16.901: Finite Element Method Project
Simulation of Flow over an Airfoil in a Wind Tunnel
Due Date: April 22, 2pm

1 Background

The flow, and in particular the forces, on an airfoil are often estimated with the help of wind tunnel testing. Unfortunately, the walls of the wind tunnel have an impact on the aerodynamics and result in a different flow over the airfoil. Some of this impact can be estimated through an inviscid, potential flow calculation. In this project, we will develop a finite element method to model the inviscid, incompressible flow over an airfoil in the test section of a wind tunnel.

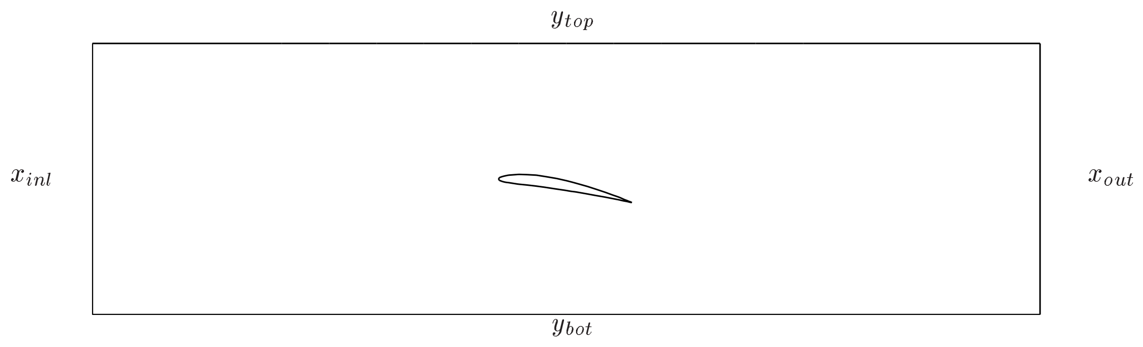

Figure 1: Wind tunnel/airfoil computational domain

Parameter	Definition	Value
x_{inl}/c	Inlet location	-3.0
x_{out}/c	Outlet location	4.0
y_{bot}/c	Bottom wall location	-1.0
y_{top}/c	Top wall location	1.0

Table 1: Wind tunnel computational domain boundary locations (Note: c is the chord of the airfoil).

The wind tunnel computational domain is shown in Figure 1 and the specific geometry values are given in Table 1. The x and y-velocity components, u and v, can be found from a streamfunction, Ψ, in two-dimensional flows,

$$u = \frac{\partial \Psi}{\partial y}, \qquad v = -\frac{\partial \Psi}{\partial x}.$$

In an irrotational flow, i.e. a flow in which the vorticity is zero, the governing equation for the streamfunction, Ψ, is,

$$\nabla^2 \Psi = 0,$$

At the inlet and outlet of the computational domain, we will assume that the flow is only in the x-direction (i.e. $v = 0$), thus the boundary conditions are,

$$\frac{\partial \Psi}{\partial x} = 0 \text{ at } x = x_{inl} \text{ and } x = x_{out}.$$

The top and bottom of the computational domain are walls and therefore streamlines of the flow. Thus, the boundary conditions on the walls will be Dirichlet conditions,

$$\Psi(x, y_{top}) = \Psi_{top}, \qquad \Psi(x, y_{bot}) = \Psi_{bot}.$$

Ψ_{top} and Ψ_{bot} can be set by fixing the mass flow rate through the tunnel. First, we note that the mass flow through the tunnel is given by,

$$\begin{aligned}
\dot{m}_{tunnel} &= \rho \int_{y_{bot}}^{y_{top}} u\, dy, \\
&= \rho \int_{y_{bot}}^{y_{top}} \frac{\partial \Psi}{\partial y}\, dy, \\
&= \rho \left(\Psi_{top} - \Psi_{bot} \right). \qquad (1)
\end{aligned}$$

To set the actual value of the mass flow, we will assume that the average upstream velocity is U_∞. With this assumption, the mass flow in the tunnel can also be written,

$$\dot{m}_{tunnel} = \rho U_\infty \left(y_{top} - y_{bot} \right). \qquad (2)$$

Thus, equating the mass flow from Equations (1) and (2) gives,

$$\Psi_{top} - \Psi_{bot} = U_\infty \left(y_{top} - y_{bot} \right).$$

Since Ψ can be raised arbitrarily by a constant (because only derivatives of Ψ determine the flow velocities), we will choose $\Psi_{bot} = U_\infty y_{bot}$ which results in the final form of the wall boundary conditions

$$\Psi_{bot} = U_\infty y_{bot}, \qquad \Psi_{top} = U_\infty y_{top}.$$

The airfoil surface will also be a streamline (i.e. a constant Ψ), thus the boundary condition on the airfoil will be Dirichlet,

$$\Psi|_{airfoil} = \Psi_a,$$

where Ψ_a is some constant. The value of the Ψ_a will be discussed in the descriptions of the tasks.

