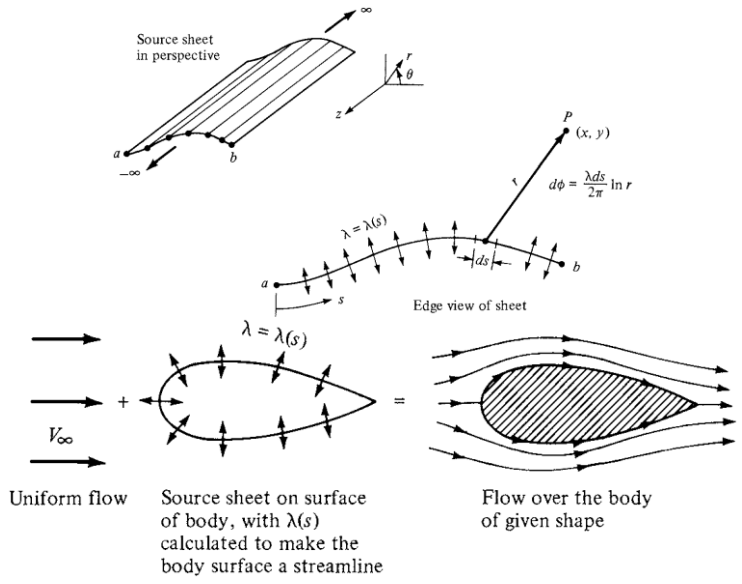
KOUSTUBH DWIVEDY, 140010016

AE 333 Source Panel Method Assignment

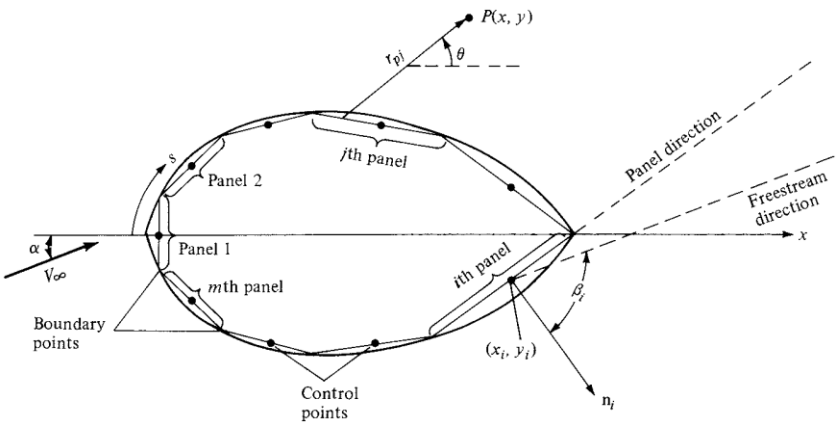
Source Panel Method:

Ideal flow over airfoil (by ideal, it is meant that the flow is incompressible, inviscid and irrotational, in short, the flow is potential flow) can be simulated using a linear combination of potential functions which satisfy the Laplace equation. These potential functions are supposed to satisfy the boundary equations both farfield as well as on boundary of object which we are modelling. Modelling simple objects like cylinders, doublets and ovals are simple enough to do by hand and intuition but for complex objects, intuition doesn't scale.

This problem is solved using a modelling technique in which we model an airfoil or any body using a line (need not be straight) of continuous source functions in which the strength of source is defined as strength per unit length. This method does not produce any net lift on the body. This is the case of non-lifting flows. This framework creates the mathematical background to go about studying general shapes but doesn't provide a set algorithm per se.



Within this framework, there exists the source Panel Method which approximates the above construct with straight lines of constant strength with are connected together. The boundary conditions are that at the center of each segment, the net velocity into the body is 0 (normal to surface). We need to ensure that the regions with high gradients are approximated with more number of panels as compared to that of low gradient surface.



