Project 1: Mesh Generation

AE 623, Computational Fluid Dynamics II, Winter 2026

Due: January 30, 11:59pm, electronically via Canvas

1 Problem Description

1.1 Geometry

In this project you will generate unstructured, triangular meshes for a turbine blade passage, illustrated in Figure 1. Coordinates for the blade surface geometry are provided with this assignment in two separate files: `bladelower.txt` and `bladeupper.txt`. The domain has two sets of periodic boundaries: one ahead of the blade and one behind it – these arise from modeling one blade out of an infinite number of identical blades stacked vertically.

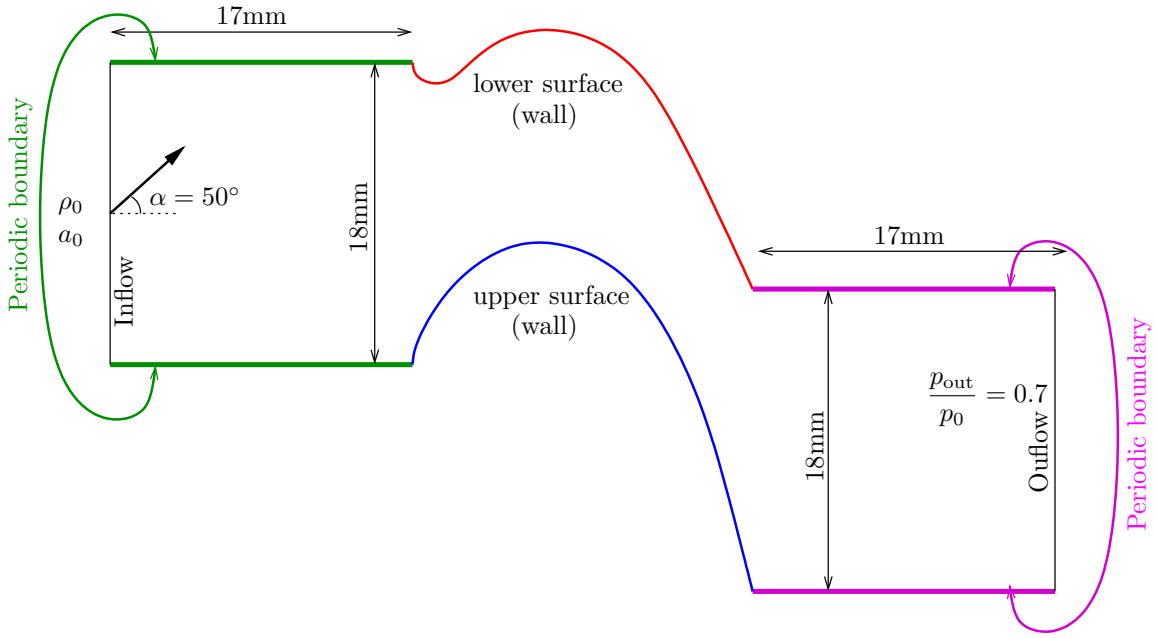

Figure 1: Computational domain for the turbine blade passage.

1.2 Meshes

Your baseline "coarse" mesh should be around 2000 cells, with smaller cells near the blade, particularly at the leading and trailing edges. Nodes placed on periodic boundaries must match. In this case, this means the same number of nodes on the upper and lower surfaces, each matched in the horizontal (x) coordinate. The .gri file format requires specification of the matching node pairs.

1.3 Uniform Refinement

Include support for uniform mesh refinement, in which each mesh element is split into four subelements through edge bisection, as shown in Figure 2. Nodes inserted on bisected boundary edges on the curved blade geometry should be projected to the true geometry, using a cubic spline of the provided points. Note, spline the lower and upper surfaces together, to ensure slope continuity at the leading edge. You will generate three refinements of your coarse mesh, with approximately 8k, 32k, and 128k elements.

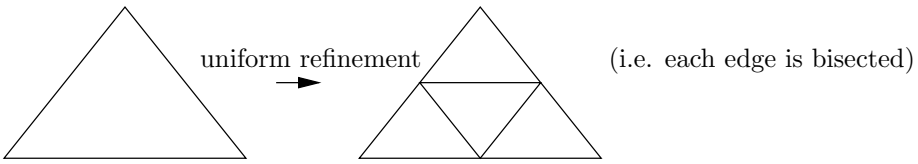

Figure 2: Uniform refinement of one cell into four cells.

1.4 Projection

Implement an ability to project points from the domain interior to the blade geometry. Describe your process and show a figure with a few sample projections. Use both the projection wall distance, d, and the x location of the projected point on the blade, x_b, to create a mesh sizing function, $h(d, x_b)$, where h is the desired edge length. Your function should produce small elements close to the wall and near the leading and trailing edges. It should be scaled to yield the coarse target mesh size with refinement. Plot contours of your sizing function as a field on one of the finer meshes.

