The Problem: Euler

In this project you will consider the *linear* Euler Equations, which are given by:

$$\frac{\partial}{\partial t}\begin{bmatrix}u\\v\\p\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}p\\0\\u\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}0\\p\\v\end{bmatrix}=0 \qquad \text{in } \Omega,$$

with boundary condition $un_x + vn_y = 0$ on $\partial\Omega$. Here, p is the pressure, u and v are the velocities in the x and y directions, and $\mathbf{n} = [n_x, n_y]$ is the outward normal to the domain. The boundary conditions imposes a velocity field that is tangent to the boundary.

To study these equations you will be given several meshes:

- `square.neu`, `square2.neu`, `square3.neu`: A square mesh on $[-1, 1]$ with increased refinement.
- `neweire.neu`: An unstructured mesh of the Ireland coastline.

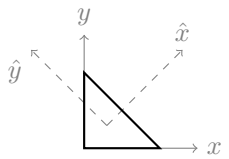
To begin, write the Euler Equations in matrix form:

$$\frac{\partial}{\partial t}\mathbf{q}+\frac{\partial}{\partial x}A\mathbf{q}+\frac{\partial}{\partial y}B\mathbf{q}=0 \qquad \text{in } \Omega,$$

where

$$\mathbf{q}=\begin{bmatrix}u\\v\\p\end{bmatrix} \qquad A=\begin{bmatrix}0&0&1\\0&0&0\\1&0&0\end{bmatrix} \qquad B=\begin{bmatrix}0&0&0\\0&0&1\\0&1&0\end{bmatrix}$$

Now in this problem we have an unstructured mesh. So we'll be considering the solution *per triangle* in the mesh. If we consider a generic triangle such as



then we will write the equations as a 1D PDE in the direction of $\hat{x}$. To do this, use a transformation $Q = \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}$. Then

$$\hat{\mathbf{x}} = Q\mathbf{x} \qquad \mathbf{x} = Q^T\hat{\mathbf{x}}.$$

Using the chain rule we can use

$$\frac{\partial}{\partial x}=\frac{\partial\hat{x}}{\partial x}\frac{\partial}{\partial\hat{x}}+\frac{\partial\hat{y}}{\partial x}\frac{\partial}{\partial\hat{y}}=\cos\theta\frac{\partial}{\partial\hat{x}}-\sin\theta\frac{\partial}{\partial\hat{y}}$$

and

$$\frac{\partial}{\partial y}=\sin\theta\frac{\partial}{\partial\hat{x}}+\cos\theta\frac{\partial}{\partial\hat{y}}$$

This gives

$$\mathbf{q}_t+(A\cos\theta+B\sin\theta)\,\mathbf{q}_{\hat{x}}+(-A\sin\theta+B\cos\theta)\,\mathbf{q}_{\hat{y}}=0$$

and $\mathbf{n} = \begin{bmatrix}\cos\theta & \sin\theta\end{bmatrix}$. Then

$$\mathbf{q}_t+(An_x+Bn_y)\,\mathbf{q}_{\hat{x}}+(-An_y+Bn_x)\,\mathbf{q}_{\hat{y}}=0$$

which is

$$\mathbf{q}_t+C\mathbf{q}_{\hat{x}}+D\mathbf{q}_{\hat{y}}=0$$

but notice that $\hat{y}$ is *tangent* to the boundary. So we ignore this term, leading to

$$\mathbf{q}_t + C\mathbf{q}_{\hat{x}} = 0$$

At this point, C is given as

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} n_x + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} n_y = \begin{bmatrix} 0 & 0 & n_x \\ 0 & 0 & n_y \\ n_x & n_y & 0 \end{bmatrix}$$

You can compute the eigenvalues/eigenvectors

$$\lambda_0 = 1 \quad \lambda_1 = -1 \quad \lambda_2 = 0$$

and

$$\mathbf{w}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} n_x \\ n_y \\ 1 \end{bmatrix} \quad \mathbf{w}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} n_x \\ n_y \\ -1 \end{bmatrix} \quad \mathbf{w}_2 = \begin{bmatrix} n_y \\ -n_x \\ 0 \end{bmatrix}$$

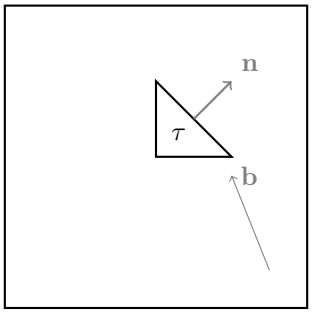
Now let $R = [\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2]$, which is orthogonal, to transform the system into a new set of decoupled equations:

$$\hat{\mathbf{q}}_t + R^T C R \hat{\mathbf{q}}_{\hat{x}} = 0$$

where $R^T C R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix}$.

