Numerical Analysis of the Moving Sofa Problem

Forrest Pratson
April 17, 2020

Duke University Math 260
Professor Eric Autry

Abstract

This paper is a numerical analysis of the moving sofa problem. It is meant to accompany the code used in attempt to solve the problem. We used Bayesian Optimization to fit parameters to find the largest sofa to fit around a 90 degree hallway turn. Our best area calculated was 2.185, this is slightly less than Gerver's sofa area of 2.219.

1 Definition of Problem

The moving sofa problem was first proposed in the early 20th century. It is a 2 dimensional analogy to the real problem of the largest rigid body that can be moved around an L shaped hallway of unit width. The area of the sofa is thus measured in relation to the hallway's width.

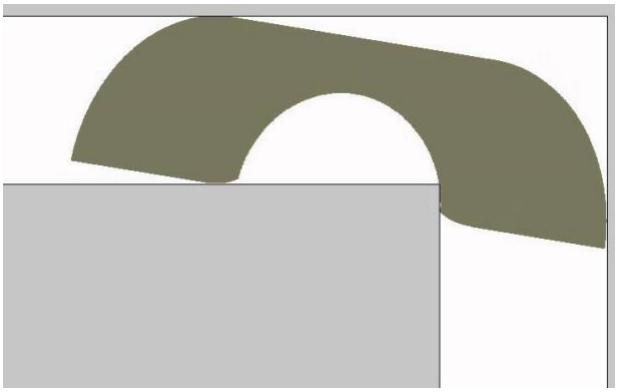

Figure 1: Example of a sofa that solves the proposed problem

John Hammersley made the first significant progress in the problem by deriving a lower bound of $\frac{2}{\pi}+\frac{\pi}{2}$, and an upper bound of $2\sqrt{2}$. However, the best shape found to date was derived by Joseph Gerver using a system of 18 different curves corresponding to differential equations. The area of Gerver's sofa is roughly 2.2195 (known as Gerver's constant) and cannot be expressed in closed form. A numerical computation by Phillip Gibbs produced a sofa identical to Gerver's sofa out to 8 significant digits. This is evidence that Gerver's sofa is likely the largest sofa. However, the problem has never been proven.

