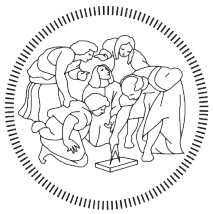


POLITECNICO
MILANO 1863

Computational Techniques for Thermochemical Propulsion

Report: Assignement 1

Incompressible fluid solver

Professor: Federico Piscaglia

Authors:
Borcea Dan - 10678208

Academic Year 2023/2024

Contents

List of Figures

1 Introduction

1.1 Physics

This report, along with the subsequent ones, aims to simulate the behavior of airflow within a combustion chamber in the presence of a flame stabilization mechanism. For this analysis, an incompressible flow solver was used, which is appropriate when the Mach number is below 0.3. As a result, the density variable in the transport equations is kept constant, and the energy equation is not solved, reducing the computational weight and time for the simulation.

The analysis is divided in three subcases:

- Steady state condition: a SIMPLE solver is used.
- Unsteady condition where maximum Courant number is set to 20;
- Unsteady condition where maximum Courant number is set to 1;

For both unsteady problems a PIMPLE solver is used, keeping the entire setup unchanged, while the maximum Courant number is modified. However, for the last case the employment of a PISO like solver would lead to a computationaly faster simulation, since the pressure-velocity coupling is not iterated several times in an outer loop, and this is done without affecting significantly its accuracy.

The flow is composed by air in gaseous phase only. Thus there is no need to solve for the transport of species and there is no need to add more complexity to the solution of the problem by means of multi-phase flow methods. Furthermore chemical reactions are not present.

1.2 Quantities of interest

The two main variables which need to be solved for are velocity and pressure, which need to be coupled by means of the momentum and pressure equations. Since the problem is dealing with an incompressible fluid the second unknown which is iterated inside the coupling loop is the ratio between the pressure and density, the latter being constant in space and time.

In order to exploit the momentum equation inside the algorithm the actual viscosity—defined as the sum of molecular viscosity and turbulent viscosity, `nut`—is required as input. To account for its effects, the Reynolds-Averaged Navier Stokes (RANS) equations are solved in conjunction with a turbulence model, specifically the k/ϵ model in this case. As a result, three additional variables are necessary:

- the turbulent kinetic energy `k`: it is proportional to the time averaged square of the velocity fluctuation vector;
- the kinetic energy dissipation rate ϵ: it is a way to estimate the energy dissipation due to the eddies at smaller scale.
- the turbulent viscosity `nut`, computed based on the two previous quantities.

2 Setup of the problem

2.1 Geometry and mesh

The domain for the simulation, made available for this study, was created by means of an arrangement of multiple neighboring regions generated with the blockMesh function. The result, which is shown in Fig.1, is a fully structured grid, composed only by hexahedra. This ensures great efficiency in solving the linear system of discretized equations thanks to the high diagonality of the matrix.

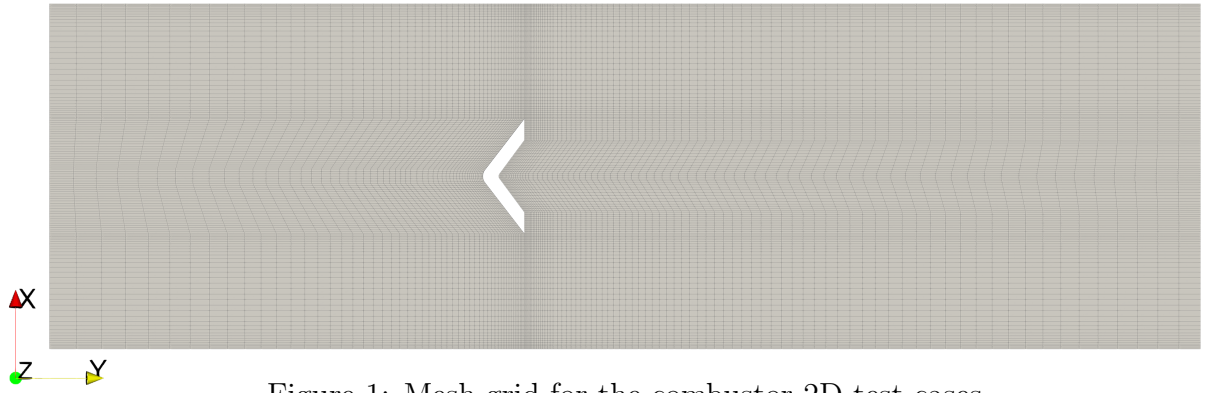


Figure 1: Mesh grid for the combustor 2D test cases

By using the blockMesh function it is also possible to refine the grid nearby the walls by means of the grading options in its dictionary. In this way velocity gradients near the walls can be well described in order to obtain more accurate viscous forces which are significantly affecting the solution.

