Project 3: Flow over a Three-Element Airfoil

AE 523, Computational Fluid Dynamics, Fall 2016

Due: November 30, 11:55pm, electronically via Canvas

Problem Description

In this project you will apply the first and second-order finite volume method to solve for compressible, subsonic flow over a three-element, high-lift airfoil. Postprocessing calculations will include calculating the force, plotting field contours, and drawing streamlines.

Geometry: Figure 1 shows the geometry of the airfoil, which consists of three elements: the leading-edge *slat*, the center *main* portion, and the trailing-edge *flap*. In convenient CFD units, the chord length is $c = 0.5588$.

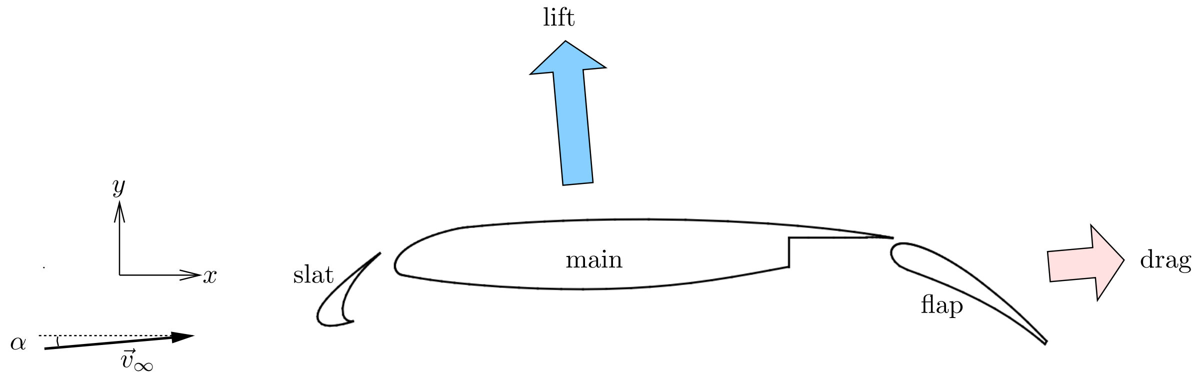


Figure 1: Geometry and output definitions for a three-element airfoil.

Units: To avoid ill-conditioning, use "convenient" $\mathcal{O}(1)$ units for this problem, in which the freestream state is

$$\mathbf{u}_\infty = [\rho, \rho u, \rho v, \rho E]^T = \left[1, M_\infty \cos\alpha, M_\infty \sin\alpha, \frac{1}{(\gamma-1)\gamma} + \frac{M_\infty^2}{2}\right]^T, \tag{1}$$

where $M_\infty = 0.25$, $\gamma = 1.4$, $\alpha = 5°$.

Boundary Conditions: The computational domain consists of the region outside the airfoil geometry and extends at least 90 chord lengths away from the airfoil. On the far-field boundary away from the airfoil, assume that the state is free-stream, $\mathbf{u} = \mathbf{u}_\infty$ as given in Equation 1. As such, impose a full-state boundary condition on the far-field. The slat, main, and flap are all impenetrable walls. Since you are solving the Euler equations, impose the inviscid-wall boundary condition on these surfaces.

Outputs: The pressure on the airfoil surface exerts a force on the airfoil. With three elements of the airfoil, we compute three separate forces. For example, the (2D) force on the slat is

$$\vec{F}'_{\text{slat}} = \int_{\text{slat}} p\vec{n}\, dl, \tag{2}$$

where p is the surface pressure and $\vec{n}$ is the normal that points out of the flow. Similar expressions hold for the other two airfoil elements, and summed together, these give the total force on the airfoil,

$$\vec{F}'_{\text{airfoil}} = \vec{F}'_{\text{slat}} + \vec{F}'_{\text{main}} + \vec{F}'_{\text{flap}}. \tag{3}$$

The drag, D', is the component of this force aligned with the free-stream velocity, $\vec{v}_\infty$, and the lift, L', is the component perpendicular to $\vec{v}_\infty$, with a positive projection on $\hat{y}$. The lift, drag, and pressure coefficients are defined respectively as

$$c_\ell = \frac{L'}{\frac{1}{2}\rho_\infty |\vec{v}_\infty|^2 c}, \qquad c_d = \frac{D'}{\frac{1}{2}\rho_\infty |\vec{v}_\infty|^2 c}, \qquad c_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty |\vec{v}_\infty|^2}, \tag{4}$$

where recall that c is the main-element chord.

