Help

The main idea of controlling the boundary conditions with the surface of the bodies

The array vol_inf_fb (volume information fluid or body) in the program contain information of what is placed in the control volume – fluid or this control volume is a body. The program calculate only the control volumes which are full with fluid. The control volumes are separated in two groups:

1. The control volumes, which are surrounded from control volumes which contain only fluid. Here the applying of the boundary conditions and check related to that is not needed.
2. The control volumes, which are neighbor at least one volume part of the body or are in the beginning or the end of the computational area. Here are included all checks, which are needed to be applied to account the boundary conditions. This calculation take more time, because of the included checks in loop.

In the program is used the array vol_inf_fb_bool from bool elements. The array is constructed in the following way:

```
if(vol_inf_fb[node] == fluid) vol_inf_fb_bool[node] = true;
else vol_inf_fb_bool[node] = false;
```

The input files of the program

The program is ruled by two types of files. One is EnteredData.txt and other files for bodies.

The “EnteredData.txt” file contain information about fluid, mesh, the way of solving data and boundary conditions for computational domain. Here is one example for “EnteredData.txt” file:

BC – Boundary Conditions

value	Name of variable in program	Description
0	//u_gas_nd	horizontal velocity of fluid in initial BC
0	//v_gas_nd	vertical velocity of fluid in initial BC
1	//T_gas_nd	Temperature of fluid in initial BC
1	//p_gas_nd	pressure of fluid in initial BC
1.095	//Ma	Mach number based on thermal velocity
0.666	//Pr	Prandtl number
1.666	//gamma	γ
0.606	//c_mu	c_{μ}
0.05	//Kn	Knudsen number
50	//Fr	Frud number
0	//SolveGravityField	0 – no gravity 1 – solve vertical gravity field, opposite to OY
5000	//Nt	number of time steps
9	//Nx	number of nodes on OX axis
122	//Ny	number of nodes on OY axis
50	//N_I	number of maximum iterations of step by time
50	//N_I_p_c - if(N_I == 1) MUST N_I_p_c >= 2, if(N_I > 1) N_I_p_c can be 1	number of maximum iterations of step when solve equation for p and T
1	//N_I_T - if(N_I == 1) MUST N_I_T >= 2, if(N_I > 1) N_I_T can be 1	this is not in use now
0.05	//ht	time step

0	//x_b	begin coordinate of computational domain on OX
1	//x_e	end coordinate of computational domain on OX
0	//y_b	begin coordinate of computational domain on OY
1.5	//y_e	end coordinate of computational domain on OY
5	//kind_of_mesh == Li_mesh = 1, Par_mesh = 2, Li_mesh_flat = 3 li, Par_mesh_flat = 4, Polynom3_flat_mesh = 5	kind of mesh
0.1	//hxmin	minimal step on OX
0.1	//hxmax	maximal step on OX
10.3	//xfmin	coordinate of where step on OX is minimal front body
10.1	//xfmax	coordinate of to where step on OX is maximal front body
10.5	//xbmin	coordinate of where step on OX is minimal back body
10.7	//xbmax	coordinate of to where step on OX is maximal back body
10.4	//xmid	where is middle of body on OX
0.001	//hymin	minimal step on OY
0.1	//hymax	maximal step on OY
0.6	//ytmin	coordinate of where step on OY is minimal top body
0.8	//ytmax	coordinate of to where step on OY is maximal top body
0.4	//ybmin	coordinate of where step on OY is minimal bottom body
0.2	//ybmax	coordinate of to where step on OY is maximal bottom body
0.5	//ymid	where is middle of body on OY
1e-008	//MaxError_Velocities	maximum error for velocities
1e-009	//MaxError_p_c	maximum error for pressure and temperature
1e-009	//MaxError_T	maximum error for temperature
1000	//max_time_for_solve	Maximum time of calculation before stop the program in hours. When program finished here is written time of calculation, including reading input data, in hours.
0	//solve_is_finished	0 – the calculation is not reached to the steady state 1 – the calculation is reached to the steady state and the program is stopped (For information of condition look at the source)
500	//Nt_save_solved_data	Number of steps on time when solved data will be saved (velocities, pressure, temperature and density)
100	//Nt_save_DragCoefficient	Number of steps on time when drag coefficient (C_D) will be saved
100	//Nt_save_temp_data	Number of steps on time when temporal data will be saved
1	//N_I_check	Number of iteration when we check is errors in solving system are lower then given

