Dido's Problem - Matlab Optimization Solution

Using Matlab and Matlab's constrained optimization function, fmincon, implement an approximation of Dido's Problem to find an approximate solution using finite dimensional optimization/nonlinear programming.

Dido's problem is concerned with determining the optimal geometric curvature of a string of fixed length that ties off at both ends and produces the maximum area within the closed loop.

Formulating a solution for Dido's Problem using optimization techniques with MATLAB (code is included in the appendix) first began by establishing the problem parameters. The total fixed length of string L was specified as 10 (arbitrary units) and kept constant through all iterations. In each iteration, a certain number of line (string) segments was chosen according to the variable N from 2 to 50, and the total resulting perimeter is calculated as the sum of all line segments subject to the equality constraint that the perimeter should equal the fixed line length L.

A more involved process was required to formulate an appropriate input variable (x) that would be provided to the fmincon function to be optimized from an initial guess. Although cartesian coordinates for X and Y could be used within a single vector (as shown in the lecture), the approach taken in this solution used an input vector R representing the distance from the origin to a point (X,Y) in the 2D plane at divided segments corresponding to an angle (theta) between 0 and 90 degrees. This decision was made so that the minimization function for the area (DidoArea) and equality constraint function (DidoPerimeter) would require only a single parameter: R.

The initial guess provided to the optimization process was generated by randomly assigning a value for each element R(i) at every angle *theta* for i=(N+1) angles, with a lower bound of L/4 and an upper bound that produces an R value who's X component is not greater than the X component of the preceding value. This ensured that for every X, there could only be a single corresponding Y value. An example for N = 3 is shown in the following figure, where the random length of element R(i=3) is bounded below by L/4 and above by (L/4 + M).

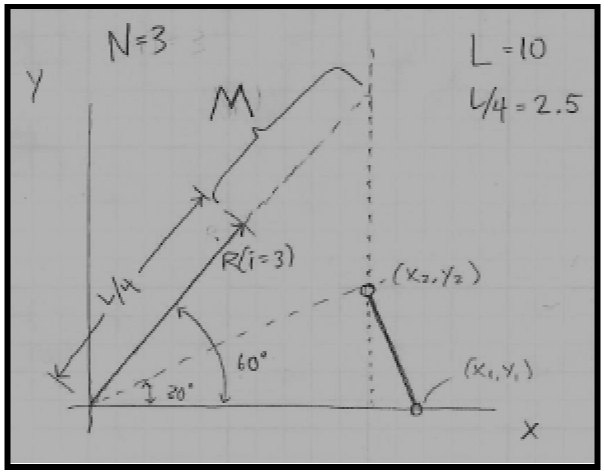
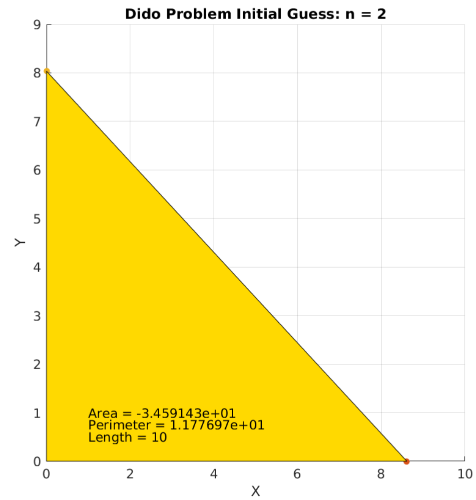
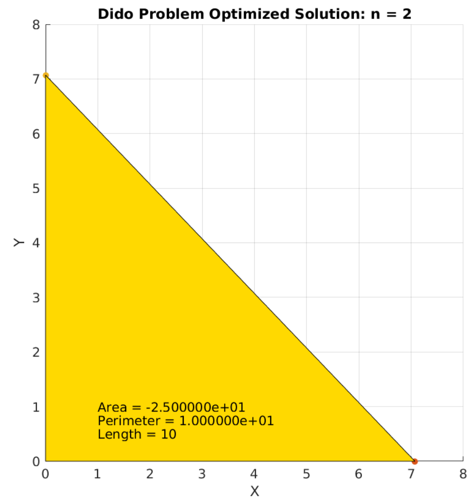
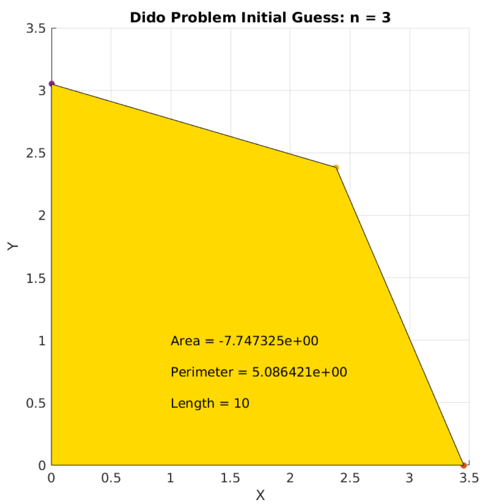