Numerical Method

Discretization: You will use both first and second-order finite volume methods to solve for the flow over the three-element airfoil. For both methods use local time stepping to march the solution to steady-state. The initial condition for the first-order solver should be the free-stream state, $\mathbf{u}_\infty$. The initial condition for the second-order solver should be the converged first-order solution. Assess convergence by monitoring the L_∞ norm of the residual vector, $|\mathbf{R}|_{L_\infty}$, and deem a solution converged when $|\mathbf{R}|_{L_\infty} < 10^{-7}$. Use the Roe flux (code provided) for the interface flux and to impose the full-state far-field boundary condition.

Time Stepping: Use forward Euler for the first-order method. With local time-stepping, the update on cell i at iteration n can be written as

$$\mathbf{u}_i^{n+1} \quad = \quad \mathbf{u}_i^n - \frac{\Delta t_i^n}{A_i}\mathbf{R}_i(\mathbf{U}^n), \tag{5}$$

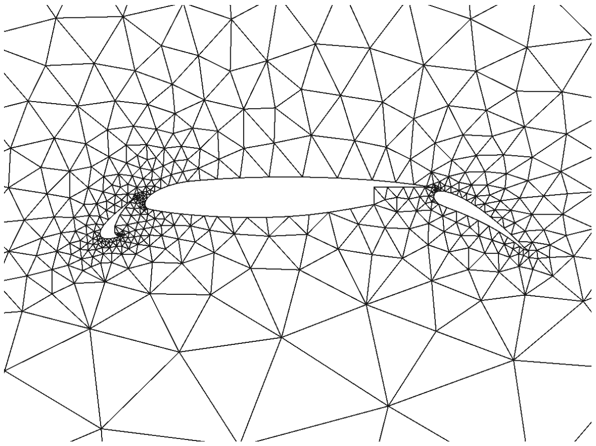
where Δt_i^n is the local time step computed from the state $\mathbf{U}^n$. For the second-order method, use the following TVD RK2 scheme,

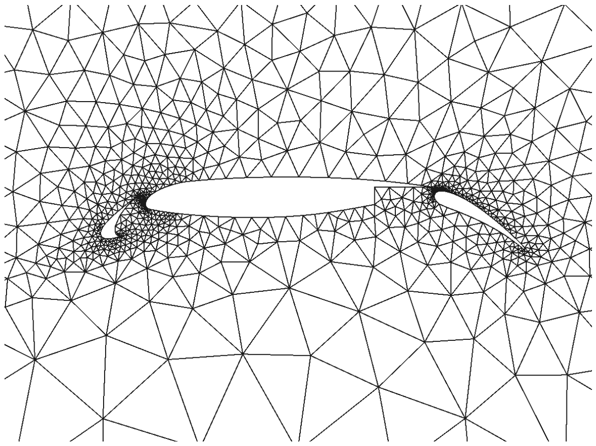
$$\mathbf{u}_i^{FE} \quad = \quad \mathbf{u}_i^n - \frac{\Delta t_i^n}{A_i}\mathbf{R}_i(\mathbf{U}^n) \tag{6}$$

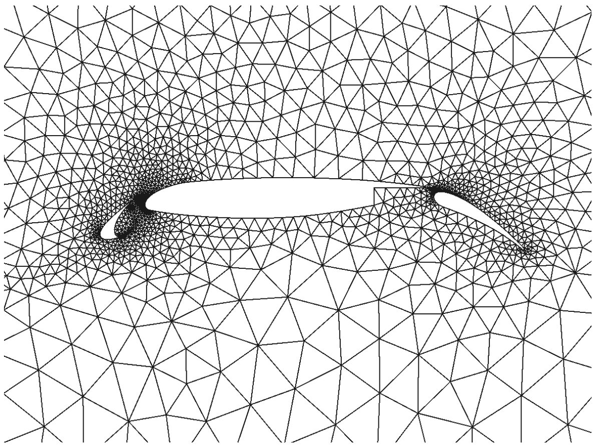
$$\mathbf{u}_i^{n+1} \quad = \quad \frac{1}{2}\left[\mathbf{u}_i^n + \mathbf{u}_i^{FE} - \frac{\Delta t_i^n}{A_i}\mathbf{R}_i(\mathbf{U}^{FE})\right]. \tag{7}$$

Note that the same local time step is used in both stages.

Meshes: Your code must run on the provided unstructured triangular meshes, illustrated in Figure 2. Four mesh resolutions are given, though you do not need to run on all of them for full credit (see Tasks). Each of these meshes is provided in the form of several text files, as described in the Provided Files section. Note these meshes do not constitute a uniform-refinement sequence. Instead, the meshes were generated adaptively by targeting flow regions important for calculating the force on the airfoil.


