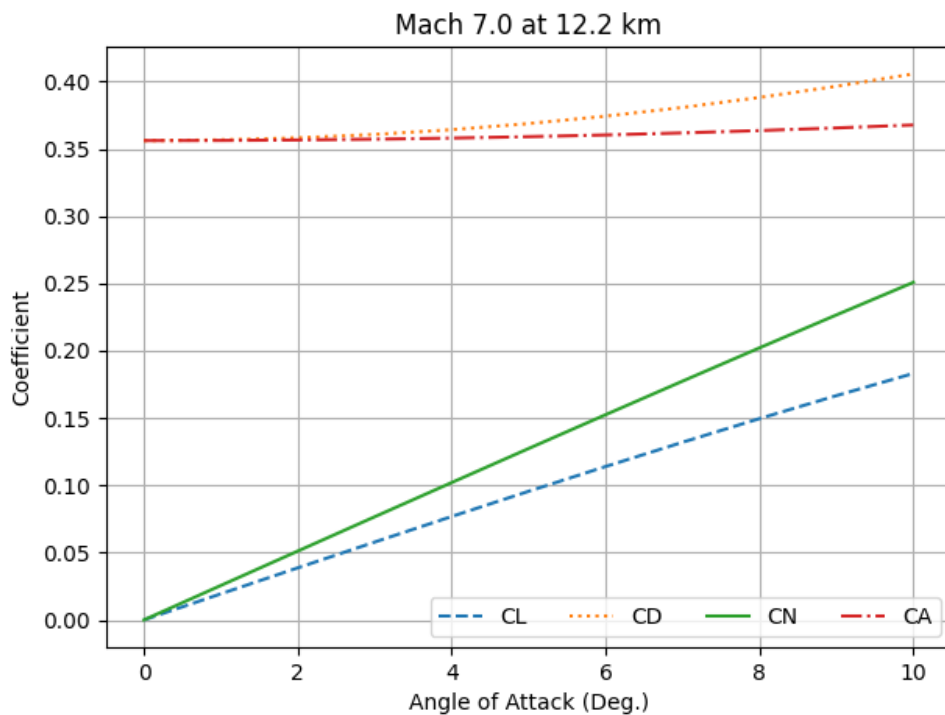


Case 1: Mach 7,  $h = 12.2$  km

$C_{p_{\max}}$  = 1.83951       $T_{\infty}$  = 216.65 K       $P_{\infty}$  = 18868.244 Pa

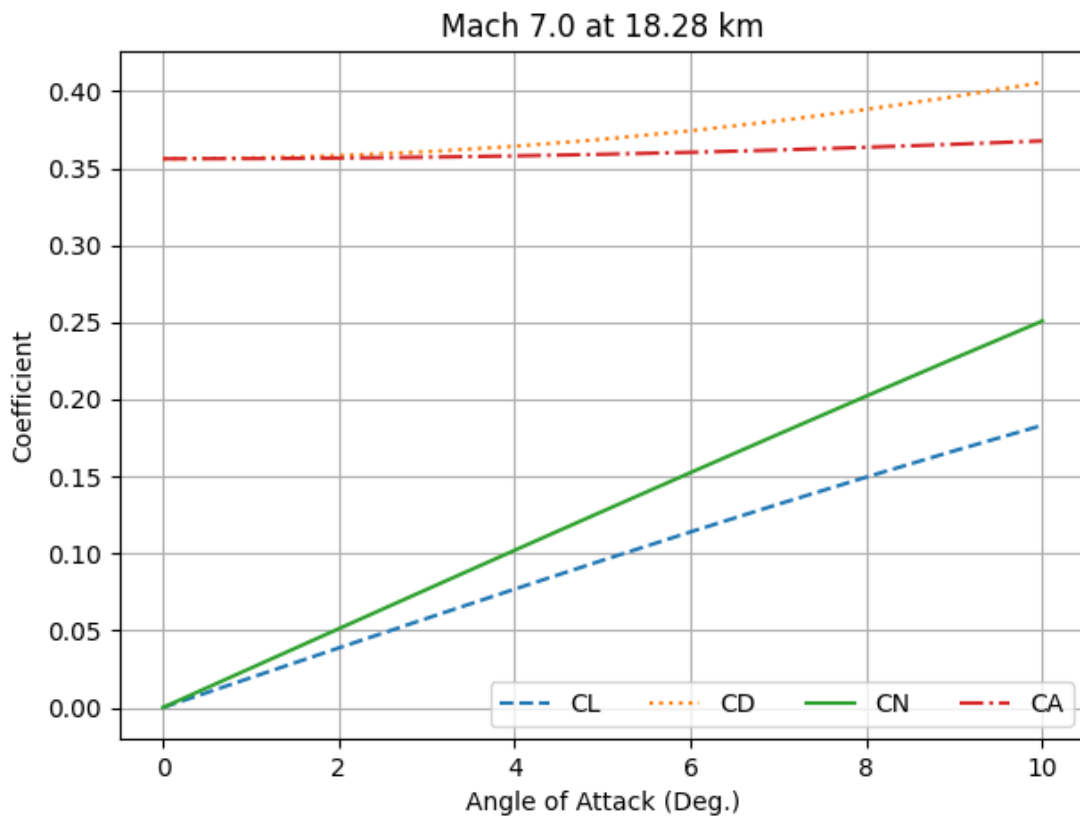
Angle of Attack (deg)	$C_A$	$C_N$	$C_L$	$C_D$
0	0.35618	0	0	0.35618
1	0.3563	0.0256	0.01937	0.35669
2	0.35665	0.05116	0.03868	0.35822
3	0.35723	0.07666	0.05786	0.36076
4	0.35805	0.10207	0.07684	0.3643
5	0.3591	0.12735	0.09557	0.36884
6	0.36039	0.15248	0.11397	0.37435
7	0.3619	0.17742	0.132	0.38082
8	0.36364	0.20215	0.14957	0.38823
9	0.3656	0.22663	0.16665	0.39656
10	0.36779	0.25084	0.18316	0.40576