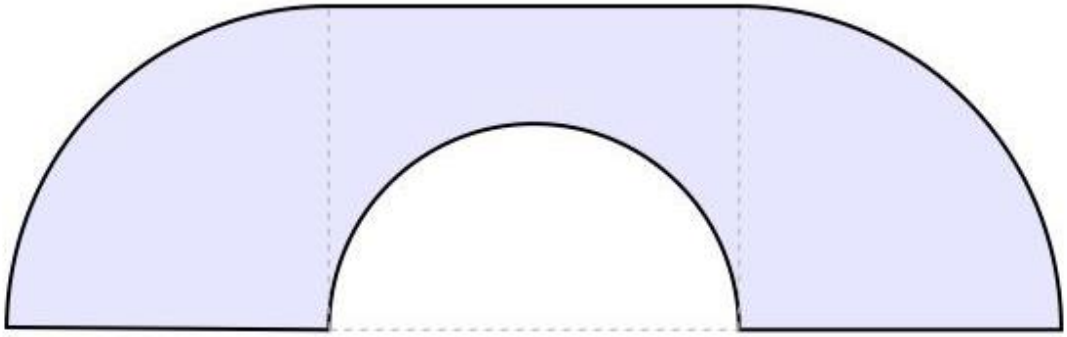

Figure 2: Hammersley's sofa

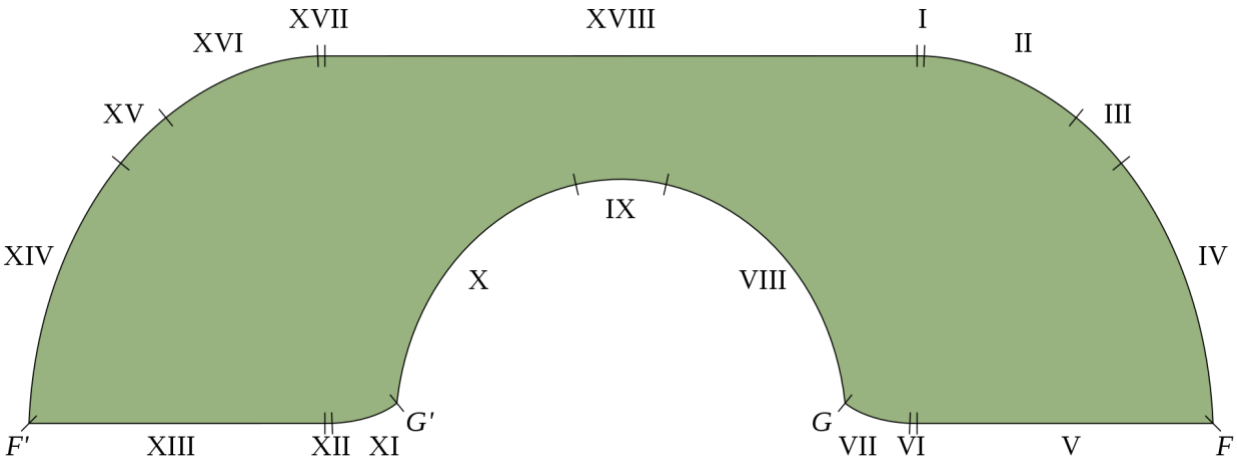

Figure 3: Gerver's sofa with 18 separate curves

2 Methods

We began by noticing that we could derive the largest sofa by finding the path of the L shaped hallway that left the largest cavity when the hallway was rotated 180 degrees. We began by defining the points of on the Hallway. We used the equations derived by user 'newzed' on Mathematics Stack Exchange:

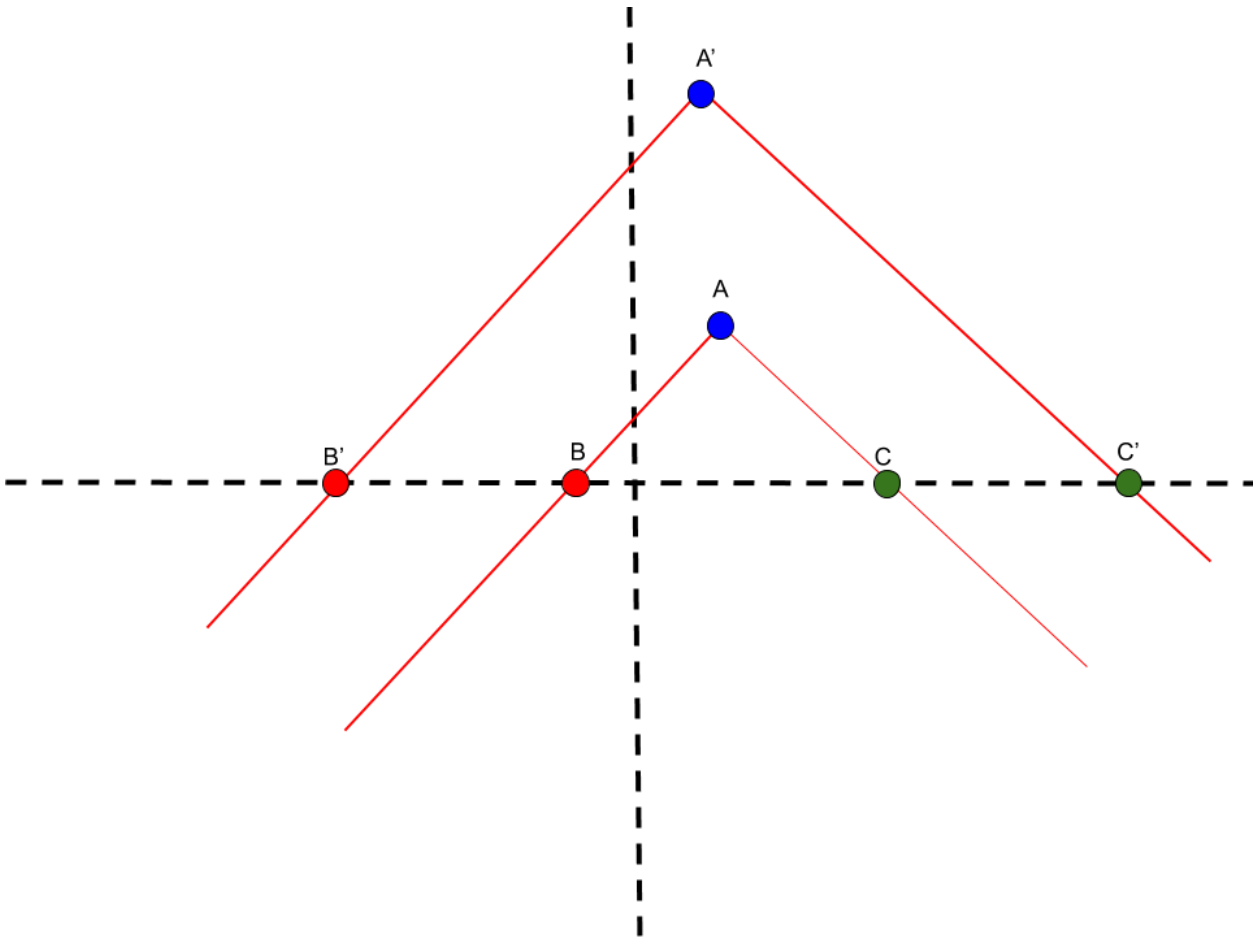

Figure 4: Definition of Hallway

$$A = < r\ cos(\alpha), t\ sin(\alpha) > \quad A' = < r\ cos(\alpha) + \sqrt{2}\cos\left(\frac{\pi}{4} + \frac{\alpha}{2}\right), t\ sin(\alpha) + \sqrt{2}\sin\left(\frac{\pi}{4} + \frac{\alpha}{2}\right) >$$

$$B' = < r\ cos(\alpha) - \frac{t\sin(\alpha)}{\tan\left(\frac{\alpha}{2}\right)}, 0 > \qquad B = < r\ cos(\alpha) - \frac{t\sin(\alpha)}{\tan\left(\frac{\alpha}{2}\right)} - \frac{1}{\sin\left(\frac{\alpha}{2}\right)}, 0 >$$

$$C = < r\ cos(\alpha) + t\sin(\alpha)\tan\left(\frac{\alpha}{2}\right), 0 > \qquad C' = < r\ cos(\alpha) + t\sin(\alpha)\tan\left(\frac{\alpha}{2}\right) + \frac{1}{\cos\left(\frac{\alpha}{2}\right)}, 0 >$$

In simple terms r is how far point A is initially from the origin, and t is how far point A is from the origin when alpha is at 90 degrees. Alpha goes from 0 to 180 degrees in the problem.

3 Calculation of Area

My code created a canvas of width 6 units centered at the origin as well as a height of 2.5 units starting from the y axis. From playing around with simulations, I found this was a good canvas size to encapsulate all possible sofa shapes. The code calculates all of the points defined above, and creates a polygon of the hallway shape. The code then passes the canvas to the polygon and calculates if each point on the canvas is within the polygon. If it is not, the point is removed from the canvas. The code then increases alpha by an increment, and repeats the calculation. The

number of points left on the canvas after alpha goes from 0 to 180 degrees is proportional to the area of the sofa.