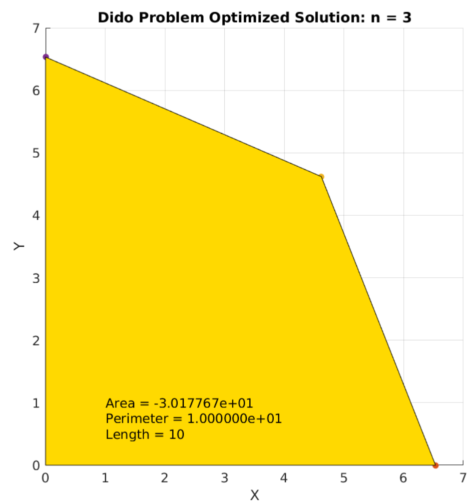
Figure: Generating an Initial Guess for R(i=3) with N=3 segments

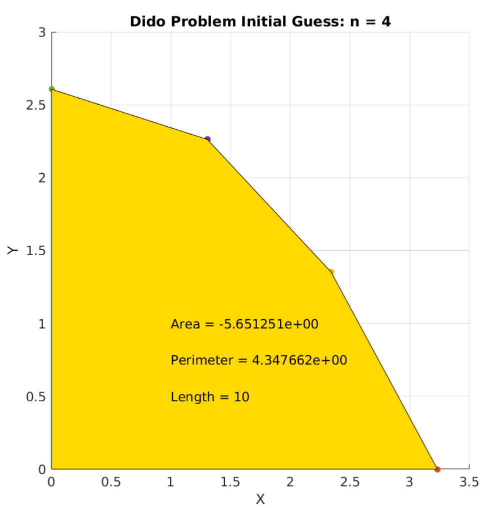
The resulting plots for each iteration of N segments for [N = 2,3,...,9,10, 25, 50] follow, where the left plot of each shows the (random) initial guess, and the plot on the right is the optimized solution for vector R. The solution (an equal-sided polygon) is optimal for all values of N up to ~50 segments, where the equality constraint for the total perimeter was no longer satisfied. This is possibly due to a rounding error of the floating point value for each angle of theta between 0 and 90 degrees.

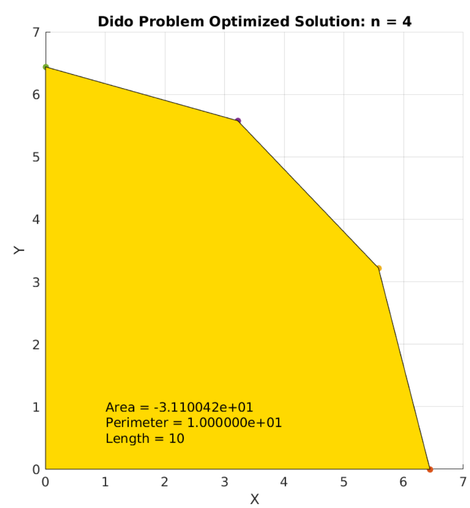

Dido Problem Initial Guess: n = 2
Area = -3.459143e+01
Perimeter = 1.177697e+01
Length = 10
X
Y