Case 2: Mach 7,  $h = 18.28$  km

$C_{p_{\max}}$  = 1.83938       $T_{\infty}$  = 216.65 K       $P_{\infty}$  = 7300.152 Pa

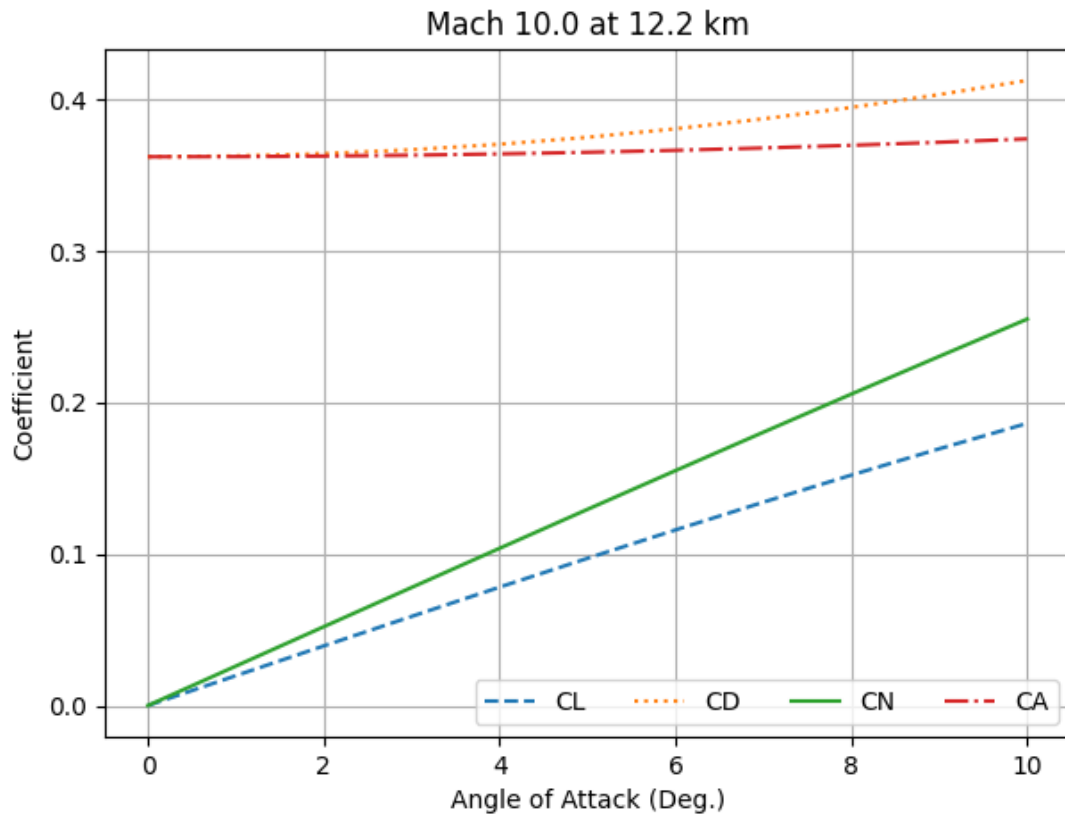
Angle of Attack (deg)	$C_A$	$C_N$	$C_L$	$C_D$
0	0.35615	0	0	0.35615
1	0.35627	0.02559	0.01937	0.35666
2	0.35662	0.05116	0.03868	0.35819
3	0.35721	0.07665	0.05785	0.36073
4	0.35803	0.10206	0.07684	0.36427
5	0.35908	0.12734	0.09556	0.36881
6	0.36036	0.15247	0.11397	0.37432
7	0.36187	0.17741	0.13199	0.3808
8	0.36361	0.20214	0.14956	0.38821
9	0.36558	0.22661	0.16664	0.39653
10	0.36777	0.25082	0.18314	0.40573