0	//ToReadSolvedDataFromFile	0 – not read data from file (the program start to solve from beginning) 1 – to read solved data from file (program continue)
1	//ToReadSolvedDataFromBinaryFile	0 – read solved data from ASCII file (.txt) 1 – read solved data from binary file (.bin)
1	//ToWriteSolvedDataToBinaryFile	0 – write solved data from ASCII file (.txt) 1 – write solved data from binary file (.bin)
0	//ToContinueFromInterpolation	0 – not change readied data 1 – after read data execute procedure to apply velocities and temperature of the body in control volumes in the body. This vanish velocity slip and temperature jump values from arrays.
0	//ToImport_data_for_circles_from_file_b	0 – to not import data for circles 1 – to import data for circles
1	//ToImport_data_for_polyhedrons_from_file_b	0 – to not import data for polyhedrons 1 – to import data for polyhedrons
0	//ToImport_data_for_circles_from_file_p	not in use
0	//ToImport_data_for_polyhedrons_from_file_p	not in use
0	//ToImport_data_for_circles_from_file_V	not in use
0	//ToImport_data_for_polyhedrons_from_file_V	not in use
1	//Periodic_boundary_conditions_about_OX	0 – not periodic BC on OX 1 – periodic BC on OX
0	//Given_velocity_on_xb	0 – the velocity on inlet is not given 1 – the velocity on inlet is given Use with cautions!
0	//Given_velocity_on_xe	0 – the velocity on the outlet is not given 1 – the velocity on the outlet is given Use with cautions!
0	//Pressure_BC	0 – not Pressure BC 1 – Pressure BC
1	//ToStartFromExactSolutionForChannelFlowWithPressureBC	0 – to initialize matrixes with conditions for velocities, pressure and temperature 1 – to start from analytical solution for compressible, isothermal, gas in microchannal
0	//p_BC_xb_correction_method == 0 --> no correction for p_BC_xb; p_BC_xb_correction_method == 1 == p_BC_xb_correction_method_Vmax_xb --> V maximum inflow == 1, p_BC_xb_correction_method == 2 == p_BC_xb_correction_method_Vmean_xb --> V mean inflow == 1 0 //Pressure_ratio_correct == 0 --> not correct pressure ratio; Pressure_ratio_correct == 1 --> correct pressure ratio;	See in source
0	//GivenReKn -> u_given_xb = Re * Kn * sqrt(15.0 * M_PI / 128.0);	See in source

4	//correct_p_BC_xb	See in source
2.2147	//u_given_xb	See in source
0.0001	//u_given_xb_error	See in source
0	//p_correction_auto	See in source
0.001	//p_correction_min	See in source
2	//p_correction_max	See in source
0.05	//p_correction	See in source
1.5	//p_BC_xb	pressure in x_b (begin of channel)
1	//T_BC_xb	temperature in x_b (begin of channel)
1	//p_BC_xe	pressure in x_e (end of channel)
1	//T_BC_xe	temperature in x_e (end of channel)
1	//dudx_0_BC_xb	0 – not apply this BC 1 - $\left.\frac{\partial u}{\partial x}\right\|_{x=x\_b}=0$
0	//durhodx_0_BC_xb	0 – not apply this BC 1 - $\left.\frac{\partial(u.\rho)}{\partial x}\right\|_{x=x\_b}=0$
0	//drhodt_durhodx_dvrhody_0_BC_xb	0 – not apply this BC 1 – calculate u for first raw control volumes from on OX: $\left.\left(\frac{\partial(\rho)}{\partial t}+\frac{\partial(u.\rho)}{\partial x}+\frac{\partial(v.\rho)}{\partial y}\right)\right\|_{x=x\_b}=0$
0	//dudx_0_BC_xe	0 – not apply this BC 1 – calculate u for first raw control volumes from on OX: $\left.\frac{\partial u}{\partial x}\right\|_{x=x\_e}=0$
0	//durhodx_0_BC_xe	0 – not apply this BC 1 – calculate u for last raw control volumes from on OY: $\left.\frac{\partial(u.\rho)}{\partial x}\right\|_{x=x\_e}=0$
0	//drhodt_durhodx_dvrhody_0_BC_xe	0 – not apply this BC 1 – calculate u for last raw control volumes from on OY: $\left.\left(\frac{\partial(\rho)}{\partial t}+\frac{\partial(u.\rho)}{\partial x}+\frac{\partial(v.\rho)}{\partial y}\right)\right\|_{x=x\_e}=0$
0	//u_uMin_Mout_0_BC_xe	0 – not apply this BC 1 – not tested
0	//dudy_0_BC_yb	0 – not apply this BC 1 - $\left.\frac{\partial u}{\partial y}\right\|_{y=y\_b}=0$
0	//dudy_0_BC_ye	0 – not apply this BC 1 - $\left.\frac{\partial u}{\partial y}\right\|_{y=y\_e}=0$
0	//dvdx_0_BC_xb	0 – not apply this BC 1 - $\left.\frac{\partial v}{\partial x}\right\|_{x=x\_b}=0$