Algorithm Used:

The potential at any point due to all panels is given by:

$$\phi(P) = \sum_{j=1}^{n} \Delta\phi_j = \sum_{j=1}^{n} \frac{\lambda_j}{2\pi} \int_j \ln r_{pj}\, ds_j$$

Where r is given by:

$$r_{pj} = \sqrt{(x - x_j)^2 + (y - y_j)^2}$$

The values of lambda are calculated by equating the normal velocity at the middle of every panel to 0. The n equations of V_normal come out to be the following (including the effects of V_infinity) which upon solving using the **linsolve** function of MATLAB gives the values of lambda:

$$\frac{\lambda_i}{2} + \sum_{\substack{j=1 \\ (j\neq 1)}}^{n} \frac{\lambda_j}{2\pi} \int_j \frac{\partial}{\partial n_i}(\ln r_{ij})\, ds_j + V_\infty \cos\beta_i = 0$$

The integral above is given using. This integral will be an n*n matrix (non symmetric with diagonal elements of value 0) where n is the number of panels:

$$I_{i,j} = \frac{C}{2} \ln\left(\frac{S_j^2 + 2AS_j + B}{B}\right) + \frac{D - AC}{E}\left(\tan^{-1}\frac{S_j + A}{E} - \tan^{-1}\frac{A}{E}\right)$$

$$A = -(x_i - X_j)\cos\Phi_j - (y_i - Y_j)\sin\Phi_j$$

$$B = (x_i - X_j)^2 + (y_i - Y_j)^2$$

$$C = \sin(\Phi_i - \Phi_j)$$

$$D = (y_i - Y_j)\cos\Phi_i - (x_i - X_j)\sin\Phi_i$$

$$S_j = \sqrt{(X_{j+1} - X_j)^2 + (Y_{j+1} - Y_j)^2}$$

$$E = \sqrt{B - A^2} = (x_i - X_j)\sin\Phi_j - (y_i - Y_j)\cos\Phi_j$$

After getting the lambda's, we get the tangential velocities at the center of every panel using:

$$V_i = V_{\infty,s} + V_s = V_\infty \sin\beta_i + \sum_{j=1}^{n} \frac{\lambda_j}{2\pi} \int_j \frac{\partial}{\partial s}(\ln r_{ij})\, ds_j$$

The above equation again required the evaluation of integral which will again be an n*n matrix (non-symmetrical with diagonal elements 0)

$$\int_j \frac{\partial}{\partial s}(\ln r_{ij})ds_j = \frac{D-AC}{2E}\ln\frac{S_j^2+2AS_j+B}{B} - C\left(\tan^{-1}\frac{S_j+A}{E} - \tan^{-1}\frac{A}{E}\right) \quad \begin{matrix}\sin\beta_i = \cos\Phi_i \\ \cos\beta_i = -\sin\Phi_i\end{matrix}$$

The pressure coefficients can now be calculated from the following:

$$C_{p,i} = 1 - \left(\frac{V_i}{V_\infty}\right)^2$$

The Lift and Drag Coefficients can now be found out using the following:

$$C_l = \frac{\sum_i^n (Cp_i * l_i * \sin(\beta_i))}{b}$$

$$C_d = \frac{\sum_i^n (Cp_i * l_i * \cos(\beta_i)}{b}$$

Where Cpi is the Pressure Coeff, li the length and βi the angle between normal and v_infinity at the i'th panel and b is the chord.

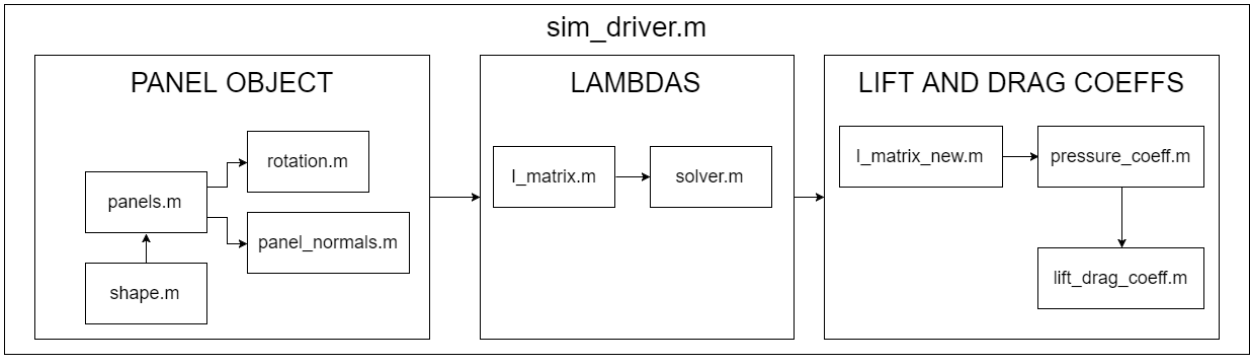
A rough check of accuracy can be made using:

$$\sum_{j=1}^{n} \lambda_j S_j = 0$$

Code Structure:

The code has been written in MATLAB and the version control is done on Github. The code and it's associated commits can be found on https://github.com/koustubh-dwivedy/AE333

The code has been made modular with the sim_driver.m script being the backbone (or simulation driver) of the code.

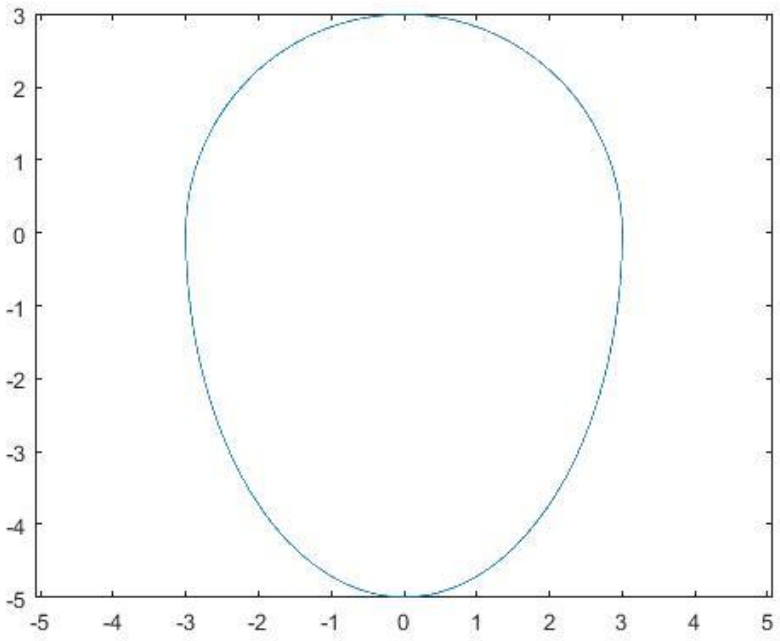


1. shape.m is a function which defines the shape of the body being simulated.
2. panels.m takes output of shape.m and creates the specified number of panels (namely the coordinates of beginning, center and end of panels)
3. rotation.m is responsible for rotating the panels created by the angle of attack specified to it.
4. panel_normals.m calculates the array of angles made by v_infinity and panel normals.
5. l_matrix.m creates the n*n matrix required in the calculation of lambdas.
6. solver.m solves for the lambdas using the inbuilt **linsolve** function of MATLAB.