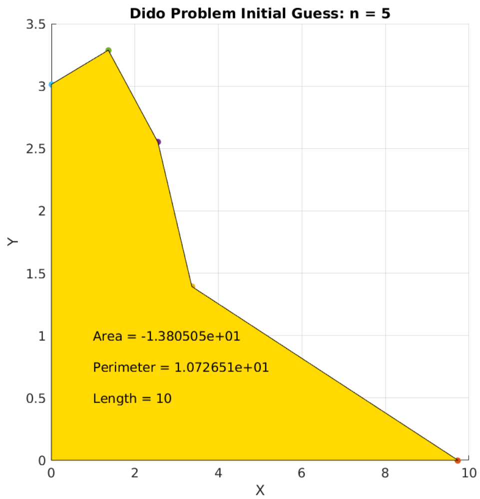

Dido Problem Optimized Solution: n = 2
Area = -2.500000e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

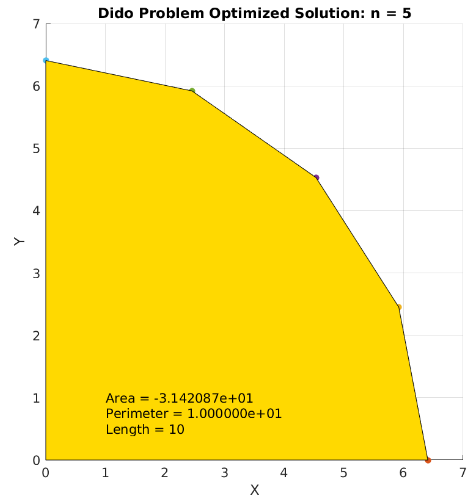

Dido Problem Initial Guess: n = 3
Area = -7.747325e+00
Perimeter = 5.086421e+00
Length = 10
X
Y

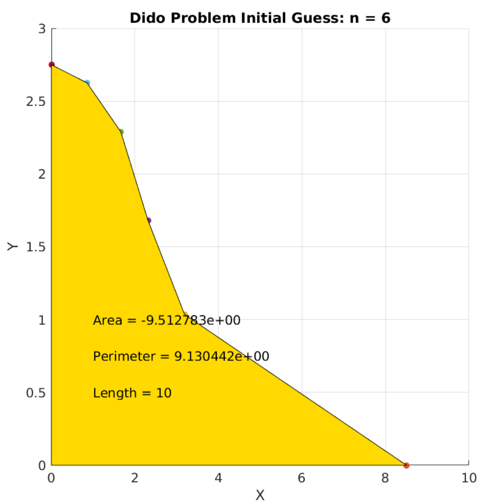

Dido Problem Optimized Solution: n = 3
Area = -3.017767e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

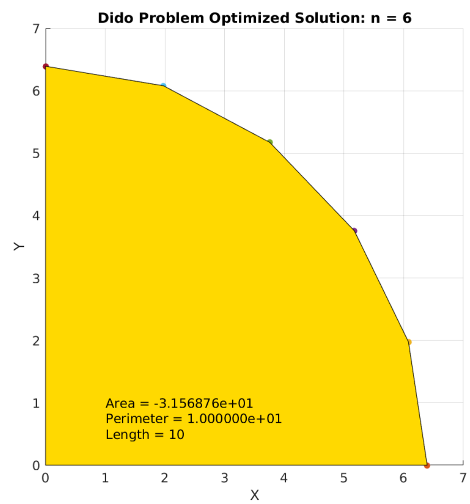

Dido Problem Initial Guess: n = 4
Area = -5.651251e+00
Perimeter = 4.347662e+00
Length = 10
X
Y