0	//dvdx_0_BC_xe	0 – not apply this BC 1 - $\left.\frac{\partial v}{\partial x}\right\|_{x=x\_e} = 0$
0	//dvdy_0_BC_yb	0 – not apply this BC 1 - $\left.\frac{\partial v}{\partial y}\right\|_{y=y\_b} = 0$
0	//dvdy_0_BC_ye	0 – not apply this BC 1 - $\left.\frac{\partial v}{\partial y}\right\|_{y=y\_e} = 0$
0	//dpdx_0_BC_xb	0 – not apply this BC 1 - $\left.\frac{\partial p}{\partial x}\right\|_{x=x\_b} = 0$
0	//dpdx_0_BC_xe	0 – not apply this BC 1 - $\left.\frac{\partial p}{\partial x}\right\|_{x=x\_e} = 0$
0	//dTdx_0_BC_xb	0 – not apply this BC 1 - $\left.\frac{\partial T}{\partial x}\right\|_{x=x\_b} = 0$
0	//dTdx_0_BC_xe	0 – not apply this BC 1 - $\left.\frac{\partial T}{\partial x}\right\|_{x=x\_e} = 0$
1	//To_Use_Kn_local_in_wall_BC	0 – use Kn number, when calculate velocity slip and temperature jump BC 1 – use local Kn number, when calculate velocity slip and temperature jump BC
1	//ToSolveContinuityEquation	0 – to not solve and write continuity equation to files 1 – to solve and write continuity equation to files
0	//use_interpolation_for_u_BC_by_time	0 – to not change inlet BC of u 1 – to change inlet BC of u
0	//t_BC_b	time to start change
2	//t_BC_e	time to end change
0.4	//u_BC_xb_t_BC_b	u inlet, when start change
0.5	//u_BC_xb_t_BC_e	u inlet, when finish change
1	//w_u	weight coefficient for u SOR ($1 \le w\_u \le 2$)
1	//w_v	weight coefficient for v SOR ($1 \le w\_v \le 2$)
1	//w_p	weight coefficient for pressure SOR ($1 \le w\_p \le 2$)
1	//w_Temper	weight coefficient for temperature SOR ($1 \le w\_Temper \le 2$)
2	//N_SubDomains_x	Number of subdomaind on OX, when use MPI
1	//N_SubDomains_y	Number of subdomaind on OY, when use MPI
0	//ToReadSolvedDataFromSeparateFileForEachSubDomain	0 – read input data for entire computational domain, like without MPI 1 – read input data from file for each subdomain
1	//ToWriteSolvedDataToFileForEntireComputationalDomain	0 – to not write solved (input) data for entire computational domain, like without MPI 1 – write solved (input) data for entire computational domain, like without MPI

0	//ToWriteSolvedDataToSeparateFileForEachSubDomain	0 – to not write solved (input) data in separate file for every subdomain 1 – write solved (input) data in separate file for every subdomain
0	//ToOnlyWriteDataFor_Nx_Ny_ForAllSubDomains_AndExit	0 – usual calculations 1 – if you want only to change the number of processes (subdomains), this will write information to file and will exit the program without any calculation.