1.5 Local Refinement

Implement local refinement for your mesh to improve resolution in important regions, as determined by your sizing function. Measure the length of each edge of the mesh and compare it to the requested mesh from the sizing function, computed at the edge midpoint. If the edge is too long, flag it for refinement. Once edges are flagged, refine all cells adjacent to flagged edges. These cells will fall into one of three categories, shown in Figure 3, and they should be refined as indicated. After

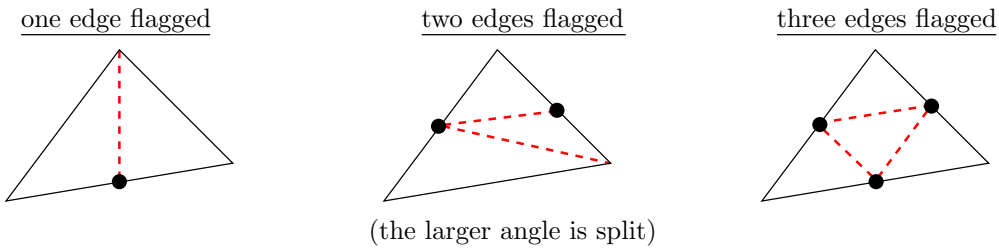

Figure 3: Refinement of triangles given edge splittings.

refinement, smooth the mesh nodes using several iterations of a simple weighted average of the current location and the average of the neighbors:

$$\vec{x}_i' = (1-\omega)\vec{x}_i + \frac{\omega}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\vec{x}_j, \tag{1}$$

where $\mathcal{N}(i)$ is the set of nodes adjacent to node i, and $|\mathcal{N}(i)|$ is the cardinality of this set (i.e. the number of neighbors). Do not apply the smoothing to nodes on boundaries, but do snap all new nodes on the boundary to the true geometry (from the splines). Repeat the refinement and smoothing until no edges are flagged.

Note, start the process with a relatively coarse mesh so that you obtain larger elements away from the blade. Tune your sizing function with a multiplicative factor to attain the desired number of elements.

1.6 Verification

As part of this project, you will be computing normal vectors and lengths for interior and boundary faces. Implement the following mesh verification test of these quantities:

- Loop over all interior and boundary faces in your mesh.
- On each face, calculate the normal vector, $\vec{n}$. Make this vector point out of the domain for boundary faces and from a consistently-chosen L to R element pair for interior faces.
- On each face, multiply the normal by the face length l and add/subtract the result to/from a running total on the adjacent elements. The goal is by the end of your calculation to have computed, for each element e,

$$E_e \equiv \sum_{i=1}^{3} \vec{n}_{ei}^{\text{outward}} l_{ei},$$

  where the subscript e denotes quantities associated with element e. This vector quantity should be zero, to machine precision, on each element.

1.7 The `.gri` Mesh File Format

Use the `.gri` format to store your meshes; a sample file, `test.gri`, is included on Canvas with this assignment. The `.gri` file contains the node coordinates, a list of boundary groups and faces, and a list of elements (cells). The general format and definitions for a `.gri` file are given below:

```
nNode nElemTot Dim
for i = 1:nNode
  x(i) y(i) [z(i)]
nBGroup
for i = 1:nBGroup
  nBFace(i) nf(i) [Title(i)]
  for j = 1:nBFace(i)
    NB(i,j,1) .. NB(i,j,nf(i))
for i = 1:nElemGroup
  nElem(i) Order(i) Basis(i)
  for j = 1:nElem(i)
    NE(i,j,1) .. NE(i,j,nn(i))
nPG "PeriodicGroup"
for i = 1:nPG
  nPGNode(i) Periodicity(i)
  for j = 1:nPGNode(i)
    NP(i,j,1) NP(i,j,2)
```

`nNode`	Number of nodes; linear and high order
`nElemTot`	Total number of elements in all groups
`Dim`	Dimension: 2 or 3
`nBGroup`	Number of boundary groups (bgroups)
`nBFace(i)`	Number of boundary faces (bfaces) in group i
`nf(i)`	Number of linear nodes per bface on group i
`Title(i)`	String title of bgroup i (no spaces)
`NB(i,j,*)`	List of nf(i) linear nodes for bface j on bgroup i
`nElemGroup`	Number of element groups (egroups)
`nElem(i)`	Number of elements in egroup i
`Order(i)`	Geometry order of elements in egroup i (q)
`Basis(i)`	String defining the geometry interpolation basis
`nn(i)`	Number of nodes per element in egroup i
`NE(i,j,*)`	List of nn(i) nodes for element j in egroup i
`nPG`	Number of periodic groups
`nPGNode(i)`	Number of node pairs in periodic group i
`Periodicity(i)`	String indicating type of periodicity for group i
`NP(i,j,:)`	Matching node pair j for periodic group i

The number of element groups is not given explicitly in the file, and instead the element reader stops when the total number of elements read (sum of `nElem(i)`) reaches `nElemTot`. The numbering for the boundary face nodes (`NB`) is not relevant: the outward normal is obtained from the geometry of the element containing these nodes. Only the linear nodes (relevant for curved meshes, in later projects) on the boundary faces are to be specified. The numbering for the element nodes, `NE`, follows the convention illustrated in Figure 1.7. All indices in the file are 1-based (they start at 1, not 0). The string defining the geometry interpolation basis, `Basis(i)`, is "TriLagrange" for triangles. The periodicity string indicates translational versus rotational periodicity. For the blade passage, it is "Translational".