Case 3: Mach 10, h = 12.2 km

$$C_{p_{\max}} = 1.87127$$

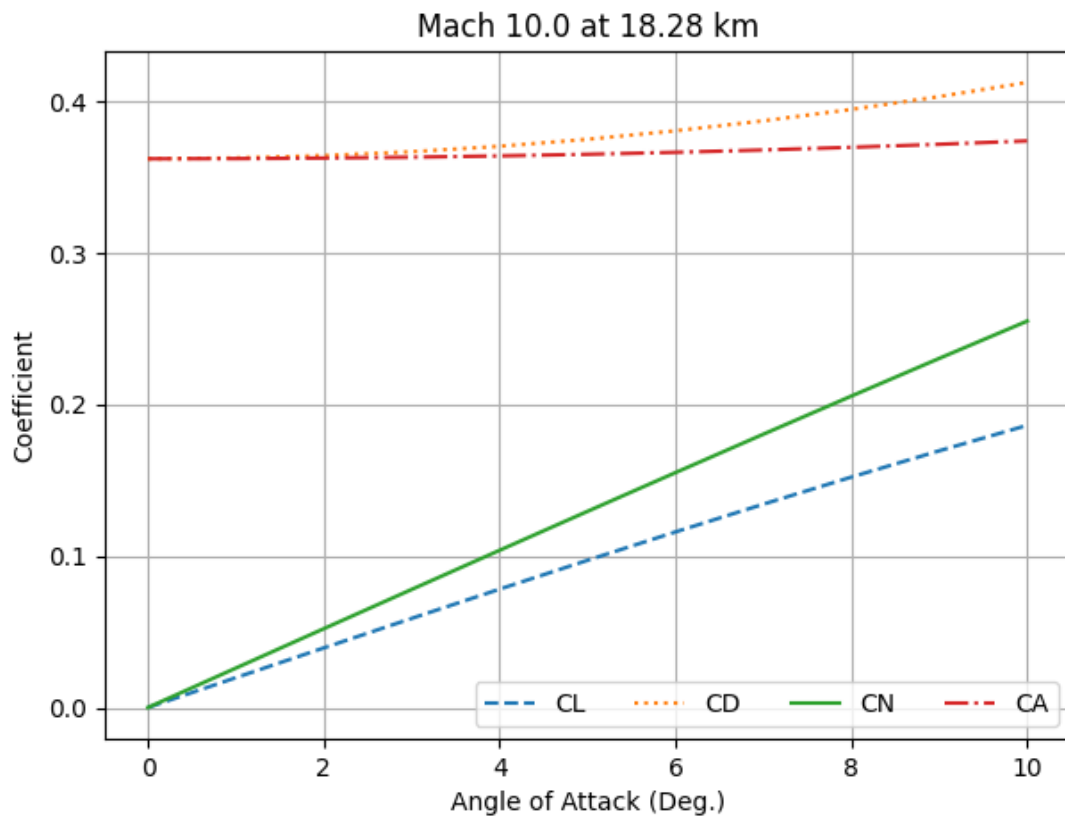
Angle of Attack (deg)	$C_A$	$C_N$	$C_L$	$C_D$
0	0.36233	0	0	0.36233
1	0.36245	0.02604	0.01971	0.36285
2	0.36281	0.05204	0.03935	0.3644
3	0.3634	0.07798	0.05886	0.36698
4	0.36423	0.10383	0.07817	0.37059
5	0.3653	0.12955	0.09722	0.37521
6	0.36661	0.15511	0.11594	0.38081
7	0.36815	0.18049	0.13428	0.3874
8	0.36992	0.20564	0.15216	0.39494
9	0.37192	0.23054	0.16952	0.4034
10	0.37414	0.25517	0.18632	0.41277