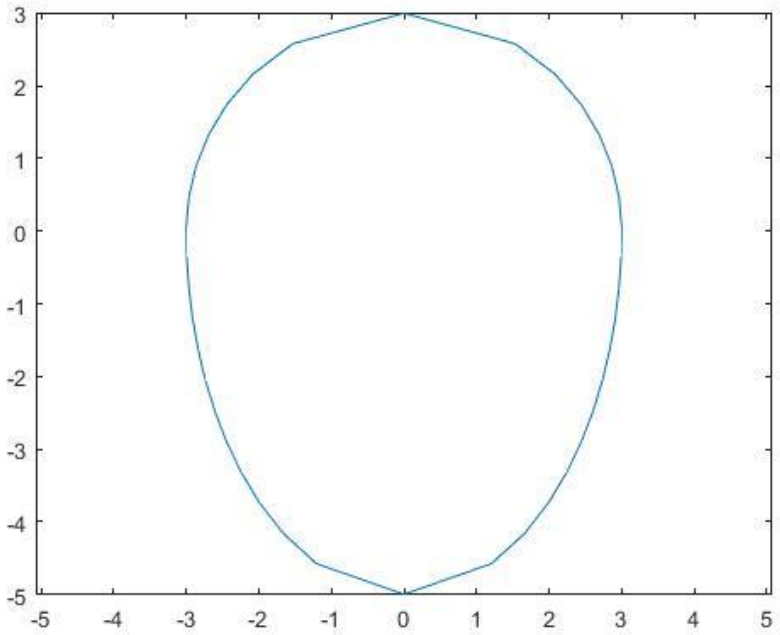
7. I_matrix_new.m creates the n*n matrix required in the calculation of v_tangential at every panel.
8. pressure_coeff.m takes output of I_matrix_new.m and evaluates the tangential velocity which it further uses in the evaluation of coefficient of pressure at every panel.
9. lift_drag_coeff.m as the name suggests, calculate the coefficient of lift and drag of the body.

Shape:

With 5000 Panels:



With 38 Panels:



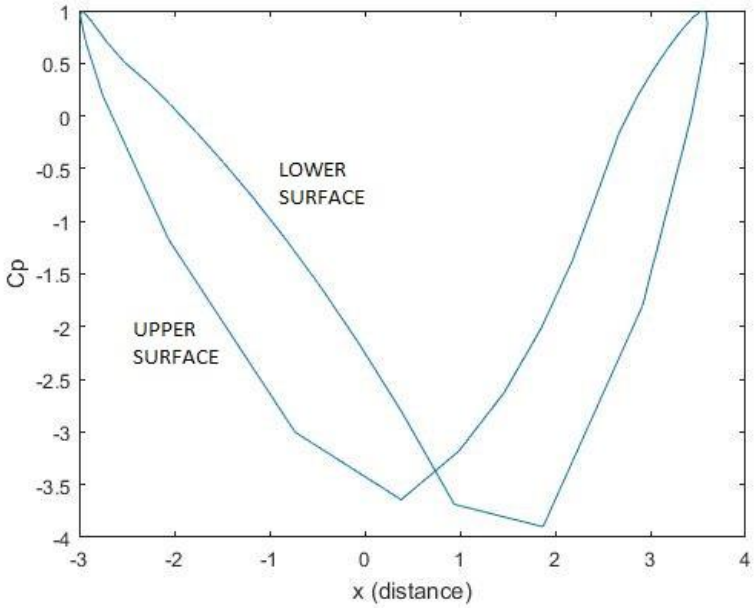
Cp Distribution:

X-Axis: Distance

Y-Axis: Cp

V_infinity = 15 m/s

Angle of Attack: -30º



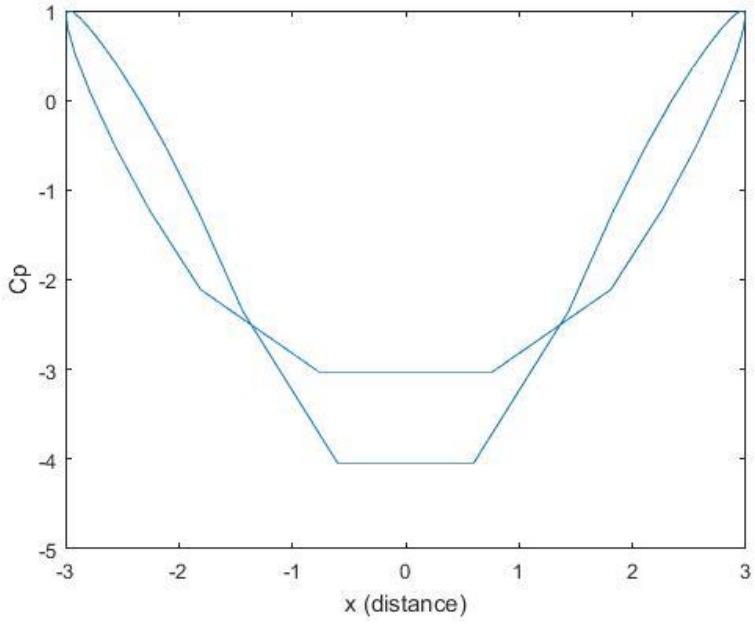
X-Axis: Distance

Y-Axis: Cp

V_infinity = 15 m/s

Angle of Attack: 0º

Plots of upper and lower surface are identical in this case.