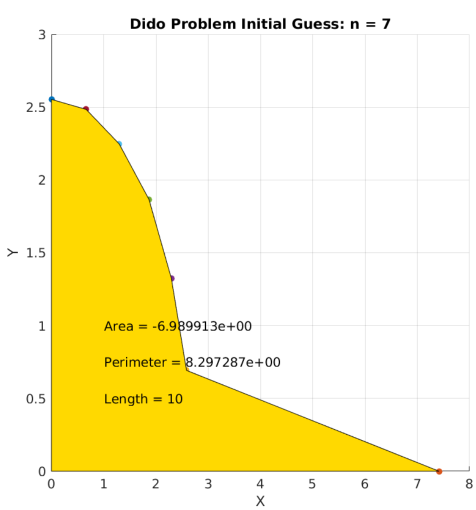

Dido Problem Optimized Solution: n = 4
Area = -3.110042e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

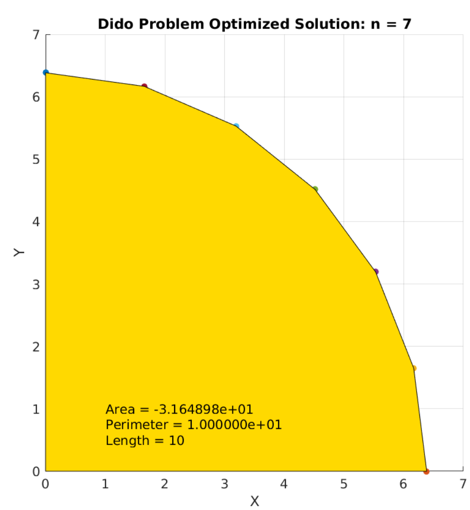

Dido Problem Initial Guess: n = 5
Area = -1.380505e+01
Perimeter = 1.072651e+01
Length = 10
X
Y

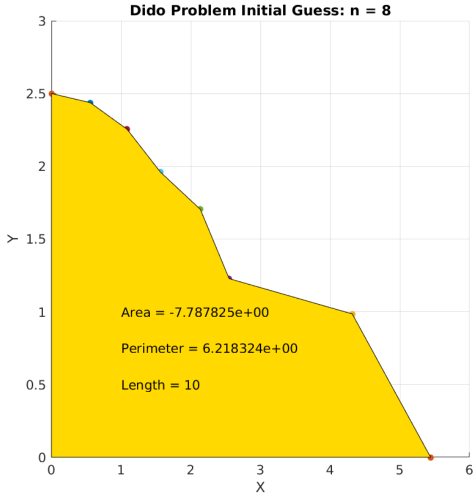

Dido Problem Optimized Solution: n = 5
Area = -3.142087e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

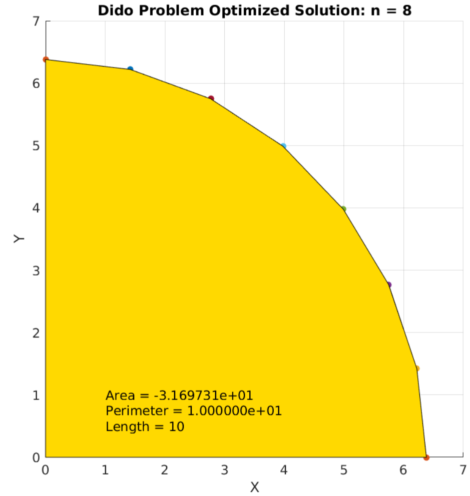

Dido Problem Initial Guess: n = 6
Area = -9.512783e+00
Perimeter = 9.130442e+00
Length = 10
X
Y