(a) Mesh 0: 1149 elements


(b) Mesh 1: 2116 elements


(c) Mesh 2: 4124 elements

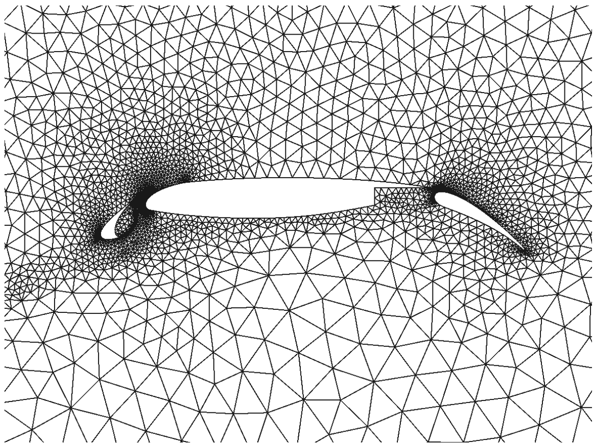

(d) Mesh 3: 8031 elements

Figure 2: Four provided computational meshes.

Output Calculation: The force calculation in Equations 2 and 3 requires an integral of the surface pressure times the normal vector. This is precisely the same quantity that is computed in enforcing the wall boundary condition, in the momentum portion of the boundary flux. Therefore, to calculate the force, you can reuse the inviscid wall flux calculation. Similarly, calculate the pressure coefficient using the momentum components of the wall flux to back out the pressure.

Field Visualization: When plotting field quantities near the airfoil, use the coordinate limits $[x_{\min}, x_{\max}, y_{\min}, y_{\max}] = [-0.3, 0.9, -0.35, 0.25]$. For a meaningful visual comparison, use the same field quantity range among all plots. Use piecewise-constant (flat) shading for the first-order results, and filled contours for the second-order results. To obtain smooth contours in the second-order cases, compute states at the mesh nodes by averaging the cell averages from the surrounding cells.

Streamline Plotting: One of the tasks asks for a plot of the streamlines. Generate this by plotting contours of the generalized stream function, ψ, which satisfies

$$\nabla \times (\psi \hat{z}) = \rho \vec{v}. \tag{8}$$

To compute ψ, use the property that the difference of ψ between two points measures the mass flow rate between those points. Referring to Figure 3, if ψ_1 is known, ψ_2 can be computed as

$$\psi_2 = \psi_1 + \int_1^2 \rho \vec{v} \cdot \vec{n}\, dl = \psi_1 + \Delta l\, \hat{F}_\rho, \tag{9}$$

where Δl is the edge length and $\hat{F}_\rho$ is the mass-conservation component of the numerical flux.

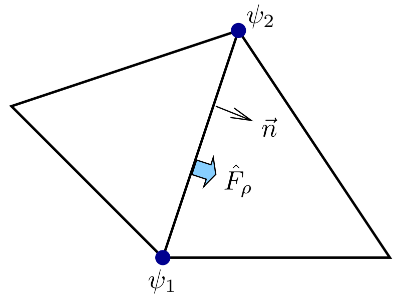


Figure 3: Definitions in the incremental calculation of the stream function, ψ.

Start the calculation by setting ψ to zero on one of the walls, e.g. the main element of the airfoil. Then loop over all interior and boundary edges: on each edge, check the values of ψ on the two nodes; if one of these is known and the other is not, then use Equation 9 to compute the unknown value of ψ. Repeat this loop over all edges multiple times, until ψ is known at all of the nodes. Note, these iterations use the mass flux on each edge, which can be pre-computed once before the iterations and stored for each interior and boundary edge. Finally, note that this calculation of ψ only works for converged flow solutions, when using the same numerical flux as in the residual calculation: otherwise, if there are any nonzero mass residuals, the node values of ψ will not be uniquely defined.

Provided Files

1. The Roe flux function, in both Python (`flux.py`) and Matlab (`flux.m`) formats.
2. Mesh files. For each mesh, these include:
   - `V.txt`: x, y coordinates of all nodes
   - `E.txt`: list of all triangular elements, as sequences of 3 nodes
   - `IE.txt`: interior faces and connectivities
   - `BE.txt`: boundary faces and connectivities

   More format information is found in each file, and the appendix shows a small example.
3. Plotting hint: helpful packages include `pdeplot` in the PDE toolbox in Matlab, and `tricontourf, tripcolor` in Python's matplotlib.pyplot.

