16.90: Project #2
Simulation of the Inviscid Supersonic Flow over a Cylinder
Due Date: April 20, 2016 at 2:30pm

1 Background

Inviscid supersonic flow is governed by the Euler equations. The Euler equations have been described in detail in the lecture notes on finite volume methods. In this project, we will numerically solve the Euler equations using a finite volume method to approximate the inviscid, supersonic flow around a cylinder at $M_\infty = 2$. The specific geometry is shown in Figure 1. The distance to the outer computational boundary

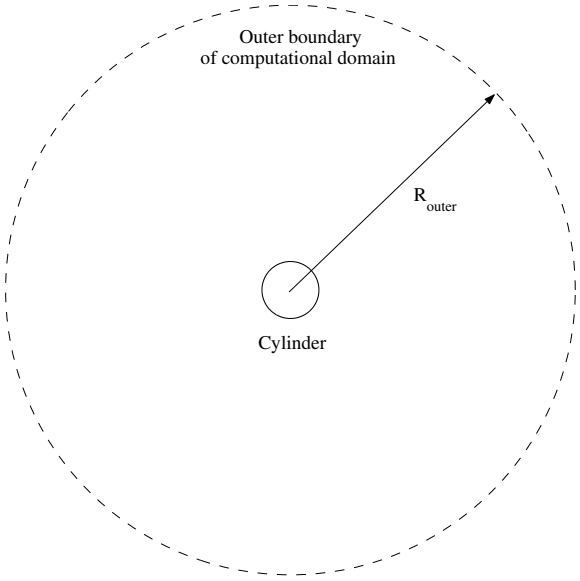

Figure 1: Computational domain

is set to 10 times the cylinder radius. All distances in the problem have been non-dimensionalized by the radius of the cylinder, thus, $R_{cyl} = 1$ and $R_{outer} = 10$.

The basic requirements for the project are to implement a finite volume method for the 2-D Euler equations, develop a grid adaptive method to improve the solution accuracy, and to apply these algorithms to solve the $M_\infty = 2$ flow around a cylinder.

The remainder of this section concentrates on technical issues in developing the adaptive finite volume algorithm.

1.1 Farfield Boundary Conditions

The easiest method to implement boundary conditions at the outer (i.e. farfield) boundary is through the flux function (i.e. **eulerflux.m**) using the state vector from the interior triangle as one of the inputted state vectors and the freestream state vector as the other inputted state vector. Note: the freestream state vector is already calculated in the **FVM.m** script and is called **Uinf**.

1.2 Local Timestepping and CFL Definition

Local timestepping is the idea of using a different timestep in each control volume to accelerate the convergence of the solution to the steady state. This means that the iterations are not accurate in time, but since the unsteady evolution of the flow is not of interest when solving a steady flow, the lack of time accuracy is not a problem. If a single value of Δt were used in each cell, then the value of Δt would need to be set so

that the calculation is stable for all cells. When the grid contains very small and large cells, the timestep will generally be constrained by the smaller cells. As a result, the larger cells are being evolved at a potentially much smaller Δt than the stability limit would require for them.

To implement local timestepping, a vector of timesteps is calculated, one timestep for each cell. Specifically, define the CFL number for a cell to be,

$$\text{CFL} = \frac{\Delta t}{2A} \sum_{i=1}^{3} s_{\max i} h_i, \tag{1}$$

where A is the area of the cell, the summation is over the three faces of a cell, $s_{\max i}$ is the maximum propagation speed for the edge, and h_i is the length of the edge. Note that for cell j, a forward Euler integration of the finite-volume discretization of the conservation law is,

$$U_j^{n+1} = U_j^n - \left(\frac{\Delta t}{A}\right)_j R_j^n.$$

Since this integration requires the value of $\Delta t/A$ for the cell, the CFL number can be used to calculate this by re-arranging Equation 1,

$$\frac{\Delta t}{A} = \frac{2\text{CFL}}{\sum_{i=1}^{3} s_{\max i} h_i}.$$

The easiest method to calculate $\Delta t/A$ is to calculate the summation of $s_{\max i} h_i$ during the flux calculations. Note that $s_{\max i}$ is calculated by **eulerflux.m** and **wallflux.m**.

For stability, the algorithm requires that CFL < 1 in general. It is recommended that CFL $= 0.9$ is used.

1.3 Adaptation and Adaptation Metric

As part of this project, you are to increase the accuracy of the simulation through the use of mesh adaptation. Mesh adaptation locally increases the mesh resolution in regions where errors are likely to be large. The question is how to quantify which cells have large errors. Furthermore, given some quantification of the error, an algorithm is required to determine which cells should be refined.

The following description is an example of an adaptive algorithm which will work reasonably well for the given problem. You may simply implement this algorithm in your project, however, you could also create your own adaptive algorithm.

Often, errors are largest when the solution has large jumps from cell-to-cell. One form of error indicator for a given cell which quantifies the cell-to-cell jumps is defined as,

$$\text{Error Indicator}_0 = \sum_{i=1}^{3} |M_i - M_0|\, h_i,$$

where M_0 is the Mach number in the cell, M_i is the Mach number in the cell across of edge i, and h_i is the edge length of edge i. On the farfield boundaries, a reasonable approach is not to include any contribution to the error indicator. At a solid wall boundary, the contribution for the edge could be defined as,

$$\text{Contribution to Error Indicator on a solid wall} = |M_{n0}| h_i,$$

where M_{n0} is the Mach number of the normal velocity component (to the edge), and h_i is the edge length of the boundary edge. This is a measure of error because the normal Mach number should be zero on a solid wall.