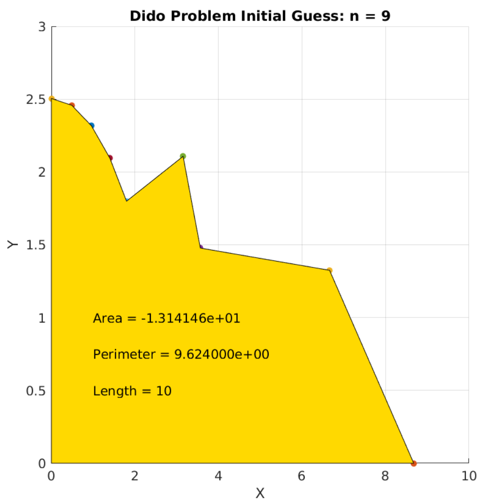

Dido Problem Optimized Solution: n = 6
Area = -3.156876e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

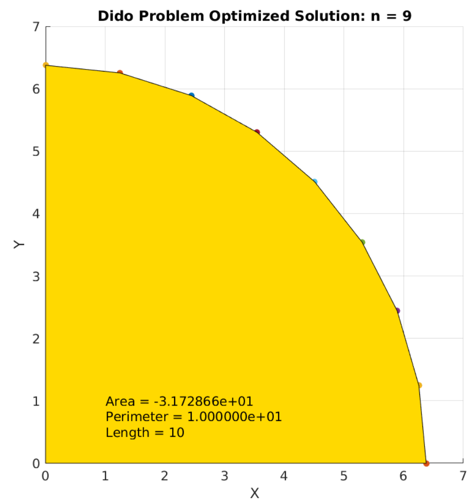

Dido Problem Initial Guess: n = 7
Area = -6.989913e+00
Perimeter = 8.297287e+00
Length = 10
X
Y

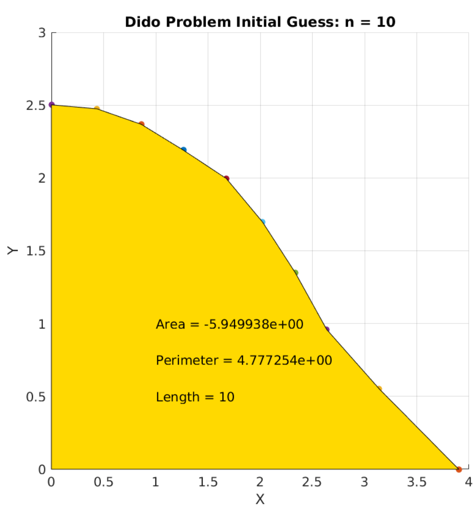

Dido Problem Optimized Solution: n = 7
Area = -3.164898e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

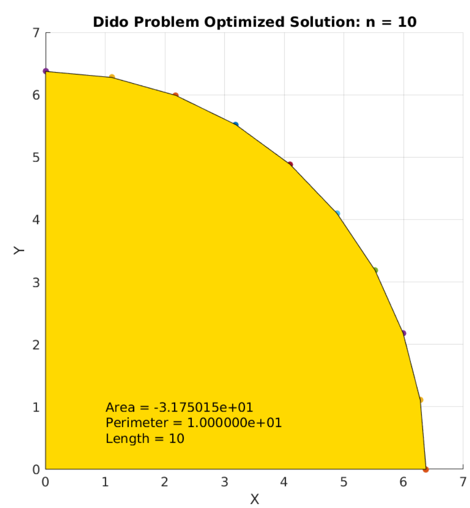

Dido Problem Initial Guess: n = 8
Area = -7.787825e+00
Perimeter = 6.218324e+00
Length = 10
X
Y

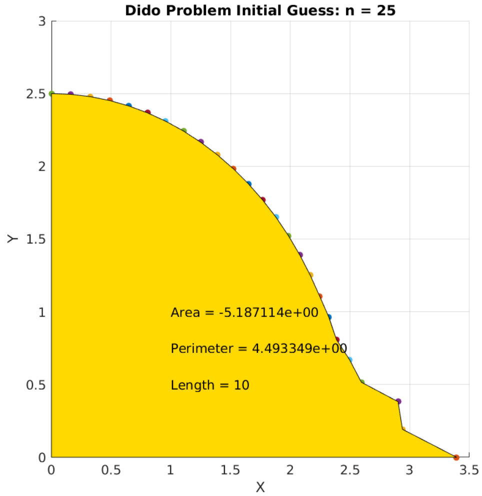

Dido Problem Optimized Solution: n = 8
Area = -3.169731e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