Discretization

At this point, it's helpful to review the basic of a discretization. Let's consider some flow field $\mathbf{b}$ in a domain Ω as in the following



We want to know the flux of some quantity q through a boundary on τ. This will be equal to the amount of flow ($\mathbf{b}$) that is normal to a surface:

$$\int_{\partial\tau} -\mathbf{b} \cdot \mathbf{n}\, q\, ds$$

Then, with the time derivative, we can note the rate of change:

$$\frac{\partial}{\partial t} \int_\tau q dx = \int_{\partial\tau} -(\mathbf{b} \cdot \mathbf{n}) q ds = \int_\tau -\nabla \cdot (\mathbf{b} q) dx$$

This means that (with constant $\mathbf{b} = [b_1, b_2]$) that

$$\int_\tau q_t + b_1 q_x + b_2 q_y\, dx = \int_\tau q_t + \mathbf{b} \cdot \nabla q\, dx = 0$$

At this point it's helpful to think of 1D example. Let τ be a line segment. In this case, an upwind scheme looks like (with advection of $a > 0$)

$$q_k^{j+1} = q_k^j - \frac{\Delta t}{\Delta x} a(q_k^j - q_{k-1}^j)$$

If we pose this in terms of cell normals of element τ then

$$q_k^{j+1} = q_k^j - \frac{\Delta t}{\Delta x} a(-1)(q_{k-1}^j - q_k^j)$$

Now for k_i as the i^{th} neighbor of k, this gives:

$$q_k^{j+1} = q_k^j - \frac{\Delta t}{|\tau|} \sum_{k_i} \left(\frac{an_i - |an_i|}{2} \right) (q_{k_i} - q_k)$$

Now in general, for a triangle τ, we write this as a sum over neighboring elements and note that the point evaluations for the flux in 1D become line integrals over the edges in 2D:

$$\frac{\partial}{\partial t} \int_\tau q \, dx = - \int_{\partial \tau} \mathbf{b} \cdot \mathbf{n} q \, ds = - \sum_{i=0}^{2} \int_{\partial \tau_i} \mathbf{b} \cdot \mathbf{n} q \, ds$$

Turning to the upwind scheme (and summing over neighboring elements of k):

$$q_k^{j+1} = q_k^j - \frac{\Delta t}{|\tau_k|} \sum_{k_i} \int_{\partial \tau_{k_i}} \left(\frac{\mathbf{b} \cdot \mathbf{n}_{k_i} - |\mathbf{b} \cdot \mathbf{n}_{k_i}|}{2} \right) (q_{k_i}^j - q_k^j)$$

From here, if the conserved quantities are considered constant per element, then the integral becomes the edge length:

$$q_k^{j+1} = q_k^j - \frac{\Delta t}{|\tau_k|} \sum_{k_i} |\partial \tau_{k_i}| \left(\frac{\mathbf{b} \cdot \mathbf{n}_{k_i} - |\mathbf{b} \cdot \mathbf{n}_{k_i}|}{2} \right) (q_{k_i}^j - q_k^j)$$

Since $h_{k_i} = 2|\tau_k|/|\partial \tau_{k_i}|$ (or the height is 2 times the area divided by the edge length),

$$q_k^{j+1} = q_k^j - \Delta t \sum_{k_i} \left(\frac{\mathbf{b} \cdot \mathbf{n}_{k_i} - |\mathbf{b} \cdot \mathbf{n}_{k_i}|}{h_{k_i}} \right) (q_{k_i}^j - q_k^j)$$

Here, $\mathbf{n}_{k_i}$ is the outward normal to edge j and h_{k_j} is the height of the triangle normal to edge j.

The extension of this to Euler above

$$\hat{\mathbf{q}}_t + R^T C R \hat{\mathbf{q}}_{\hat{x}} = \hat{\mathbf{q}}_t + \Lambda \hat{\mathbf{q}}_{\hat{x}} = 0$$

gives (for each face i):

$$\hat{\mathbf{q}}_k^{j+1} = \hat{\mathbf{q}}_k^j - \frac{1}{h_{k_i}} (\Lambda - |\Lambda|) \left(\hat{\mathbf{q}}_{k_i}^j - \hat{\mathbf{q}}_k^j \right)$$

Now we can transform back to $\mathbf{q}$ by left multiplying the equation by R and right multiplying the equation by R^T.

