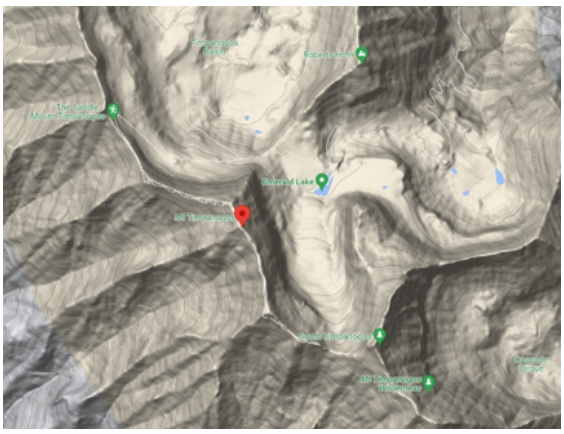
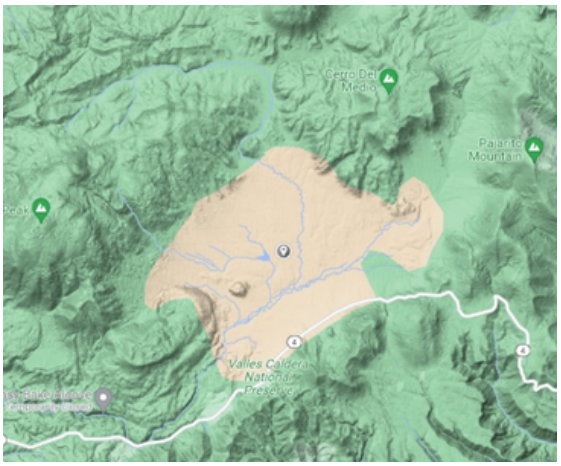
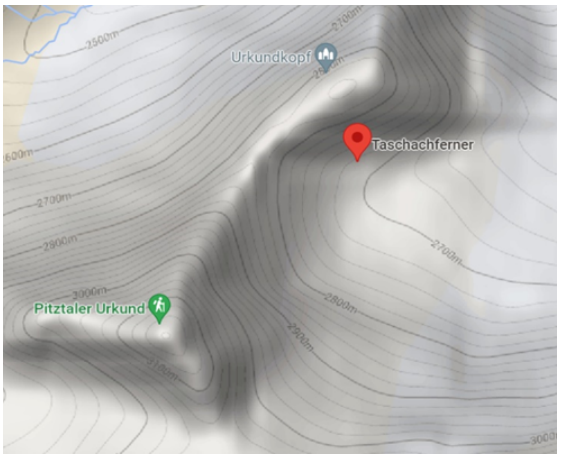
Optimization HW 1 Memo

Daniel Cheney

Familiarization with Contours and Constraints

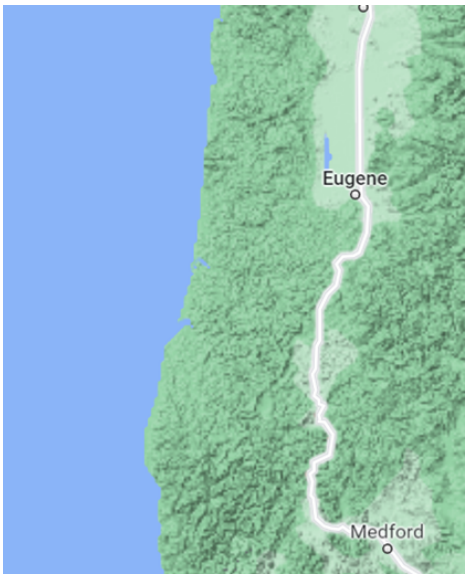
i. Mountain Peak
   1. Mt. Timp (40.39116, -111.64584)
   2. 
ii. Mountain Basin
   1. Valle Caldera, NM (35.864553, -106.4817)
   2. 
iii. Long Valley
   1. Grand Canyon (36.430531, -111.860550)

2. 

iv. Saddle Point
   1. Taschachferner, Austria (46.90033, 10.81641)
   2. 