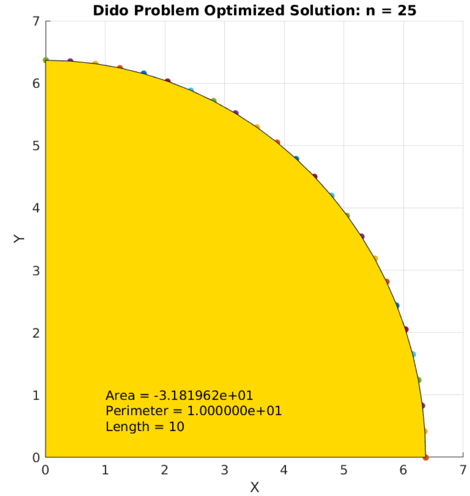

Dido Problem Initial Guess: n = 9
Area = -1.314146e+01
Perimeter = 9.624000e+00
Length = 10
X
Y

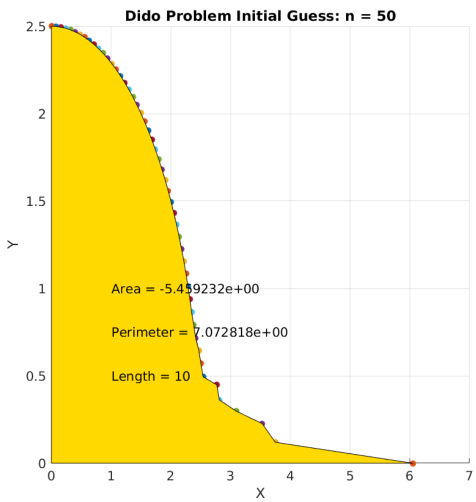

Dido Problem Optimized Solution: n = 9
Area = -3.172866e+01
Perimeter = 1.000000e+01
Length = 10
X
Y

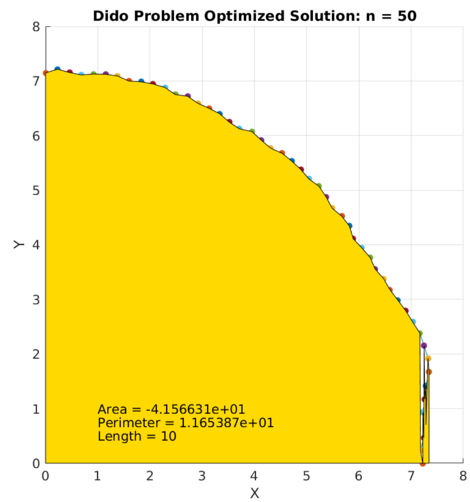

Dido Problem Initial Guess: n = 10
Area = -5.949938e+00
Perimeter = 4.777254e+00
Length = 10
X
Y


Dido Problem Optimized Solution: n = 10
Area = -3.175015e+01
Perimeter = 1.000000e+01
Length = 10
X
Y


Dido Problem Initial Guess: n = 25
Area = -5.187114e+00
Perimeter = 4.493349e+00
Length = 10
X
Y


Dido Problem Optimized Solution: n = 25
Area = -3.181962e+01
Perimeter = 1.000000e+01
Length = 10
X
Y


Dido Problem Initial Guess: n = 50
Area = -5.459232e+00
Perimeter = 7.072818e+00
Length = 10
X
Y


Dido Problem Optimized Solution: n = 50
Area = -4.156631e+01
Perimeter = 1.165387e+01
Length = 10
X
Y