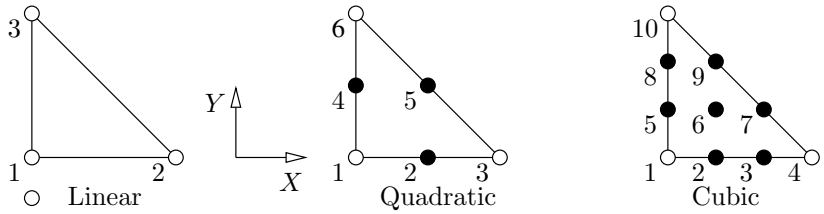

Figure 4: Node ordering for linear, quadratic, and cubic triangles, shown in element reference space $(X, Y, [Z])$. The white nodes denote the linear nodes. Higher-order triangles follow the same pattern. In this project, you are generating meshes with only linear triangles, but later projects will use high-order curved elements.

2 Tasks

Address the following tasks in your report.

1. **[15%]** Generate a coarse, unstructured, triangular mesh, with around 1000-1500 elements, using an existing unstructured mesh generation program, or a method of your choice. You should not cluster the elements in any specific areas: leave that to the refinement with your sizing function. Describe your initial mesh generation approach, include an image of your mesh in your report, and include the `.gri` file in your `.zip` archive.

2. **[15%]** Write a code that processes the mesh generated in the previous task and generates the following matrices (all 1-based):

   - `I2E`: a mapping from interior faces (faces are edges in 2D) to elements. Note, treat edges on periodic boundaries as interior edges. The number of rows is the total number of interior faces in the mesh. Each row contains four integers, `elemL, faceL, elemR, faceR`, where `elemL, elemR` are the indices of the two adjacent elements. To avoid ambiguity, define the left element (L) as the one with the smaller element index. `faceL, faceR` are numbers between 1 and 3 inclusive that encode which *local* face in each element corresponds to the interior face under consideration. In each element, local face number i is the face opposite local node i.

   - `B2E`: a mapping from boundary faces to elements and boundary groups. The number of rows is the total number of boundary faces. Each row contains three integers: `elem, face, bgroup`, where `elem` is the element adjacent to that boundary face, `face` is the boundary face's local face number within that element, and `bgroup` is the boundary index of that face. The boundary index should correspond to the order of boundary groups in the `.gri` file.

   - `In`: normal vectors for interior faces. The number of rows is the number of interior faces. Each row contains two floating-point numbers: the x and y components of the normal that points from the L to the R element.

   - `Bn`: normal vectors for boundary faces. The number of rows is the total number of boundary faces. Each row contains two floating-point numbers: the x and y components of the normal that points outward from the adjacent element.

   - `Area`: the area of each element. The number of rows is the total number of elements, and each row contains one number: the area of the element.

   Include in your report a description of the methods used, and a printout of these matrices for

the test mesh included with this assignment.

3. [**15%**] Implement the mesh verification test for your coarse mesh, using the normal vectors in `In` and `Bn`. Compute the maximum magnitude of the error E_e on each element, for both the test mesh and your coarse mesh. Discuss the results.
4. [**15%**] Implement the projection and wall distance calculations, and develop a sizing function. Present the results requested in Section 1.4.
5. [**15%**] Implement the local refinement capability described in Section 1.5, and apply it along with the sizing function to generate the coarse, $\sim$ 2000-element mesh. Describe your algorithm and show the final mesh.
6. [**15%**] Generate three uniform refinements of your locally-refined mesh from the previous part. Include pictures of these meshes in your report, and the mesh files in your `.zip` archive. Also run the verification test on each of the refined meshes.

Your codes should not use any algorithms that that have an order "N^2" cost, where N can be the number of elements, nodes, edges, etc. Accompanying this assignment are codes for reading and plotting `.gri` files in Python and Matlab.

3 Deliverables

Each group should turn in, electronically via Canvas,

1. A technical report as a `.pdf` file that describes your methods and results, and that answers the questions. The report should be professional, complete, and concise. **10%** of your grade will be determined by the presentation: neatness, labeling of axes, spelling, etc.
2. Documented source files of all codes you wrote for this project, as one `.zip` archive. You should provide all files necessary to run your program(s).

This is a group assignment. At the end of the report, summarize the contributions of each group member to the method development, code, results, and report.