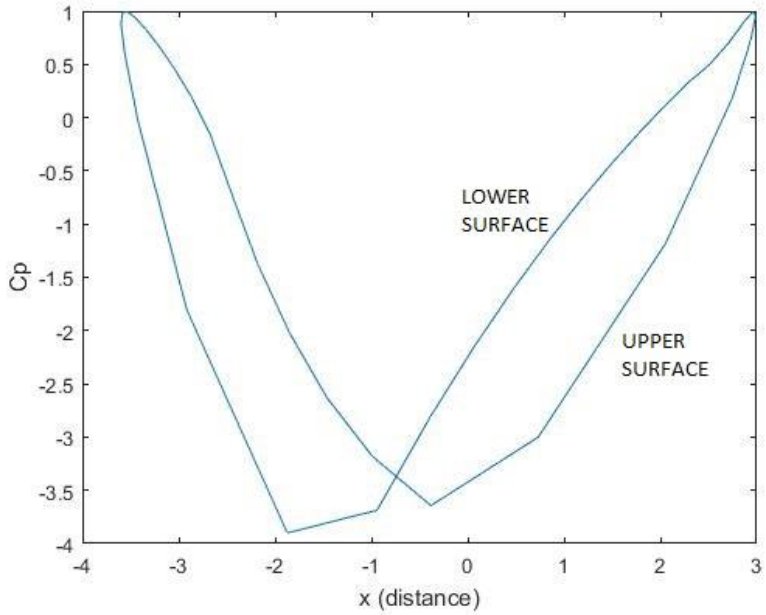


X-Axis: Distance

Y-Axis: Cp

V_infinity = 15 m/s

Angle of Attack: 30°

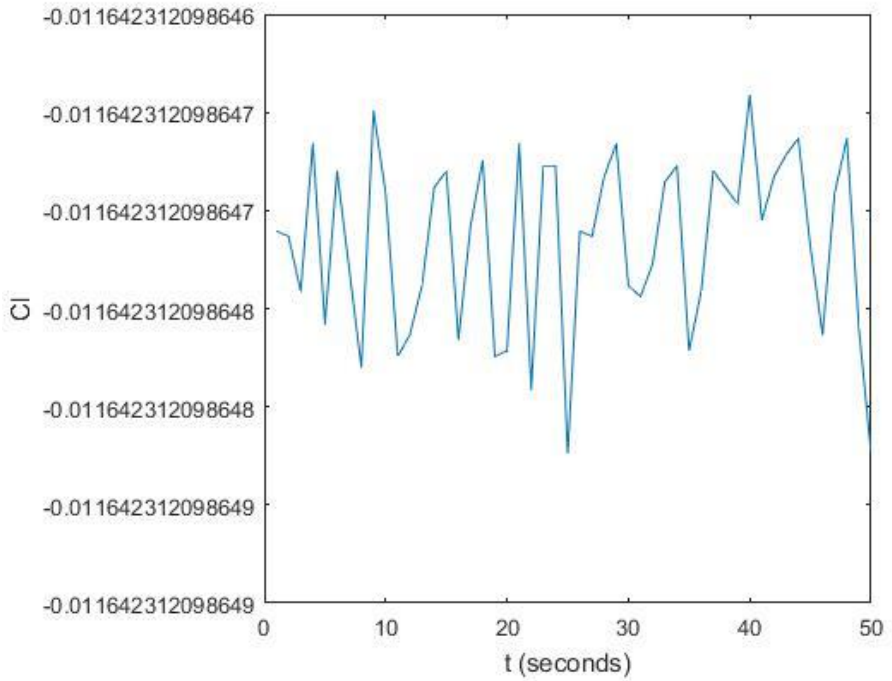


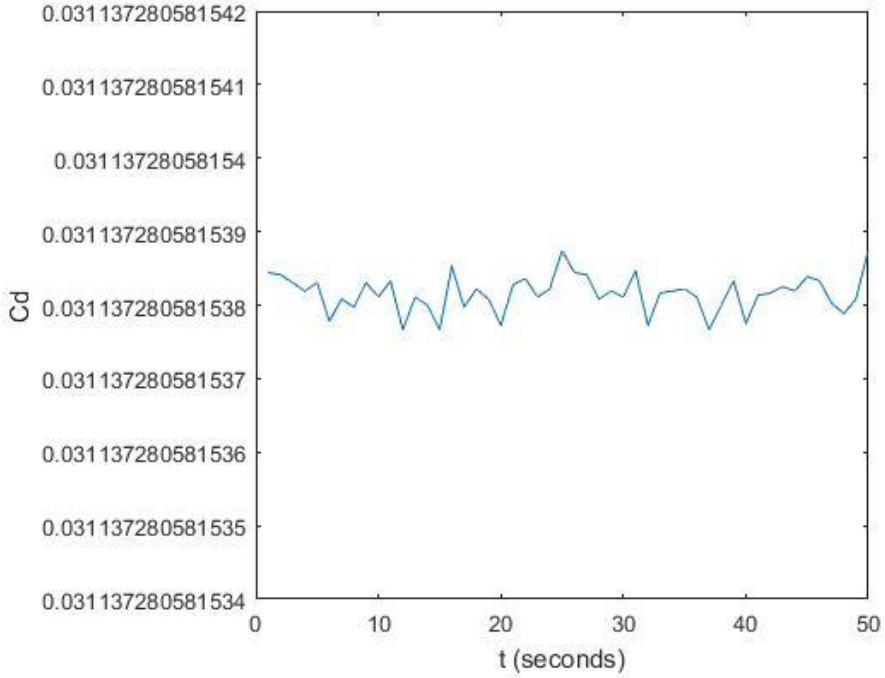
Cl & Cd Distribution:

V_infinity: 15*|cos(t)|

X-Axis: t (time)