The last decision which must be made is how to choose which cells to adapt. A simple approach is to adapt a fixed percentage of the cells, η. This can be done by sorting the cells according to the error indicator and choosing the top η percent. A reasonable value of η is around 25%.

2 Comments on Matlab Files

Several Matlab-related files are available on the webpage in a zip file. The following is a brief description of these files:

- **FVM.m**: This is the main script which you should modify to implement the finite volume method. Comments inside the script indicate where additions need to be made. The script already plots the solution, the residual, and the drag coefficient during the iterations. A restart capability exists so that the additional iterations can be made from a solution which is currently in memory. This is useful to further converge a solution on a current mesh, as well as to converge a solution just after adaptation. The following variables control the restart capability:
  - **FVMrestart**: if set to 1, then **FVM.m** will continue to iterate from the solution in memory. Otherwise, the grid and solution will be re-initialized.
  - **ntol**: The maximum number of iterations to run before terminating. Note: if a restart occurs the number of iterations will start at the previous iteration. Thus, for additional iterations to occur, **ntol** must be set to greater than the current number of iterations.
  - **Rtol**: The tolerance for the maximum residual (in $\log_{10}$). The iterations will terminate if the maximum residual is less than this value.
- **cyl_initmesh.m**: This script creates an initial mesh and the corresponding mesh data structures for the cylinder problem. You should not need to modify it. This script calls a mesh generation routine in the Matlab PDE Toolbox (called **initmesh**). You will need to use Matlab on a computer which has this toolbox installed.
- **cylgeom.mat**: This is a Matlab data file which describes the computational domain and is loaded by **cyl_initmesh.m** and **cyl_adaptmesh.m**. You should not need to modify it. It needs to be in the same directory (i.e. folder) that the other scripts are stored.
- **SetupMesh.m**: This Matlab script generates the mesh data structures used in **FVM.m**. You should not need to modify it. It is called by **cyl_initmesh.m** and **cyl_adaptmesh.m**.
- **SetupEdgeList.m**: This Matlab script generates the edge data structures used in **FVM.m**. You should not need to modify it. It is called by **SetupMesh.m**.
- **eulerflux.m**: This function calculates the two-dimensional upwinded flux given the left and right state vectors, a unit normal vector pointing from the right triangle to the left triangle, and the ratio of specific heat. It returns the flux entering the left triangle and the maximum magnitude propagation speed on the edge (this propagation speed is used in calculating the timestep). Note: the flux entering the right triangle is just the opposite of the flux entering the left triangle. You should not need to modify this script, but you will need to call it in your modifications of **FVM.m**. See the top of the script for more information on how to call this function.
- **wallflux.m**: This function calculates the two-dimensional flux at a wall boundary edge given the interior state vector, the unit normal vector pointing into the computational domain, and the ratio of specific heat. It returns the flux entering the computation domain and the maximum magnitude propagation speed on the edge (this propagation speed is used in calculating the timestep). You should not need to modify this script, but you will need to call it in your modifications of **FVM.m**. See the top of the script for more information on how to call this function.
- **CalcForces.m**: This script calculates the lift and drag coefficients acting on the cylinder. You should not need to modify it. It is called by **FVM.m**.
- **cyl_adaptmesh.m**: This script adapts an existing mesh given a list of triangles to refine. The list of triangles must be in the vector, **RefineList**. The entries of this vector are the integer indices of the triangle to be refined. This script generates the adapted mesh and transfers the solution from the original mesh to the adapted mesh. After this script is called, the new mesh and data structures, and

the transfered solution reside in the same variable names as the original mesh. Thus, **FVM.m** can be directly applied to iterate the solution on the adapted mesh using the restart capability. You should not need to modify this script.

3 Project Requirements

3.1 Finite Volume Method for the 2-D Euler Equations

The first task is to develop a finite volume method for the flow around a cylinder. Apply this method to solve the $M_\infty = 2$ flow around the cylinder on the initial grid (i.e. without adaptation). For this required task, turn in a plot of the Mach number distribution. Also, turn in plots of the convergence of the maximum residual and the drag coefficient (versus iteration). Once your finite volume method is working, you should converge your solution so that the drag coefficient is converged to approximately the second decimal place, i.e. X.YZ.

3.2 A Grid Adaptive Method

Develop a method to identify which cells to adapt. Include a description of the method you developed. Apply this method to the $M_\infty = 2$ flow around the cylinder. Include plots of the adapted grids, the corresponding Mach number distributions, and the convergence histories of maximum residual and the drag coefficient. Is the drag coefficient converging as the mesh is adapted? What's your best estimate for the drag coefficient for this problem?

3.3 Report

Your report on the above requirements must be submitted to the Stellar website in a single PDF document by the project due date given above. Your report only needs to document these requirements.

3.4 Commented Source Code

A thoroughly commented MATLAB source code (or codes if you broke the solver into more than one module) must be turned in with the project. You will be graded on the clarity by which your comments describe what you are doing in the code. Your comments should be descriptive enough for a person with some exposure to fluid dynamics and Finite Volume Methods to clearly understand everything you are doing in the code. Turn in any codes which you have modified onto the Stellar site. It would be easiest if the files were included in a single archive file. The files can be uploaded until the project due date given above.