Table 1: Mesh characterization

Parameter	Value	Validation
Number of cells	15677	-
Max aspect ratio	27.85	OK
Max non-orthogonality	39.64	OK
Mean non-orthogonality	16.60	OK
Max skewness	0.84	OK

The geometry is two-dimensional, meaning that the third direction is represented by one single cell and the flow is not affected by the presence of the walls orthogonal to it. By running the function checkMesh it is possible to visualize the characteristics of the grid, which are shown in table 1. As it can be seen there are no criticalities in the quality of the grid, ensuring good convergence and stability for the solution. Hence the majority of errors in CFD simulations have their source in the meshing process. Such results for the mesh quality could not be reached by using the snappyHexMesh function.

2.2 Boundary conditions and initial conditions

The airflow enters the domain through two patches: the inlet surface on the right side (`inlet`) and the two lateral back surfaces of the V-shaped flame stabilizer (`fuel_inlet`). These surfaces, along with the outlet on the left side of the mesh, are defined as patch type boundaries. All other surfaces are set as wall boundaries, except for the two surfaces normal to the Z-direction, which are specified as empty. The empty boundary type indicates that the simulation is two-dimensional, meaning the walls tangent to the flow in the Z-direction do not physically exist and do not interfere with the flow. Additionally, no flow can enter or exit the domain through these surfaces.

All these settings are specified inside the blockMeshDict file. The boundary conditions and intial conditions for each field need to be set inside the 0 folder of the directory. For each quantity the unit of measurement, the initial field and the boundary condition for the previously specified boundaries need to be set inside its specific file. All three test cases share the same blockMeshDict and 0 files for this study.

2.2.1 U field

The velocity is solved in m/s and its field at the initial timestep is uniform and equal to the null vector. For the two inlet patches the `fixed value` type boundary condition is set and the flows are uniform and determined by their relative vector, (0 10 0) for the `inlet` and (0 15 0) for the `fuel_inlet`. The same setting cannot be used also for the `outlet`, otherwise the mass

conservation would be overconstrained. Thus the `zeroGradient` condition is used. For all the non `empty` walls the `no slip` condition is employed, meaning that the value for the velocity at the interface between the wall and the external cell is always zero. For the velocity value at the cell center, which is not null, an interpolation is done.

2.2.2 p field

As already mentioned, to reduce the computational burden, the ratio between pressure and the constant density is solved directly inside the solver. Thus the unit of measurement that needs to be specified inside this file is m^2/s^2. Both initial field and `outlet` boundary condition are set uniform and equal to zero because the pressure quantity is solved with respect to a reference one while for both walls and inlets the `zeroGradient` condition is imposed.

2.2.3 nut, ϵ and k fields

Table 2: Initial conditions and BCs for turbulent quantities

-	units	initial field	inlets BC	walls BC	outlet BC
k	m^2/s^2	uniform 0.375	turbulentIntensity-KineticEnergyInlet	kqRWallFunction	zeroGradient
ϵ	m^2/s^3	uniform 2	turbulentMixingLength-DissipationRateInlet	epsilonWallFunction	zeroGradient
nut	m^2/s	uniform 0	calculated	nutkWallFunction	calculated

The settings for boundaries of the three fields are summerinzed in Tab. 2.

The inlet boundary conditions for k and ϵ are two useful methods to estimate their values at the inlet of the flow. Based on two empirical quantities which can be assumed, the mixing length for ϵ and the turbulence intensity for k, the two quantities can be computed: at the inlet ϵ is a function of k and the mixing length while by means of the turbulence intensity and the flow velocity an estimation for the fluctuation of the latter can be retrieved in order to compute the turbulent kinetic energy.

The three wall boundary conditions for k, ϵ and nut are a way to give a good estimation of the local values for the three quantities in vicinity of a wall based on other local quantities that are characterizing the flow.