Case 4: Mach 10, h = 18.28 km

$$C_{p_{\max}} = 1.87205$$

Angle of Attack (deg)	$C_A$	$C_N$	$C_L$	$C_D$
0	0.36248	0	0	0.36248
1	0.3626	0.02605	0.01972	0.363
2	0.36296	0.05206	0.03937	0.36455
3	0.36355	0.07802	0.05888	0.36714
4	0.36439	0.10387	0.0782	0.37075
5	0.36546	0.12961	0.09726	0.37536
6	0.36676	0.15518	0.11599	0.38097
7	0.3683	0.18056	0.13433	0.38756
8	0.37007	0.20573	0.15222	0.3951
9	0.37207	0.23064	0.1696	0.40357
10	0.3743	0.25527	0.1864	0.41294



## Part 1 – Discussion

To calculate the aerodynamic coefficients, the body angles for the discretized points starting at the end point were calculated using tangent wedge summation but the 2<sup>nd</sup> and 3<sup>rd</sup> points were used to calculate the nose axial coefficient. Since the modified Newtonian method is used,  $C_p$  max is calculated as 2.0 minus the ratio of the air density upstream and downstream the shock using the 1962 Atmospheric model and equilibrium curve fits. The max  $C_p$  will be used in the tangent wedge summation to calculate the aerodynamic coefficients. The  $C_L$  and  $C_D$  were calculated using the normal and parallel components of the axial and normal coefficients,  $C_A$  and  $C_N$ . As the angle of attack increases and as the local angle along the nose geometry decreases, the error would increase, especially for the rear nose portion. This is due to the fact  $C_p$  at the rear is not 0 in contrast to regular Newtonian theory where the  $C_p$  would be 0 in the rear portion of the nose since it is facing away from the freestream. This is because the assumption is that the downstream conditions of the normal shock would apply to flow around the nose geometry.

Extra Credit:

Oblique Shock:

ObSA (deg.)	P E5 (Pa)	T (K)	h (kJ/kg)	rho E-1 (kg/m^3)	u (m/s)	Cp (kJ/kg)
45.04140	4.37247	2015.66620	2341.76754	7.34352	333.75649	1.11789
42.73165	4.01274	1884.02159	2169.01648	7.20985	325.98740	1.10925
40.50620	3.66981	1754.78773	2003.77791	7.07889	317.81176	1.10121
38.35649	3.34365	1629.05989	1846.19807	6.94711	309.39250	1.09355
36.26896	3.03326	1507.27923	1695.90719	6.81097	300.84120	1.08607
34.23671	2.73851	1389.97209	1552.92421	6.66763	292.26288	1.07866
32.25569	2.45954	1277.59143	1417.37859	6.51472	283.75006	1.07123
30.31699	2.19574	1170.14955	1289.00522	6.34954	275.35854	1.06373
28.41674	1.94714	1067.83532	1167.83893	6.16969	267.15941	1.05611
26.55024	1.71360	970.72504	1053.82796	5.97240	259.22498	1.04835
24.71504	1.49519	878.95315	947.01327	5.75482	251.64568	1.04046
22.90702	1.29175	828.30064	847.31589	5.51339	244.52903	1.03241
21.12306	1.10318	741.49014	754.69311	5.24410	238.02107	1.02423
19.36235	0.92961	660.12239	669.19100	4.94278	232.32807	1.01591
17.61662	0.77041	584.18619	590.47820	4.60339	227.71588	1.00469
15.89425	0.62660	514.09168	518.97395	4.22586	224.46752	1.00469
14.18004	0.49893	454.52025	454.52025	3.84197	220.84794	1.00469
12.48507	0.38588	396.90858	396.90858	3.40272	220.05712	1.00469
10.79607	0.28666	345.45598	345.45598	2.90430	223.39018	1.00469
9.12105	0.20185	299.89694	299.89694	2.35575	233.07362	1.00469
7.45090	0.13105	258.76663	258.76663	1.77258	253.39118	1.00469
5.78841	0.07447	218.90496	218.90496	1.19063	293.39638	1.00469
4.13196	0.07300	217.66517	217.66517	1.17385	212.60893	1.00469
2.47705	0.07300	217.66517	217.66517	1.17385	127.52665	1.00469
0.58843	0.07300	217.66517	217.66517	1.17385	30.30303	1.00469

