Computational Fluid Dynamics Lab

Project Group 3

Uncertainty Quantification and 2D Navier-Stokes Equations with Arbitrary Geometries

Authors:
Grant Bartel
Ionut Farcas
Ayman Saleem

This project incorporates the *2D Navier-Stokes* Solver with Arbitrary Geometries along with *non-intrusive Uncertainty Quantification* (UQ) methods.

Moreover, for the Navier-Stokes solver we developed a *serial* solver as well as a *parallel solver* using *domain decomposition with MPI*. The UQ part was parallelized using MPI, thus offering the option to use *two levels of parallelism*.

The first part is the 2D Navier-Stokes Solver with Arbitrary Geometries. The chosen scenario is *flow over a step*, which was used for Worksheet 3. The solver is modified and it returns the re-attachment point–the **Quantity of Interest** (QoI)–and the time of occurrence–steady state time. This is determined by checking for the point where the direction of the horizontal-velocity-component reverses in the fluid layer above the floor. We also incorporated the option to input the dimensions of the .pbm file used for encoding the arbitrary geometries at runtime. Its purpose is to return the re-attachement point and the time when it occurs depending on the input *Reynolds number* or *viscosity*, and it serves as the numerical solver for the *Uncertainty Quantification* part. The output of the solver is a *.mc data file*, which contains the Reynolds number or the viscosity, the re-attachement point, and the time of occurrence. Each datafile is unique and is based on a unique id (*see 3 in the enumeration below*). The solver was implemented using the C programming language with MPI.

The input parameters of the solver are [1]:

1. a **flag** with which the user decides whether the input variable is the Reynolds number or the viscosity, as follows:
$$\begin{cases} \text{if flag} = 1, \text{ then } \textit{Reynolds number} \\ \text{else}, \textit{viscosity} \end{cases}$$
2. the **Reynolds number** or the **viscosity**, depending on the above flag;
3. a **unique ID** for each input variable, that will be used for generating unique output data files;
4. the **x-dimension** for the .pbm file;
5. the **y-dimension** for the .pbm file;

Thus, the **first task** is to **compile** the **solver code** so that to generate the executable (i.e., *sim*) that is used in the UQ part.

[1] every input parameter should be greater or equal to 0.

After the executable for the solver is compiled we generate the UQ program. The UQ program calls the solver. In the UQ part, we implemented the **Monte Carlo** sampling (MCS) method that is used as an uncertainty propagation method in the case when the *Reynolds number* or the *viscosity* contains uncertainties. This is modelled as a **continuous random variable**, where the user can choose between the textbfnormal or **uniform** [2] probability distribution functions (PDF) as options to generate the random variables. The *Mersenne twister* pseudorandom number generator (PRNG) was chosen to actually generate these variables, based on the selected PDF. For each generated sample, the Navier-Stokes solver will be called via a **system call** using the above mentioned input parameters and after the solutions are obtained, the statistics (i.e., the **mean** and **variance**) are computed.

The input parameters of the UQ program are [3]:

1. a **flag** with which the user decides which UQ method to use [4], as follows.:
$$\begin{cases} \text{if flag} = 1, \text{ then } MCS \\ \text{else}, another\ method \end{cases}$$

2. a **flag** with which the user decides whether the input variable is the Reynolds number or the viscosity, as follows.:
$$\begin{cases} \text{if flag} = 1, \text{ then } Reynolds\ number \\ \text{else}, viscosity \end{cases}$$

3. a **flag** with which the user decides whether the used distribution is normal or uniform:
$$\begin{cases} \text{if flag} = 1, \text{ then } normal \\ \text{else}, uniform \end{cases}$$

4. a **flag** with which the user decides the solver type: serial or parallel:
$$\begin{cases} \text{if flag} = 1, \text{ then } sequential\ solver \\ \text{else}, parallel\ solver \end{cases}$$

5. the **number of samples** used for the MC simulation;

6. the **mean** for the random variable;

7. the **standard deviation** for the random variable;

8. the **x-dimension** for the .pbm file;

[2] the random variables are taken of the form m + s*ζ, where m is the *mean*, s the *standard deviation* and $\zeta \sim N(0,1)$ *or* $\sim U(0,1)$;

[3] every input parameter should be greater or equal to 0,as in the solver case

[4] this is for future development; for the moment, we have only MCS;

9. the **y-dimension** for the .pbm file;

Thus, the **second task** is to **compile** the **UQ code** so that to generate the executable (i.e., *sim _UQ*) that will be used to run the entire simulation. The UQ part was implemented using the C++ programming language plus MPI.

To plot the distribution of the separation point the MATLAB script `combine_data.m` (in the folder *final_samples/*) should be run. The *.mc* files must be placed inside a separate folder within *final_samples/*).

It is important to mention that for the random number generating, we use the *Boost* library, which is included in the submitted file. Moreover, the application code is available at `https://github.com/grantathon/computational_fluid_dynamics/tree/master/project`.