General homework note: *Please upload your solutions as one* .pdf *document. Scans of hand-written work are acceptable, but please use an app that produces legible documents if using your phone or tablet. Supporting codes should be uploaded separately as a* .zip *archive of source code files.*

1. The Euler equations in quasi-linear form [25%]

The compressible Euler equations are presented in Section 5.3.1 of the notes. The given *conservation* form can be written in *quasi-linear* form as follows (for 1D):

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0 \quad \Rightarrow \quad \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial x} = 0$$

a) Compute the linearization matrix $\mathbf{A} \equiv \frac{\partial \mathbf{F}}{\partial \mathbf{u}}$. Perform the calculations by hand (you may check with a computer).

b) Compute the eigenvalues and right eigenvectors of $\mathbf{A}$. Normalize the three eigenvectors so that they each have a value of 1 in the first entry. Again, perform the calculation by hand.

c) Classify the system of equations as hyperbolic, elliptic, parabolic, or hybrid.

3. The Newton-Raphson method [25%]

Numerical simulations, particularly those involving the finite-difference method, may use mappings between a *physical space*, on which the problem is defined, and a *reference space*, on which the problem is easier to solve. Consider a reference space consisting of a unit square, $0 \leq X \leq 1$, $1 \leq Y \leq 1$, and a reference-to-physical space map given by

$$\begin{aligned} x &= X + 0.3XY + 0.1Y^3 \\ y &= Y - 0.4X^2 + 0.6X \end{aligned}$$

a) Provide a computer-generated plot of the unit-square reference domain mapped into physical space.

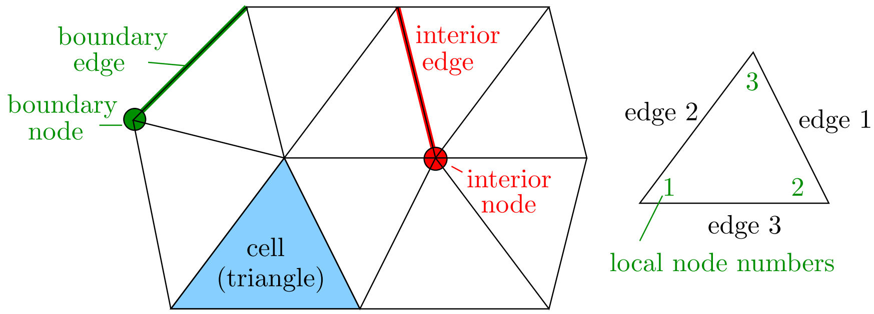
b) Write a program that inverts the map using the Newton-Raphson method. That is, given (x, y) the code should return (X, Y). In addition to uploading your code, give the resulting (X, Y) for the following three points:

$$\begin{aligned} (x, y) &= (0.4, 0.6) \\ (x, y) &= (0.5, 0.7) \\ (x, y) &= (0.8, 0.9) \end{aligned}$$

Use $(X, Y) = (0.5, 0.5)$ as the starting guess, and for each run provide the history of the residual convergence with Newton iteration. *Hint: you can debug your code with simpler mappings, e.g. the identity map $x = X, y = Y$, or a proportional scaling, $x = 2X, y = 2Y$. Also, your residual norms should converge quadratically when close to the root, so you should not need many Newton iterations.*

7. Mesh connectivity [50%]

In this problem you will write a code that processes a triangular mesh, in particular identifying the connectivities between cells. Such information is needed by methods that compute fluxes between cells, e.g. the finite-volume discretization. Definitions relevant to triangular meshes are given in the figure below.



Together with this assignment, you are given two files: `V.txt`, which contains (x, y) coordinates of the mesh nodes, and `E.txt` which defines the mesh triangles as triplets of node indices. Note the definition of local edge numbers based on local node numbers in the figure above.

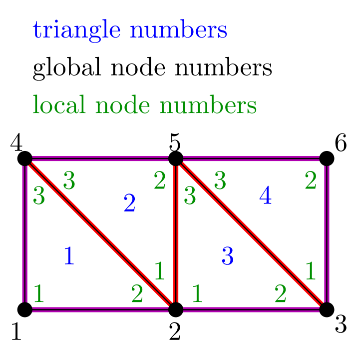
a) Make a plot that shows the triangles in the mesh and include two views: one showing the entire mesh, and one showing a zoom in the area $-1 \leq x \leq 2$ and $-1 \leq y \leq 1$.

b) Write a code that identifies the cell-to-cell connections in the mesh. Your algorithm should be of **linear complexity** in terms of the number of nodes and cells. That is, you should not use any nested loops where both inner/outer loops range over all cells or nodes. Your code should write a text file, `C.txt`, that contains the connectivity information in the form of four space-separated numbers per connection:

$$t_1 \; e_1 \; t_2 \; e_2$$

where t_1 and $t_2 > t_1$ are numbers identifying adjacent triangles, and e_1 and e_2 are the local edge numbers of the edge shared between the two triangles. The connectivities should be sorted first by increasing t_1, and then by increasing t_2. Include the output file `C.txt` together with your code submission in the `.zip` archive.

Example: The figure below presents a `E.txt` and `C.txt` files (as matrices – do not include the commas in your `C.txt` file) for a small mesh. This is a good small test case for verifying your code.



E.txt $\begin{bmatrix} 1, & 2, & 4 \\ 2, & 5, & 4 \\ 2, & 3, & 5 \\ 3, & 6, & 5 \end{bmatrix}$ C.txt $\begin{bmatrix} 1, & 1, & 2, & 2 \\ 2, & 3, & 3, & 2 \\ 3, & 1, & 4, & 2 \end{bmatrix}$