The other type of files are files which contain information about solving bodies. This files contains geometry of body, temperature jump and velocity slip boundary conditions. The files are named like:

solve_polyhedrons_b.txt – this mean:
solve – the body will be solved
polyhedrons – that is file which contain information about polyhedron
b – mean body
.txt – mean that this is text file (ASCII code)

I will explain contain of file solve_polyhedrons_b.txt:

To decide which nodes from mesh are inside and which are outside polyhedron we need to give a coordinate of two points inside polyhedron. Two points must not be in the same line with any of the vertexes of polyhedron.

The coordinate for vertexes of polyhedron must be entered by clockwise or back of clockwise. The important is to be in one of two directions. The tests are made with clockwise direction.

1	//N_polyhedrons	number of polyhedrons
4	//N_polyhedron_vertex	number of vertex of polyhedron
-1.1 -0.1	//point1_inside_x point1_inside_y	coordinate of first point inside polyhedron (x, y)
15.01 -2.11	//point2_inside_x point2_inside_y	coordinate of second point inside polyhedron (x, y)
0 0	//u_polyhedron v_polyhedron	velocity of polyhedron
0	//pressure in polyhedron - must be 0	pressure in polyhedron
1	//Temperature in polyhedron	temperature in polyhedron
1	//is_VelocitySlipBC_body	0 – non-slip BC 1 – use velocity slip BC
1.1466	//F_VelocitySlip	Velocity slip coefficient of polyhedron
0.01	//w_VelocitySlipBC	not used variable
1	//is_TemperatureJumpBC_body	0 – no temperature jump BC 1 – temperature jump BC
2.1904	//F_TemperatureJump	temperature jump coefficient of polyhedron
0	//dTdn_on_wall_0	0 – not apply this BC 1 – $\left.\frac{\partial T}{\partial n}\right\|_{wall} = 0$ - temperature of the wall is equal to the temperature of the fluid next to the wall.
0	//ToSolveDragCoefficientForThisBody	0 – not solve C_D 1 – solve C_D and write data to file

-10.00001	0.100001	x and y coordinate of first vertex of polyhedron
1000.0001	0.100001	x and y coordinate of second vertex of polyhedron
1000.0001	-10.000001	x and y coordinate of third vertex of polyhedron
-10.00001	-10.000001	x and y coordinate of forth vertex of polyhedron

Output data for drag coefficient.

The output file is with name CD.0.txt. This name contain the following information:
CD – Drag Coefficient
0 – number of body for which are solved data. Numbers are 0, 1, 2 … n.
txt – this is text file (ASCII code)

The file contain following columns

ht	step by time in time when is solved drag coefficient
CD_fr_x	drag coefficient of friction on OX
CD_p_x	drag coefficient of pressure on OX
S_for_CD_bottom	area of bottom surfaces of body
S_for_CD_front	area of front surfaces of body
S_for_CD_top	area of top surfaces of body
S_for_CD_behind	area of behind surfaces of body
CD_fr_bottom	drag coefficient of friction on bottom surfaces
CD_p_bottom	drag coefficient of pressure on bottom surfaces
CD_fr_front	drag coefficient of friction on front surfaces
CD_p_front	drag coefficient of pressure on front surfaces
CD_fr_top	drag coefficient of friction on top surfaces
CD_p_top	drag coefficient of pressure on top surfaces
CD_fr_behind	drag coefficient of friction on behind surfaces
CD_p_behind	drag coefficient of pressure on behind surfaces

Output files for velocities, pressure, temperature and density.
The data is put in order:

$p(i, j) \leftrightarrow p(node), node = i + j * Nx;$

This arrange is very useful if we want to solve a part of all computational domain. In process of solving there is check for boundary conditions. This check is for volumes to the body surfaces. The problem is that the volumes to the body surfaces are in more cases around 4% of all volumes. This arrange will be very useful when we make parallelization of algorithm.