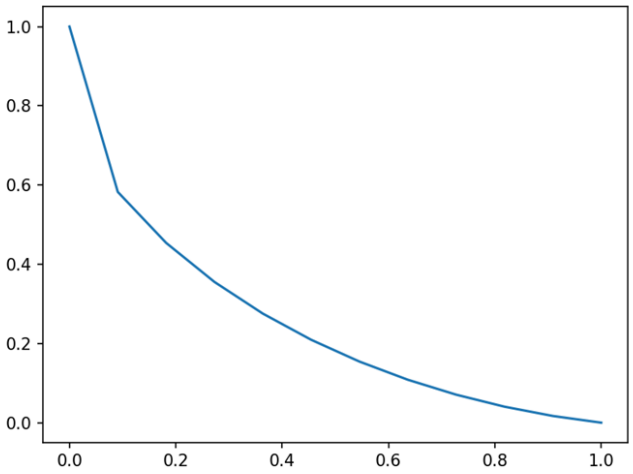
v. Equality Constraint
   1. I-15 (39.37256, -112.07602)
   2. 

vi. Inequality Constraint
   1. Oregon Coast (44.50241, -124.11033)

2. 

Unconstrained Brachistochrone Problem

The final Path for n = 12 discretization points (10 segments) is visualized in the plot to the left.



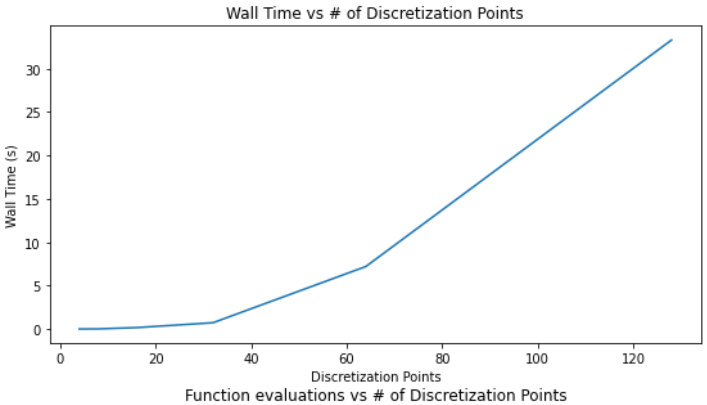
The total travel time for this path was **0.631 s.** I broke the objective function into two functions. One, was **delta_t** that found the time to travel a given segment (without gravity). The outer function used a for loop to go through the entire curve and found the total travel time for the curve. I used the argos keyword in the scipy.optimize.minimize function to pass in the x vector. This was because when testing different segment lengths this made it much easier to change the x vector.

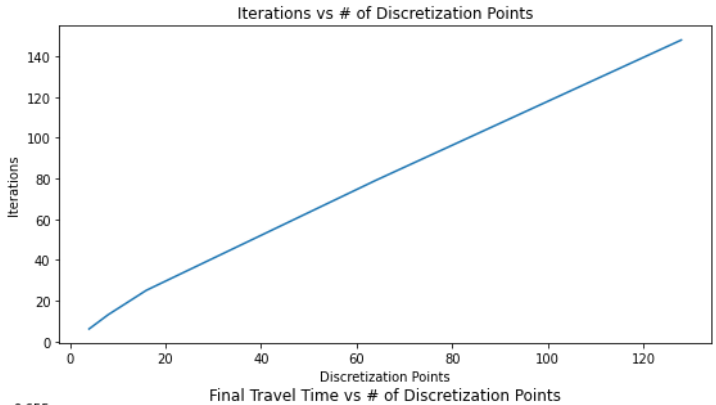
The effects of increasing the problem dimensionality were exponential. The statistics for 4, 8, 16, 32, 64, and 128 discretization points are shown below. The last plot shows the travel time on the curve which does decrease as the number of design variables increases.

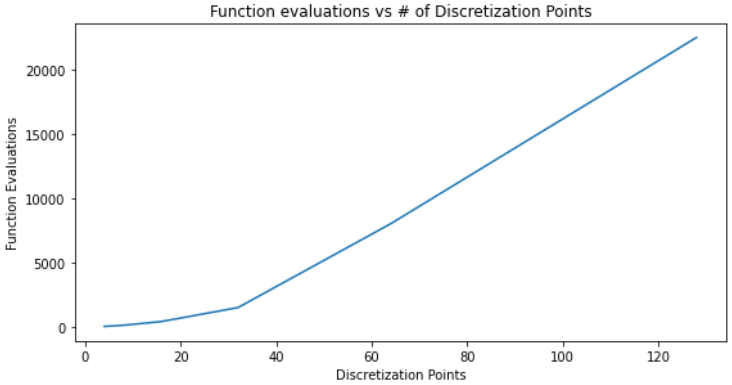
I tried the following methods to implement a warm start:

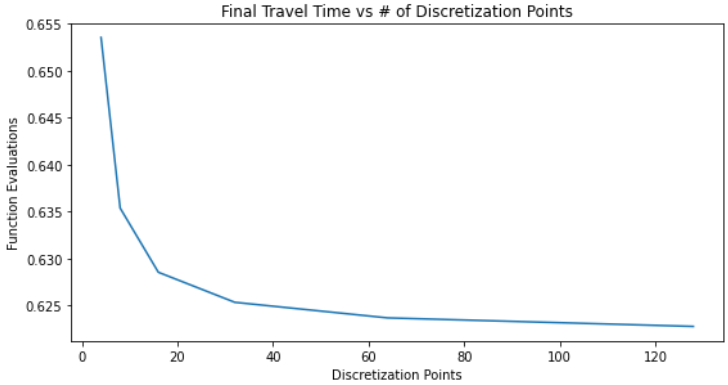
1. Interpolating from the smaller result vector from the previous trial to get a larger (from n=8 to n=16) y vector for the initial guess
2. Creating a straight line from the initial to final points (not from previous trials, but closer than a vector of zeros)

Neither of these methods seemed to give better results than a vector of zeros in terms of computation speed or curve performance, so the plots below are all from a cold start. This is likely due to user error. In all statistics, the computational requirement was exponential as the design variables doubled.







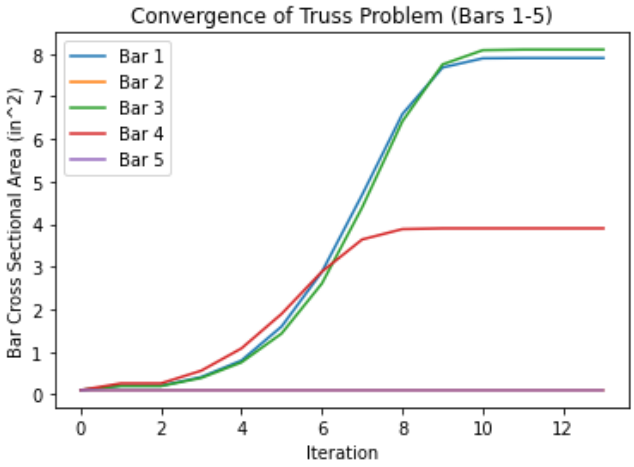


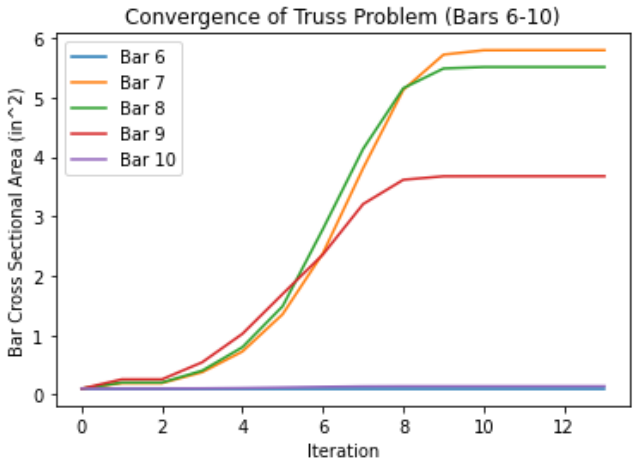
Constrained Truss Problem

Optimal Mass: 1497.6 lbs

Optimal Cross Sections: [7.9, 0.1, 8.1, 3.9, 0.1, 0.1, 5.798, 5.515, 3.677, 0.141]

Required Function Calls (truss): 146





Above are two plots showing the convergence of each bar to its final, optimized cross-sectional area. This optimization did not take as long as I thought it would, and the optimizer chose to leave certain bars at the initial cross-sectional area (0.1 in^2) while making others significantly larger. This could be avoided by having some sort of upper bound on the size of the cross-section.

My approach for this problem was to have one bound, where the design variables all had to be at least 0.1 in^2 in cross section. Then I added a constraint where all of the final yield and compressive stresses were less than the max given in the problem statement. The objective function return the mass from the truss function in order to minimize it.