2 Tasks

2.1 Finite Element Method for Symmetric Flows (50%)

The first task is to develop a FEM solver for the flow around a symmetric airfoil at zero angle of attack. In this case, the flow will be symmetric (since the airfoil is also located at the center of the wind tunnel), thus, the value of the streamfunction on the airfoil must be zero to insure symmetry. Thus, $\Psi_a = 0$.

A grid of triangular elements has been generated for this airfoil and the data is stored in the MATLAB datafile **g0012.mat** (Note: since this datafile is stored in a binary form, you cannot edit it). A plot of the grid is shown in Figure 2. A skeleton MATLAB source code for your solver is in **tunflow.m**. Please read the comments in the source code both for information on how the grid data is stored and for how to write the rest of the code.

Documentation for this task is as follows:

- A thoroughly commented MATLAB source code (or codes if you broke the solver into more than module). 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 aerodynamics and Finite Element Methods to clearly understand everything you are doing in the code.
- Plots of the streamfunction contours for all cases simulated in this project.
- Note: since the algorithm is a standard FEM discretization of Laplace's equation, no other description of the algorithm is required beyond that provided in the source code.

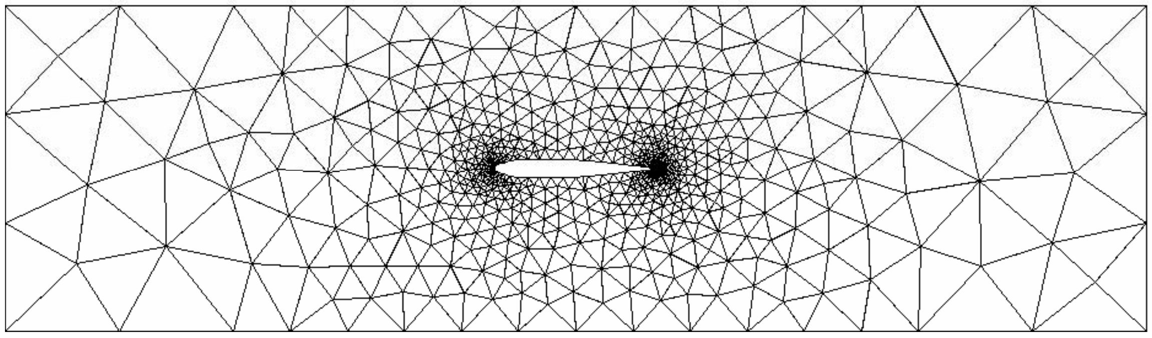

Figure 2: Grid for NACA 0012 problem

2.2 Calculation of Pressure Coefficient (10%)

The pressure coefficient in an inviscid, irrotational flow can be found using Bernoulli's equation,

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2} = 1 - \left(\frac{q}{U_\infty}\right)^2,$$

where $q = \sqrt{u^2 + v^2}$ is the local flow speed. The velocity components in each element can be computed from the gradients of the streamfunction and can then be used to find C_p in each element. Store your calculated C_p into the array **Cp** in the **tunflow.m** source. This will then be plotted when **tunplotflow.m** is called at the end of **tunflow.m**.

- Include a brief (but complete) description of how you calculated C_p from the FEM solution for Ψ.
- Include plots of the C_p distribution for all of your results in this project.

2.3 Modification for Non-symmetric Flows (20%)

For non-symmetric flows, the streamfunction on the airfoil surface is not known prior to the simulation. Thus, it must be found by some other constraint. To do this, we will use the Kutta condition. The Kutta condition requires the flow to leave the leading edge smoothly. To enforce this condition, we will require that the flow at the trailing edge be aligned along the bisector of the trailing edge. The geometry of the trailing edge is shown in Figure 3. For the flow to be aligned along $\vec{n}_{te}$, Ψ must be constant along this direction

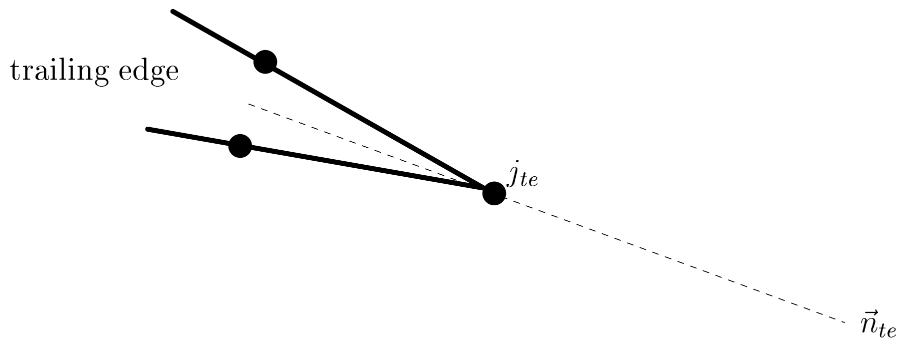

Figure 3: Trailing edge geometry. $\vec{n}_{te}$ = direction vector of trailing edge bisector. j_{te} = index of node at the trailing edge.