AoA: -30°

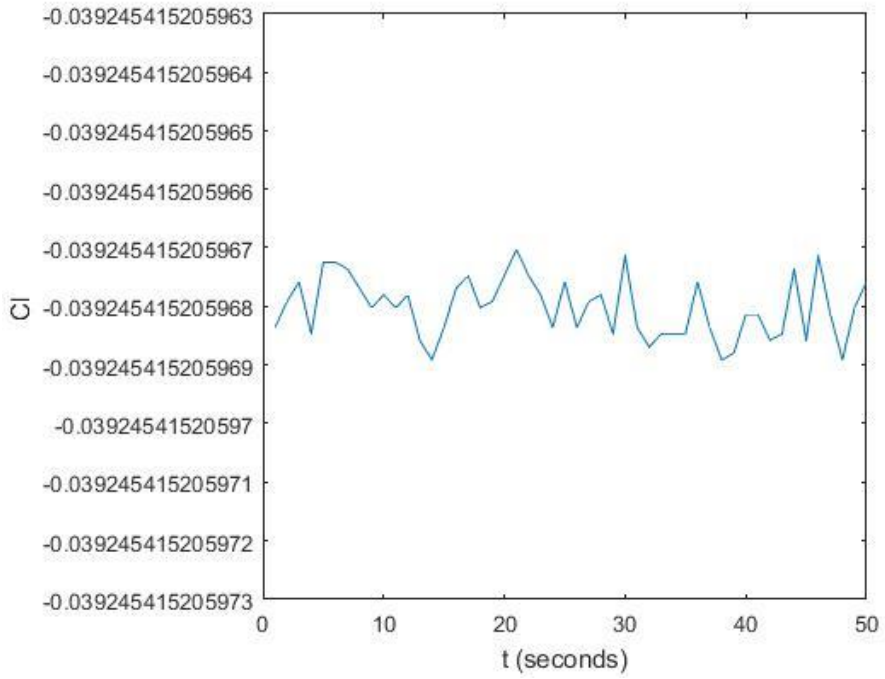


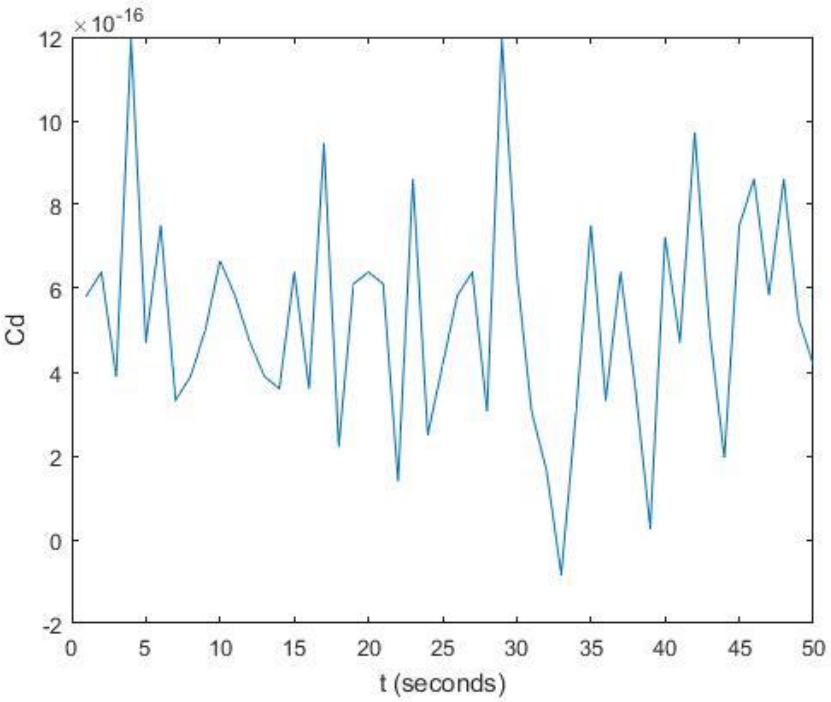


V_infinity: 15*|cos(t)|

X-Axis: t (time)