Tasks

1. [**20%**] Implement a first-order finite volume method to perform a flow solve on a given mesh. The code should:
   - read and process the mesh files,
   - iterate until the residual (L_∞ norm) is less than the tolerance of 10^{-7},
   - calculate the lift coefficient and contributions from the slat, main, and flap,
   - monitor and log the residual norm and output convergence every 10 iterations.
2. [**15%**] Run your code on at least the first two meshes, index 0 and 1 [1]. For each mesh:
   (a) Plot the residual and c_ℓ output convergence with iterations.
   (b) Give the converged total c_ℓ and c_d values, and the slat, main, flap breakdown.
   (c) Show converged contours of the Mach number and pressure for all meshes run.
   (d) Plot the surface pressure coefficient, $c_p(x)$ over the airfoil (all elements).
3. [**10%**] Write a code that plots streamlines, and show streamlines for all cases run. For the contours, use 50 levels of ψ between -0.1 and 0.1 in the given units. Also, calculate the flow rate in the slat/main and main/flap gaps for all cases run.
4. [**45%**] Implement a second-order version of the finite-volume method and repeat the previous parts (same point allocation). Note:
   - Use the first-order solution as a starting guess on each mesh
   - Plot states with linear variation in each cell
   - Use the correct second-order fluxes in the streamline calculation

[1] You will receive bonus points for running on meshes 2 and 3, [**+2.5%**] for each mesh and order (first, second), up to a total of 10 possible bonus points.

Deliverables

You should turn in **two files** electronically via Canvas:

1. A technical report as a `.pdf` file that describes your methods and results, and that addresses all of the above tasks. The report should be professional, complete, and concise. **10%** of your grade will be determined by the professionalism (neatness, labeling of axes, spelling, etc.) of your report.
2. Documented source files of all codes you wrote for this project, as one `.zip` archive.

This is an individual assignment. You can discuss the project at a high level with each other, but you must turn in your own work.

Appendix: Sample Mesh and Connectivity

Figure 4 defines key terms and conventions in the description of an unstructured triangular mesh. Note that: (1) the three nodes that define a cell (element) are ordered counterclockwise; (2) on an interior edge defined by two nodes (n_1, n_2), the left element is the one that is on the left as we traverse the edge from n_1 to n_2; and (3) local edge i in an element is opposite local node i.

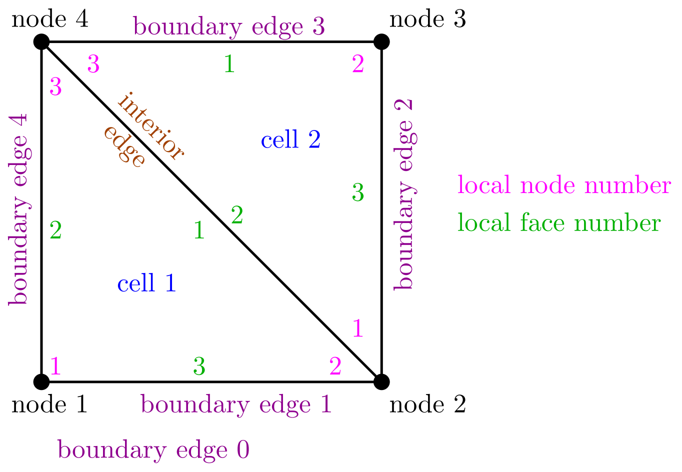


Figure 4: Sample mesh with definitions of various quantities.

The corresponding mesh files for this mesh would look as follows:

Listing 1: V.txt

```
# x, y coordinate of each node
0.000000000000000000e+00 0.000000000000000000e+00
1.000000000000000000e+00 0.000000000000000000e+00
1.000000000000000000e+00 1.000000000000000000e+00
0.000000000000000000e+00 1.000000000000000000e+00
```

Listing 2: E.txt

```
# n1, n2, n3 (1−based) node numbers of each element
1 2 4
2 3 4
```

Listing 3: IE.txt

```
# n1, n2, eL, eR, fL, fR for each interior edge
# n1, n2 = two nodes that define the edge
# eL, eR = element on left/right as we walk from n1 to n2
# fL, fR = local edge numbers of this edge in each element
#             note: local edge f is opposite local node f
4 2 2 1 2 1
```

Listing 4: BE.txt (index designations are for the airfoil case)

```
# n1, n2, eL, index, fL for each boundary edge
# n1, n2 = two nodes that define the edge
# eL = element next to the boundary edge
# index = boundary group index: farfield(1), main(2), slat (3), flap (4)
# fL = local edge number in eL
#       note: local edge f is opposite local node f
1 2 1 1 3
2 3 2 2 3
3 4 2 3 1
4 1 1 4 2
```