Oblique Shock Discussion:

The downstream condition at each point is calculated using the equilibrium curve fit considering the oblique shock angle. To find the shock angle, scipy's root solver function

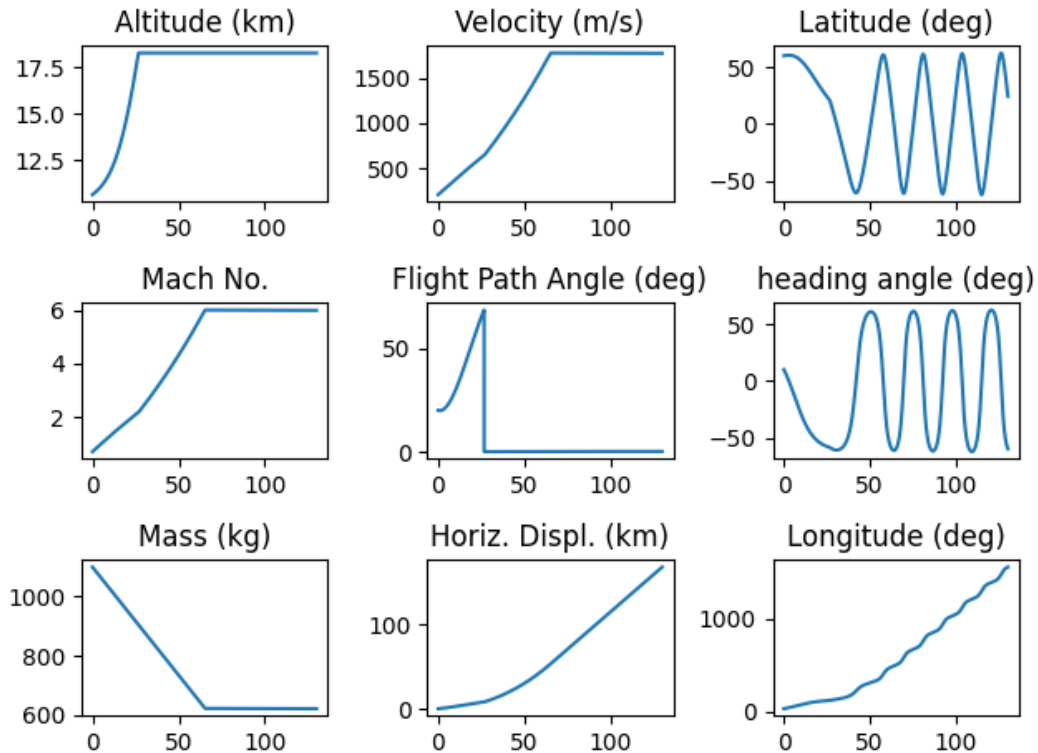
(<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html>) was used to find

the shock angle given the local body angle and the density ratio while updating the normal mach

number component downstream at each iteration until it reached a solution. These conditions

were computed at Mach 10, altitude 18.28 km.