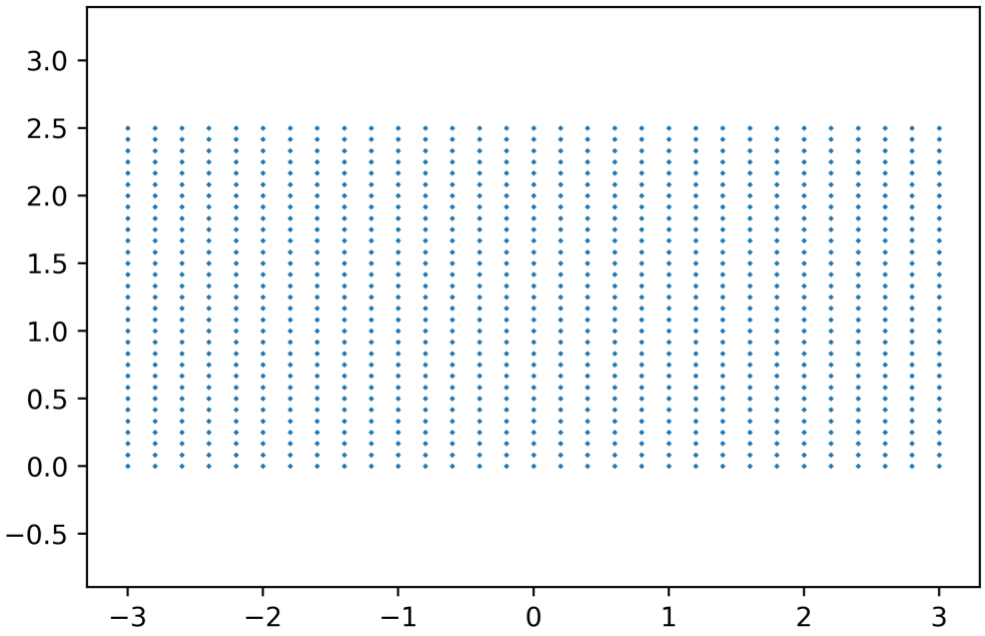

Figure 5: Example Sparse Canvas

My code, given a value of r and t, would use this method to calculate the area of the sofa.

4 Optimization

In order to get reliable results, our code needed a large canvas size. This resulted in a long computation time. Thus, we elected to use Bayesian optimization to find the maximum sofa area. Bayesian optimization is best used when the computation of area is expensive to calculate, or the results are noisy.

Bayesian optimization works in the following way. Given some objective function, the algorithm starts by initializing a gaussian process model of the function with some initial set of points. For our calculation, we found initializing the model with 25 random points was a good start. Each point being a value of r and a value of t spanning between 0 and 1. The program then generates 10,000 random points for possible testing, it uses the gaussian process model from initialization to calculate the predicted value that sample would have, as well as the standard deviation. We then used Bayesian conditional probability to calculate which of the samples had the highest probability of generating a higher function value than the current best value using the following formula

$$P = \Phi\left(\frac{\mu - best}{\sigma}\right)$$

Where Φ is the normal continuous distribution function, mu is the sample's prediction, best is the highest function value so far, and sigma is the sample's standard deviation. The algorithm then uses the sample that has the highest probability of being greater, and calculates its value using the objective function. It then refits the gaussian process model with the calculated point

included and repeats the whole process over again. Using this method, the algorithm can usually converge to the maximum value of a function within less than 100 computations of the objective function. This highlights its benefit to our project, where each computation of area takes a little more than a second. Conventional optimization methods that require thousands of computations would take hours to run when this takes a couple minutes depending on your computer.

5 Results

Using the methods outlined we got the following results. We began by looking at the error associated with our calculation. We plotted the absolute error of our calculation against Hammersley's sofa (r=.5,t=.5) as a function of canvas resolution. Looking at the plot, we found that a canvas of resolution of 500 gave acceptable error while still calculating within a reasonable time.