2.3 Test case 1: steady

2.3.1 controlDict

The three test cases share multiple settings inside the controlDict which are relevant: the application foamRun, the solver incompressibleFluid and the functionObjects. The three function that are used for the post processing are: residuals, forcesIncompressible and yPlus. All of them are employed to satisfy part of the requirements of the assignment while other quantities are extracted in Paraview.

Since the first test case is a steady condition simulation, no progression in time is done during the simulation and the settings for time quantities are used only to distinguish the different iterations, to define their number and to keep the same structure of controlDict when passing from steady to unsteady problems and viceversa. For this test case 2000 iterations are done (if convergence is never reached) while the results are saved and logged every 100. Since the evolution of the flow between iterations is not relevant the purgeWrite setting is activated to keep only the last iteration in the results.

2.3.2 fvSolution

To perform a steady-state simulation the SIMPLE algorithm is employed. `nCorrectors` and `nOuterCorrectors` are not specified in the fvSolution file, meaning that the solver will consider them as unitary by default. However, the `residualControl` setting will dictate the convergence criteria for the inner loop: it is considered converged when the residuals at the first inner iteration drop below the specified thresholds. The employment of one single outer iteration is counterbalanced by the multiple iterations defined by the fictitious timestep.

Table 3: Mesh characterization

field	residual control	tolerance	relTol
U	1e-4	1e-5	0.1
p	1e-3	1e-6	0.1
pcorr	-	1e-6	0
k	-	1e-5	0.1
ϵ	-	1e-5	0.1

In Tab. 3 the residual control and the tolerances for the solving of each equation. When `relTol` setting is different from 0 the solving of the specific equation is stopped before the tolerance is satisfied. This technique, which is relevant especially during the first iterations, gives a faster convergence by ensuring the correct coupling between pressure and velocity.

The `relaxationFactors` values of 0.3, 0.7 and 0.5 for respectively pressure, velocity and turbulent quantities are typical choices for this type of problem and are shared by all test cases.

No `nNonOrthogonalCorrectors` were used, given the good quality of the mesh.

2.3.3 fvSchemes

In the `fvSchemes` file the discretization methods in time and space are specified for each quantity in the transport equations, including time derivatives, spatial gradients, divergence terms for convection and laplacian terms for diffusion. For space issues the various settings are not presented in the report.

2.4 Test cases 2 and 3: unsteady

As previously state the unsteady versions of the problem share all the settings with exception of the Courant number. Moreover their the setup is identicaly for many aspects to the one presented in the first test case. Thus, only the main modifications will be shown in the next sections.

2.4.1 controlDict

Since the problem is now evolving in time, the iterations and the physical time progression are equivalent. For this study a total period of 0.5 seconds is chosen. The timestep is stritlcy correlated to the Courant number, which can be up 20 and 1 respectively for the second and third case. An additional precaution to be done is to make sure that the intial timestep given by `deltaT` setting respects these two limits. The two selected values are 1e-4 and 5e-6 respectively.

To make sure the eventual oscillations and disturbances are well catched in the the post process a `writeInterval` of 0.002 s is set with an `adjustableRunTime` type of control.

2.4.2 fvSolution

Inside this file the most relevant modifications are done to convert the previous case into an unsteady problem.

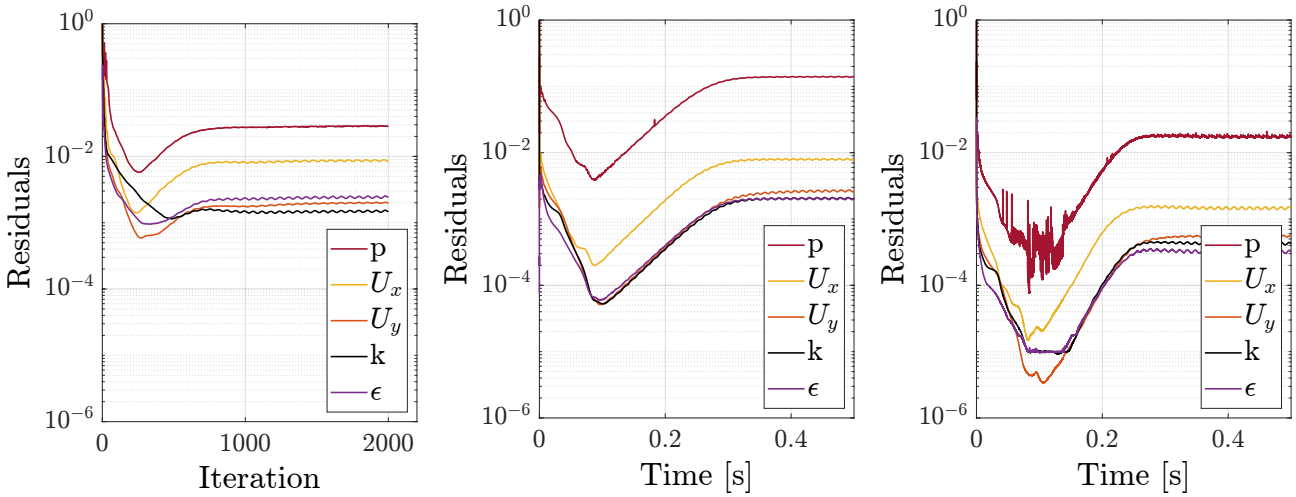
To the tolerance settings in the first test case, additional conditions are set to the final iteration of each linear system: the relTol setting is set to zero so that the linear system is iterated until the absolute residual tolerance is respected, just before exiting the loop.

