4</sub>	X <sub>5</sub> $Dx_5$	$x_6$ $Dx_6$		$X_{n-2}$ $DX_{n-2}$ $S_{n-2} \leftarrow DX_{n-2}$	$Dx_{n-1}$	X <sub>n</sub>
	-DX	$U \rightarrow U X$	$S_1 \rightarrow -DX$	52					-DX	on-2 → DX	on-1 →	

Figure 1: Figure representing the utilization of two ghost cells around the grid, for grid of reduced size

## 1.2 Debugging

Earlier code had a major bug, which in that it implemented the far-side velocity boundary conditions incorrectly. The corrected velocity boundary conditions in the x and y directions for the far side are given below.

*u*-velocity for the east boundary:

$$u_{n_x-3,j} = 0 u_{n_x-2,j} = u_{n_x-3,j} (1)$$

*v*-velocity for the north boundary:

$$v_{i,n_y-3} = 0$$
  $v_{i,n_y-2} = v_{i,n_y-3}$  (2)

This can be understood with the help of the following figure.

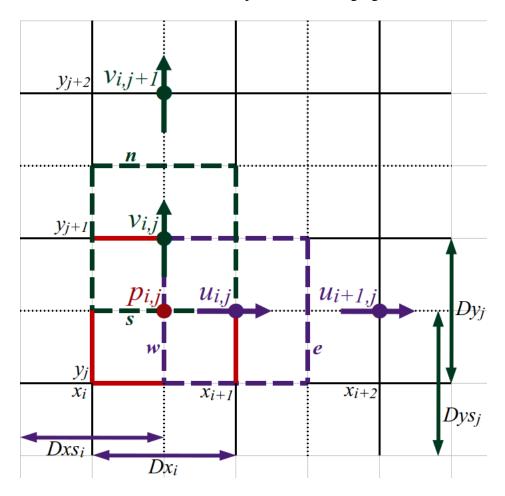


Figure 2: Representation of the utilization of two ghost cells around the grid, for grid of reduced size

#### 1.3 Results

The code was solved for the case of a lid-driven cavity flow in a two-dimensional square box. The results are shown in the following figures.

Figure 3: Comparing the solution provided by Ghia et. al versus the numerical solution for u-velocity values for all y at the center of x -axis, for a  $121 \times 121$  grid.

Figure 4: Comparing the solution provided by Ghia et. al versus the numerical solution for v-velocity values for all x at the center of x-axis, for a  $121 \times 121$  grid.

## 1.4 Ongoing tasks

- 1. Calculating the virtual force for a cylinder inside a lid-driven cavity
- 2. Employ parallel computing by using ultraMPP C++ library to execute the code on parallel cores
- 3. Solve cavity-driven flow for 3D