(since lines of constant Ψ are streamlines of the flow). Thus, the Kutta condition becomes,

$$\nabla\Psi \cdot \vec{n}_{te} = 0 \text{ at the trailing edge.}$$

Within the FEM discretization, we will implement this condition in a weighted residual form, specifically,

$$R_{te} \equiv \iint_{\Omega} w_{te} \nabla\Psi \cdot \vec{n}_{te}\, dA = 0,$$

where Ω is the entire computational domain and w_{te} is the weight function at the trailing edge node j_{te}. If N_v is the total number of vertices, we now have N_v equations arising from the weighted residuals and Dirichlet boundary conditions and one equation for the Kutta condition. The unknowns are the N_v values of Ψ at the nodes and the value of the streamfunction on the airfoil Ψ_a. Thus, we have $N_v + 1$ equations and unknowns. To implement the Kutta condition, the following procedure is suggested:

1. Add Ψ_a at the end of the vector of unknowns, i.e. in the **tunflow.m** code, we will let $psi(Nv+1) = \Psi_a$. To do this, we need to modify the Dirichlet boundary conditions on the airfoil. Suppose the index j was for a node on the airfoil. The residual equation at this node for symmetric flows was,

   $$R(j) \equiv psi(j) = 0,$$

   but now becomes,

   $$R(j) \equiv psi(j) - psi(Nv + 1) = 0.$$

   Thus, the stiffness matrix will now have a non-zero entries in the $Nv + 1$ column corresponding to its dependence on $psi(Nv + 1)$. Then, for this first step in making the Kutta condition modifications, we add the following equation,

   $$R(Nv + 1) \equiv psi(Nv + 1) = 0.$$

   Thus, if you have implemented this step correctly, the solution will be identical to that from Section 2.1 when run on the NACA 0012 grid.

2. Next, we discretize the R_{te} equation as given above. Note, this will require a little bit of code just to find the trailing edge node and the trailing edge bisector direction. Then, you need to calculate the weighted residual equation (and the corresponding matrix entries) which will include contributions from any triangle attached to the trailing edge node. For this weighted residual equation, you will probably want to use Gaussian quadrature to evaluate the integrals in each of these trailing edge triangles (see Section 2.5 for the necessary formula). When you have this implemented, test it on the NACA 0012 case. Since the flow is symmetric, this condition should produce a solution nearly identical to previous solution in which Ψ_a was hard-wired to zero.

3. Finally, run this modified solver on the **g4408.mat** and **g4408aoa10.mat** grids. These grids are both for a NACA 4408 airfoil with **g4408.mat** set at zero angle of attack and **g4408aoa10.mat** set at 10 degrees angle of attack. If the solver is working properly, you should be able to zoom into the trailing edge region and see that the streamfunction contours follow the trailing edge bisector direction.

4. Provide streamfunction plots to support the validity of your Kutta condition implementation and include C_p distribution plots.

5. Make sure to comment your implementation of the Kutta condition thoroughly in your code as described in Section 2.1.

2.4 Derivation of $\vec{x}$ to $\vec{\xi}$ Differential Area Relationship (10%)

Prove that,

$$dA = |J| d\xi_1\, d\xi_2 \text{ where } J = \frac{\partial x}{\partial \xi_1}\frac{\partial y}{\partial \xi_2} - \frac{\partial x}{\partial \xi_2}\frac{\partial y}{\partial \xi_1}.$$

Hint: calculate the area of a trapezoid with corners at $\vec{x}(\xi_1, \xi_2)$, $\vec{x}(\xi_1 + d\xi_1, \xi_2)$, $\vec{x}(\xi_1 + d\xi_1, \xi_2 + d\xi_2)$, and $\vec{x}(\xi_1, \xi_2 + d\xi_2)$.

2.5 Derivation of Single Point Gauss Quadrature (10%)

For the reference element in the ξ-plane with vertices at $(0,0)$, $(1,0)$, and $(0,1)$, consider the following single point Gauss quadrature rule,

$$\iint_{T_{ref}} g(\xi_1, \xi_2)\, d\xi_1\, d\xi_2 \approx \frac{1}{2} g(\frac{1}{3}, \frac{1}{3}).$$

Show that this quadrature is exact for any linear function of ξ_1 and ξ_2,

$$g(\xi_1, \xi_2) = c_0 + c_1\xi_1 + c_2\xi_2,$$

but not for a general bilinear function,

$$g(\xi_1, \xi_2) = c_0 + c_1\xi_1 + c_2\xi_2 + c_{12}\xi_1\xi_2,$$

where c_0, c_1, c_2, and c_{12} are constants.