AoA: 0º

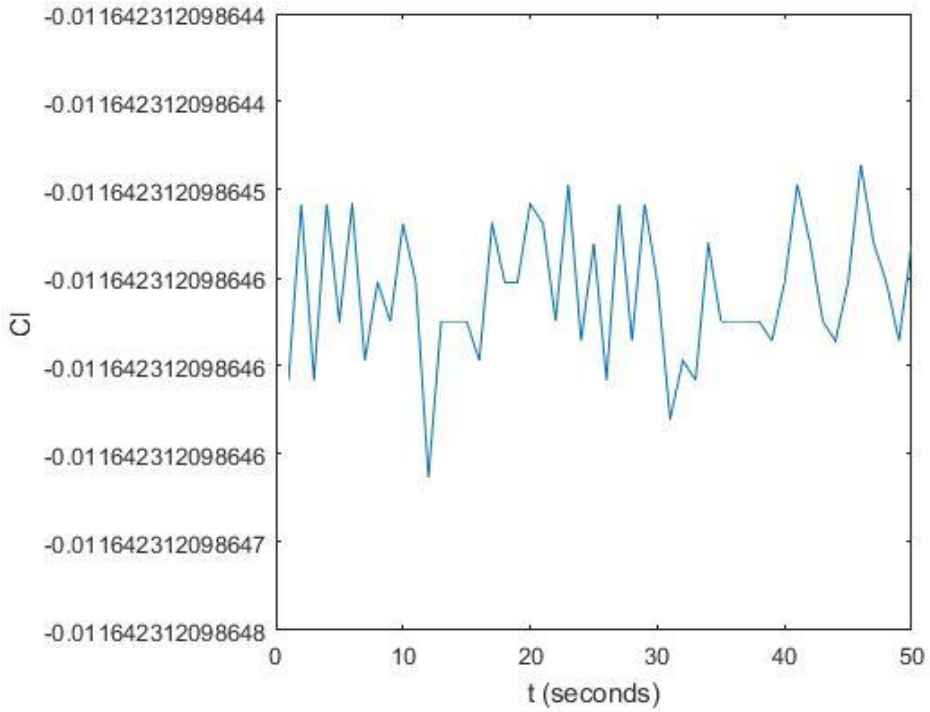


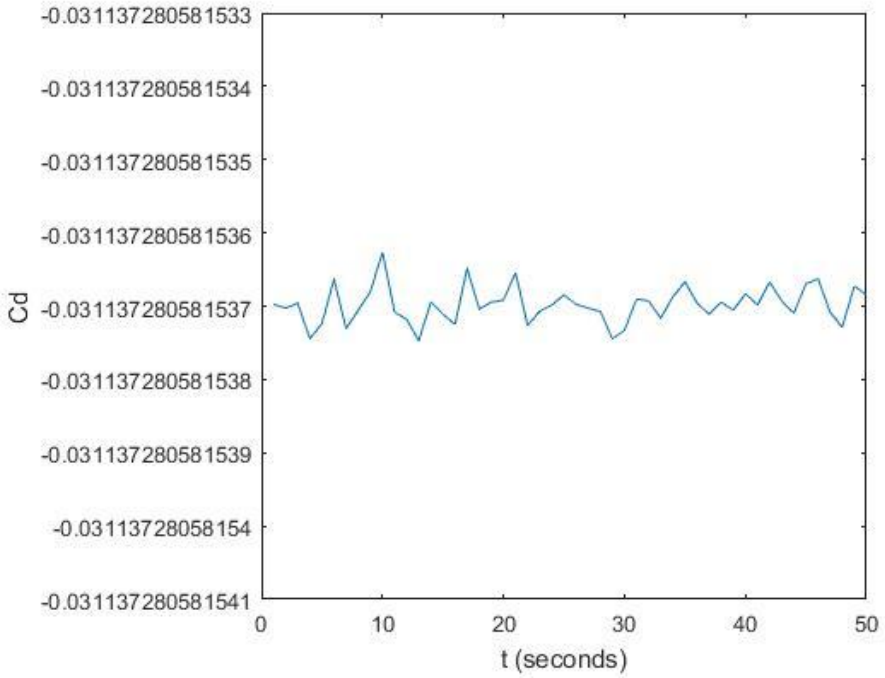


V_infinity: 15*|cos(t)|

X-Axis: t (time)

AoA: 30°





Conclusions:

We observe that the Cd and Cl are not changing with respect to time. There are slight fluctuations which can be attributed to numerical and floating point errors. Another observation which was made was that as the number of panels are increased in the code, the value of Cl and Cd converge to 0 due to 0 circulation as implied in d'Alemberts Paradox. The code was tested for n=5000 panels, in which case the coefficients fluctuated in the range of 10^-16 which is roughly 0 in computer floats.

The simulation were roughly instantaneous when the number of panels lied in the range of 500. In the range of 5000, the simulations were slow to run.

In my example of 38 panels, the Cl and Cd values were fluctuating about a non 0 number which is due to the fact that we approximated Cp of the entire panel with a single number which is incorrect. Also, there were inherent numerical errors which were propagating during the entire simulation. This is correctly reflected when we perform the simulation for high values of panels.

The reason why Cd and Cl do not change with time is that the change in velocity is reflected at every point instantaneously which can be seen as taking the earlier field and multiplying it with a single scalar constant. This causes no change in the Cp which in turn reflects in our plots of Cl and Cd.