The PIMPLE solver is imposed with 1 `nCorrectors`, 100 maximum `nOuterCorrectors` and a residual control of 1e-4 in the outer loop for all the fields. These are stricter conditions for convergence with respect to the SIMPLE case because having a temporal differential scheme implies an additional diagonal term in the linear system, thus a more diagonally dominant matrix and a faster convergence with respect to the steady state problem.

The turbOnFinalIterOnly, which forces the turbulence model to be solved only once and at the end of the outer loop, is disabled hence problems with highly turbulent nature are not well suited for this semplification. For the problem under examination the presence of Von Karman like aerodynamic effects are expected.

3 Results

3.1 Convergence



(a) SIMPLE (b) PIMPLE, Co<20 (c) PIMPLE, Co<1

Figure 2: Outer Loop initial residual evolution .

In Fig. 2 the residuals in each iteration are plotted for the three problems. They were obtained by means of the `residuals functionObject` in the `controlDict` file and are related to the first iteration in the first outer loop in the coupling mechanism. In the steady SIMPLE algorithm there is one single outer loop and the coupling between pressure is done by the multiple fictitious timesteps. Since the initial residual is never lower than the threshold imposed by the `residualControl` the simulation has not converged. The reason for this is that a steady state simulation is not appropriate for a system which is aerdynamically unstable and the solver is not able to converge to a unique solution.

In the second and third test cases the initial residuals are even larger with respect to the SIMPLE case but they have a different meaning. In these cases the initial residual is just a consequence of the integration in time: the faster the evolution of the flow is, the greater the initial residuals. Hence the lowest peaks for all five traces correspond to the dumping of the initial transient given by the initial conditions. The following increase of the residuals is strictly connected to the generation of the Von Karman eddies due to the instability of the flow. The residuals for the simulation with lower Courant number is characterized by lower residuals because the integration timestep is smaller and each fraction of time , which needs to be solved as a frozen steady condition, is more similar to the previous one, with respect to the higher Courant number simulation.

A better way to understand if the unsteady problems are solved correctly and if the pressure and velocity are properly coupled is to plot the initial residual at the last outer iteration, which must be lower than the `residualControl` by definition. To do this the necessary data were extracted from `log.foamRun` file and plotted for the second test case in Fig. 3

