Zach Swain

MEEG 332

Chapter 8 Coding Challenge: Panel Methods

For part 1, the approach taken was to specify a relevant domain over which to define the polynomial shape of the airfoil. At first, .075 was used as the maximum x over which to define the polynomial, but the end did not reach zero as it should. It was determined that .757 was a more suitable value to set as a maximum for the given polynomial. The polynomial only defined the top half of the symmetric airfoil, so its negative was used to define the bottom half. This resulting shape was then plotted with relevant formatting and equal scaling to produce the plot prompted.

For part 2, the number of corners/sides was taken to be 100. The finite x and resulting zeta and y values then needed to be redefined to coincide with equally spaced sides and corners of the approximating polygon. This was done, and a plot was produced to verify its accuracy, but not delivered as it was not prompted for.

For part 3, the length of each segment was determined and stored, using the x and y values solved for in part 2 and relevant algebra.

For part 4, the panel midpoints were found and stored; again using the x and y values previously solved for and relevant algebra.

For part 5, e\_t and e\_n dummies were initially defined to provide the correct dimensioning to then iteratively append to. This was done using the given-defined e\_t and e\_n formulas in the prompt.

For part 6, the given-defined Uinf, A\_ij, and b parameters first had to be defined as they are the relevant parameters when solving for Q = A\_ij/b. This was done in a similar vein as part 5 in iteratively appending dummy initial set-ups, although they differ in specifics.

For part 7, a gridspacing in x and y were first defined, and later optimally tweaked, to set up the desired meshgrid. The r\_ij, u\_ij, and v\_ij parameters were then iteratively solved via their previous prompt definitions. They were then utilized in conjunction with the x and y meshgrids to provide a quiver plot overlaid on the prepviously provided plot in part 1.

For part 8, a streamline spacing was first defined, and later optimally tweaked. The streamline function was then used to produce overlaid streamlines of this spacing over the plot produced in part 7.

For part 9, the v\_ij term first had to be defined, as it was the remaining parameter to be stored before the dotting could be performed. This was again done iteratively through stored values in x and in y to produce the resulting u’s and v’s at each i,j midpoint. This v\_ij was then dotted with the stored e\_t value (initially to some dismay, but TA Gene pointed out they should be transposed as-defined) to produce the desired v\_tj. This result was then plotted against x\_j as prompted.

For part 10, given-defined constant values were first stored for later use, and then a dummy P set up was initialized. Actual P values were then iteratively appended, and the results were plotted against x\_j as prompted.