AE 333/707: PrL 1. Due on 26 Aug 2022. Total=10

This is an optional programming assignment that you may take up in lieu of the 10-mark first quiz. If you submit this, I will NOT evaluate your first quiz.

This is a GRADED INDIVIDUAL work; submission deadline is at the beginning of class on the due date mentioned in the title.

Before the deadline, submit a link in Moodle to a Google Colab document of your work, including detailed formulation/explanation, code listing and the results and discussions. Make sure that the Colab workbook can be EXECUTED by all (TAs and me); please do verify this for hassle-free grading. If you are using any language other than Python, then please submit the file containing your code – i.e., '*.m' file for MATLAB or '*.c' file for C, etc. Needless to say, comments are essential in your code.

Apart from this, you have to submit a hard-copy printout of the same Colab workbook in class by the deadline. If you are using some other language, then submit a proper report of your work – including the formulation, CODE, results and discussions.

You are expected to work independently on this assignment, although you may consult the TAs or the instructor. I may conduct a separate viva to gauge your individual understanding of your own solution! Any hint of plagiarism or academic misconduct will, at a minimum, incur a grade penalty for all parties involved.

1. (10 points) Consider the potential flow comprising of the superposition of an array of 10 equal clockwise vortices of total strength 0.15 units (i.e., 0.015 units each) distributed along a 1 unit-long line at an angle of -10° to an oncoming (left-to-right) uniform stream of speed 1 unit.

   Write a computer code to compute and plot the streamlines for this particular potential flow. The code can be in any computer language of your choice.

   Instructions:

   - No points will be given if you use the in-built 'contour' function of your language, or any in-built streamline plotting routine for that matter.
   - Instead, you should take the following steps:
     1. Formulate the analytical expression for the velocity vector field from the velocity potential function of the given flow.
     2. Create a function that returns the velocity vector field for your flow (two inputs – x and z – and two outputs – u and w).
     3. The streamline plotting function itself should take the following input arguments:

- A handle to the above velocity vector field function.
- The initial point on the streamline (i.e., (x_0, z_0)).
- The step size along the streamline, Δs.
- The rectangular bounding box of the figure (i.e., minimum and maximum limits of x and z) where the streamlines should end.

4. Starting from the initial point, the streamline plotting function should calculate the point (x_i, z_i) on the streamline as $x_i = x_{i-1} + u_{i-1}\Delta s/V_{i-1}$ and $z_i = z_{i-1} + w_{i-1}\Delta s/V_{i-1}$, where $u_i \equiv u(x_i, z_i)$, etc., and $V_i = \sqrt{u_i^2 + w_i^2}$. This is essentially the simplest possible integrator (or space marching routine) – the forward Euler integrator. The plot should be a set of straight lines joining the sequence of the above points. The streamlines should cease whenever the new point falls outside the specified rectangular bounding box.
5. Finally, write a driver routine that calls the streamline plotting function multiple times, once for each streamline (i.e., each initial point). It is here that you will hard-code the step size Δs and the bounding box of the plot. Try to place the streamlines at equal intervals in the far-upstream (freestream) flow.

- The flow pattern figure you generate from the code should not be modified afterwards; all figure elements should be plotted by the code directly.
- You do NOT need to draw arrows on the streamlines.
- Try to make your code efficient – in particular, it shouldn't take more than a few minutes to run on a standard laptop (or in Google Colab).
- Discuss how you would improve your algorithm (no need to program it).