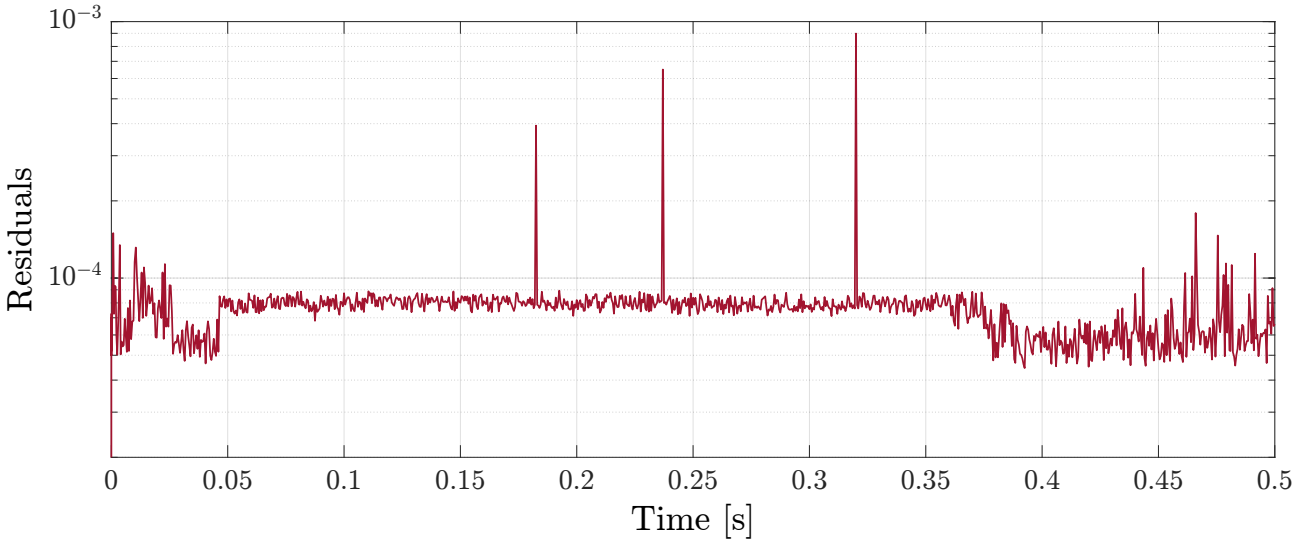


Figure 3: Initial residuals for the final iteration in the outer loop for the PIMPLE Co<20 case.

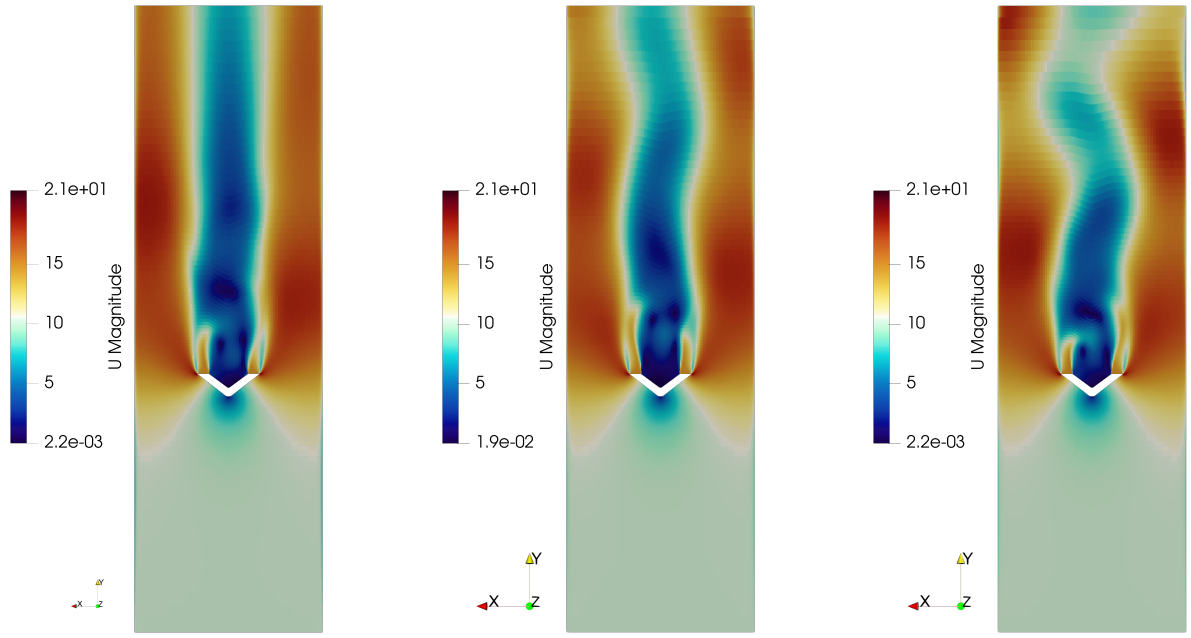
As it can be seen the initial residuals are lower than the required threshold for almost all the iterations. The timesteps which appear to be critical were checked in the log file and it was discovered that only three of them did not converge because they require more than 100 iterations, and represent the first timesteps of the simulation. The reason for which the rest of the badly uncoupled solutions are present is that the extracted data from the log are related to the additional iteration made by the solver as soon as the `residualControl` gets satisfied. For reasons that are still unknown the initial residual for this iteration results to be higher then the previous value. One hypothesis to justify this could be an incorrect setup of the under-relaxation factor for the final iteration.