## Flight Simulation



## Simulation Discussion

The 3-D, non-planar rotating equations of motion were used to calculate the main flight variables graphed above. The equations were time derivatives of positions and parameters of the vehicle relevant to flight performance, so to get updated values integration was done by accumulating the differences of each variable using Euler's method to loop until the accumulation of discretized time reaches or surpass the engine burnout time. To simulate elevator manipulation when reaching the target Mach 6 and 60,000 ft altitude, the flight angle and its derivative was set to 0 degrees and the thrust offset angle also set to 0 for longitudinal, steady flight. An additional throttle parameter was also added to be able to reduce the thrust to keep a steady flight velocity in hopes to reach a farther horizontal displacement prior to burnout.

## Part 2

### Stagnation point heating

$$\dot{q}_{stag}'' = \boxed{745.863 \text{ W/cm}^2}$$

$$\text{Pr} = \boxed{0.769}$$

$$\mu_e = \boxed{8.769 \times 10^{-5} \text{ kg/m}^* \text{s}}$$

$$\rho_e = \boxed{0.917 \text{ kg/m}^3}$$

### Cone section (note body point 1 is the stagnation point)

$$T^* = \boxed{1525.440 \text{ K}} \quad H_r = \boxed{4569.371 \text{ kJ/kg}}$$

$$\text{Pr}^* = \boxed{0.886}$$

### Part 2 - Discussion:

If the oblique shock at each local body angle was calculated to find the heat rate instead, the heat flux would be smaller and decline faster along x-axis of the nose geometry. This is because the normal component of the Mach number is used in the conversion of flow properties across each oblique shock. As a result, there will only be proportional changes to downstream conditions due to less freestream kinetic energy conversion. The recovery enthalpy will decrease while the wall enthalpy stays constant. Other downstream conditions such as density, the Stanton number, and the Mach number will decrease compared to the normal shock assumption.



Body Point	$Re^* (x10^6)$	$C_f (x10^{-3})$	$C_H (x10^{-3})$	$\rho_e u_e C_H$ (kg/m <sup>2</sup> -s)	$\dot{q}''_{w,x}$ (W/cm <sup>2</sup> )
2	0.78375	5.54627	3.00686	1.04147	445.2354
3	1.38439	4.94977	2.68347	0.92946	397.3507
4	1.96393	4.61542	2.50221	0.86668	370.5106
5	2.52517	4.38913	2.37953	0.82419	352.3446
6	3.07044	4.22082	2.28828	0.79258	338.8328
7	3.6017	4.08823	2.2164	0.76768	328.1891
8	4.12055	3.97966	2.15753	0.7473	319.4733
9	4.62838	3.88822	2.10796	0.73013	312.1331
10	5.12642	3.80955	2.06531	0.71535	305.8179
11	5.61569	3.74073	2.028	0.70243	300.2929
12	6.09711	3.6797	1.99491	0.69097	295.3935
13	6.57153	3.62496	1.96524	0.68069	290.9996
14	7.03968	3.57541	1.93838	0.67139	287.022
15	7.50219	3.5302	1.91386	0.6629	283.3923
16	7.95972	3.48865	1.89134	0.6551	280.0568
17	8.4128	3.45023	1.87051	0.64788	276.9731
18	8.86193	3.41453	1.85116	0.64118	274.1069
19	9.30762	3.38119	1.83308	0.63492	271.4301
20	9.75031	3.34991	1.81612	0.62904	268.9193
21	10.19041	3.32046	1.80016	0.62351	266.5553
22	10.62834	3.29263	1.78507	0.61829	264.3216
23	11.06449	3.26626	1.77077	0.61334	262.2041
24	11.49921	3.24118	1.75718	0.60863	260.1909
25	11.93289	3.21727	1.74421	0.60414	258.2716
26	12.36587	3.19442	1.73182	0.59985	256.437
27	12.97326	3.16393	1.7153	0.59412	253.9896

### Software Instructions:

The script ran on Python 3.9.13 using:

Numpy: 1.23.4

Scipy: 1.9.2

Matplotlib: 3.6.0

Python can either be installed using an alone installer or Anaconda (see docs:

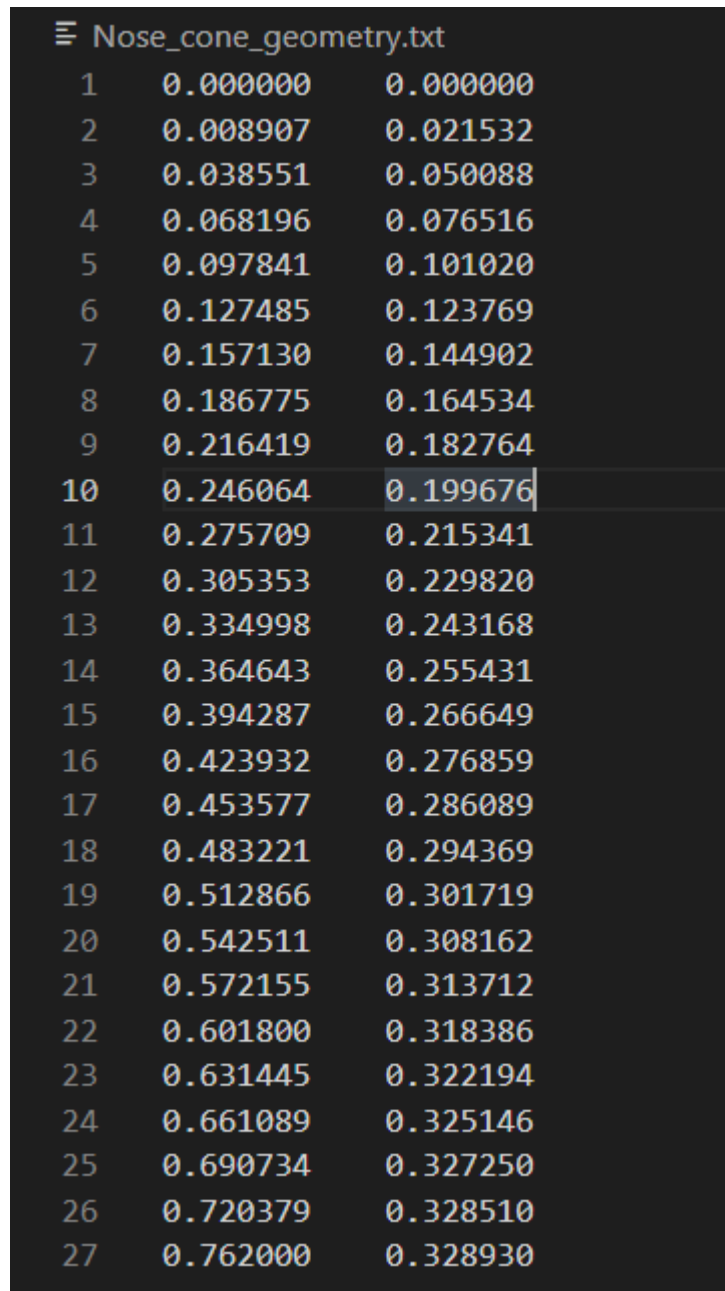
<https://docs.anaconda.com/anaconda/install/index.html>) and the packages were installed using

Pythons package manager, pip (pip should come automatically, if not follow the instructions:

<https://pip.pypa.io/en/stable/installation/>). The pip install command is: “pip install {name of package}” or for a specific package version: “pip install {name of package}=={specific version number}”.

To run the actual scripts, go to the file ‘hyperdesign.py” and input the desired run code for the variable “\_\_RUN\_\_”, then run in your favorite IDE/editor. Make sure all the python files that came along with it are in the same file.

Another detail to note: in the accompanying nose geometry file, the first row labeled “X Y” were removed for simplicity. If you are having trouble, try remove that row and DO NOT leave an empty space in the first row, the script may read it in as a value. Other than that, the coordinates should be the same. The end results are shown below opened in Visual Studio Code.



	X	Y
1	0.000000	0.000000
2	0.008907	0.021532
3	0.038551	0.050088
4	0.068196	0.076516
5	0.097841	0.101020
6	0.127485	0.123769
7	0.157130	0.144902
8	0.186775	0.164534
9	0.216419	0.182764
10	0.246064	0.199676
11	0.275709	0.215341
12	0.305353	0.229820
13	0.334998	0.243168
14	0.364643	0.255431
15	0.394287	0.266649
16	0.423932	0.276859
17	0.453577	0.286089
18	0.483221	0.294369
19	0.512866	0.301719
20	0.542511	0.308162
21	0.572155	0.313712
22	0.601800	0.318386
23	0.631445	0.322194
24	0.661089	0.325146
25	0.690734	0.327250
26	0.720379	0.328510
27	0.762000	0.328930