$$\mathbf{q}_k^{j+1} = \mathbf{q}_k^j - \frac{1}{h_{k_i}} R (\Lambda - |\Lambda|) R^T \left(\mathbf{q}_{k_i}^j - \mathbf{q}_k^j \right)$$

$\Lambda - |\Lambda|$ has only one entry in the middle (which is -2). This yields in term of the velocities u and v, and pressure p we have

$$\begin{bmatrix} u_k^{j+1} \\ v_k^{j+1} \\ p_k^{j+1} \end{bmatrix} = \begin{bmatrix} u_k^j \\ v_k^j \\ p_k^j \end{bmatrix} + \Delta t \frac{1}{h_{k_i}} \begin{bmatrix} n_x \\ n_y \\ -1 \end{bmatrix} \left(n_x^{k_i} (u_{k_i}^j - u_k^j) + n_y^{k_i} (v_{k_i}^j - v_k^j) - (p_{k_i}^j - p_k^j) \right)$$

Now writing over each of the faces we arrive at

$$\begin{bmatrix} u_k^{j+1} \\ v_k^{j+1} \\ p_k^{j+1} \end{bmatrix} = \begin{bmatrix} u_k^j \\ v_k^j \\ p_k^j \end{bmatrix} + \Delta t \sum_{i=0,\ldots,2} \frac{1}{h_{k_i}} \begin{bmatrix} n_x^{k_i} \\ n_y^{k_i} \\ -1 \end{bmatrix} \left(n_x^{k_i} (u_{k_i}^j - u_k^j) + n_y^{k_i} (v_{k_i}^j - v_k^j) - (p_{k_i}^j - p_k^j) \right)$$

Putting it together ...

So how do you piece this together into a solver?

1. Check the orientation and read the mesh. This step is completed for you in `project-template.ipynb`.
2. Set up the arrays. You will need several arrays to for metrics for the mesh the element lists (`E`) and (`V`). Namely you will need to initialize the following:
   - `nx`, a `ne` $\times 3$ array of values of the x-component of the normal to each edge.
   - `ny`, a `ne` $\times 3$ array of values of the y-component of the normal to each edge.
   - `h`, a `ne` $\times 3$ array of values for the triangle height orthogonal to each edge. The height is two times the area of the triangle divided by the edge length. This you will fill in below.
   - `cx`, a `ne` $\times 1$ array of values for the x-component of the cell center.
   - `cy`, a `ne` $\times 1$ array of values for the y-component of the cell center.
3. Construct Element Relationships. This step is completed for you. It constructs `E2E`, `V2V`, and `Enbrs`.
4. Set the initial values. Here you will use $u = 0$ and $v = 0$ (corresponding to still water), and consider a Gaussian for a pressure pulse:

   $$p(x, y) = e^{-(x^2+y^2)/0.1} \quad \text{or for the costline mesh,} \quad p(x, y) = e^{-((x-200)^2+(y+400)^2)/10000}$$

   Each u, v, and p should be a `ne` $\times 1$ array, so one value for each element. You can plot this in the next step with Paraview to verify your initial data profiles.
5. Setup "right" and "left" elements. This is done for you. After this, for each "left" element (in `mapL`) you will know the "right" element that neighbors it for each edge (in `mapR`). Sometimes these are called "+" and "-" elements.
6. Time step. Here the boundary conditions are already implemented as $u_R = -u_L$, $v_R = -v_L$, and $p_R = p_L$, for each boundary edge (where R and L represent the left and right elements). Then you will define the flux and update using the derivation in the previous section. This is the most difficult step but you'll be able to do it with only a few lines of code.

At this point you should have a running solver. For your write-up, address the following

1. Verify that your simulation "works" by using the square meshes and initial conditions

   $$\begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cos(\pi x)\cos(\pi y) \end{bmatrix}$$

   which has the exact solution

   $$\begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} \sin(\pi x)\cos(\pi y)\sin(\pi t\sqrt{2})/\sqrt{2} \\ \cos(\pi x)\sin(\pi y)\sin(\pi t\sqrt{2})/\sqrt{2} \\ \cos(\pi x)\cos(\pi y)\cos(\pi t\sqrt{2}) \end{bmatrix}$$

   What order of accuracy do you observe?
2. The stability condition for this problem is

   $$\Delta t \leq \frac{1}{2}\min(h^{k_i})$$

   Do you observe this stability condition computationally?
3. Use the coastal mesh to try a pressure pulse. Qualitatively does the solution make sense?
4. Try to refine the mesh (with `Vnew, Enew = refine_mesh.refine2dtri(V, E)`). Does the solution quality improve?