3.2 Post processing

The contour for the velocity field is shown in Fig. 4, plotted for the last iteration in each case. It can be noted that the aerodynamic instability is more evident in the PIMPLE solution with lower Courant number. The reason for this is that with a smaller temporal step it is easyer to catch the physics of unsteady turbulent phenomena, while solvers with high Courant number tend to dampen this effect.

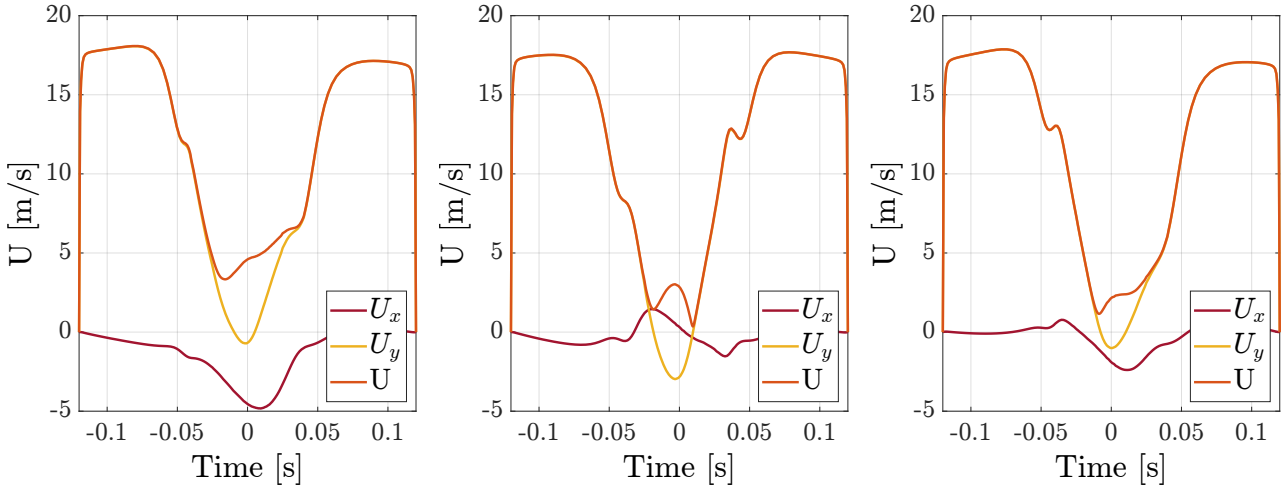
Fig. 5 presents the velocity profile along the line $y = 0.1$, illustrating both the total magnitude and its decomposition into the two dimensions. The comand was done in Paraview and its data were extracted in a .csv file.

The post processing for y^+ and the drag force was prepared before the simulation inside the `controlDict file` by using the `functionObjects` and it is shown respectively in Tabs. 2 and 5, at the final iteration. The total value for the latter quantity can be retrieved by summing up the contributions of pressure and viscous effects.



(a) SIMPLE (b) PIMPLE, Co<20 (c) PIMPLE, Co<1

Figure 4: Velocity field for SIMPLE case at iteration 2000 and for PIMPLE cases at time 0.5s.



(a) SIMPLE (b) PIMPLE, Co<20 (c) PIMPLE, Co<1

Figure 5: Velocity profile along y=0.1 line for SIMPLE and PIMPLE cases, last iteration

-	min y^+	max y^+
splitter	33.52	57.38
back	11.51	19.05
sides	16.96	29.26

(a) SIMPLE

-	min y^+	max y^+
splitter	33.50	57.47
back	10.29	13.86
sides	9.70	28.40

(b) PIMPLE, Co<20

-	min y^+	max y^+
splitter	33.46	57.82
back	11.52	18.10
sides	9.71	30.15

(c) PIMPLE, Co<1

Table 4: Minimum and maximum values of y^+ at wall patches (last iteration).

SIMPLE		PIMPLE, Co<20		PIMPLE, Co<1	
pressure	viscous	pressure	viscous	pressure	viscous
3.180e-02	-2.3e-04	3.068e-02	-1.4e-04	3.187e-02	-1.3e-04
total = 3.157e-02		total = 3.054e-02		total = 3.174e-02	

Table 5: Total force along y direction.