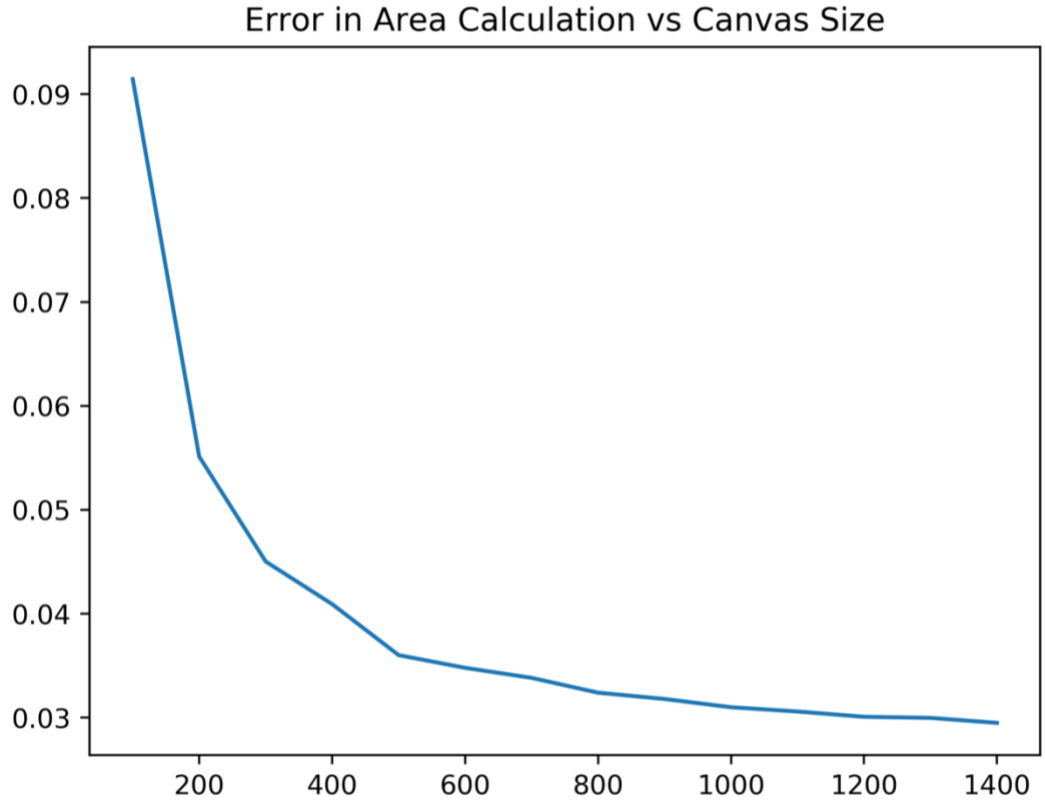

Figure 6: Error in calculation vs canvas resolution.
Note: the number of points on the canvas is 15 res^2.

Next, we used Bayesian optimization to find the sofa with the best area. Our results differed slightly depending on the run, but our best calculation was a sofa of area 2.185:

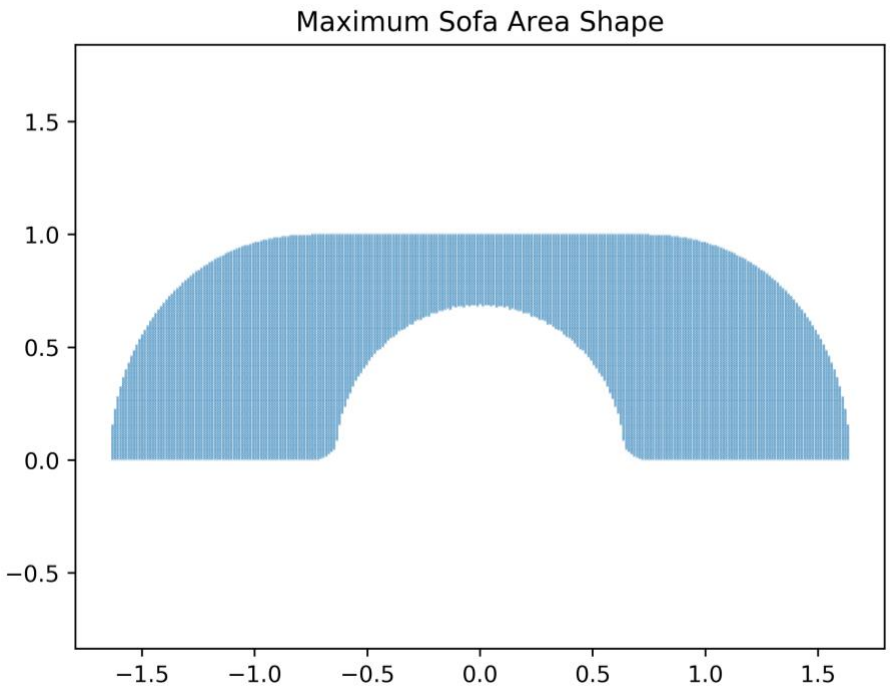

Figure 7: Visualization of optimized sofa shape.

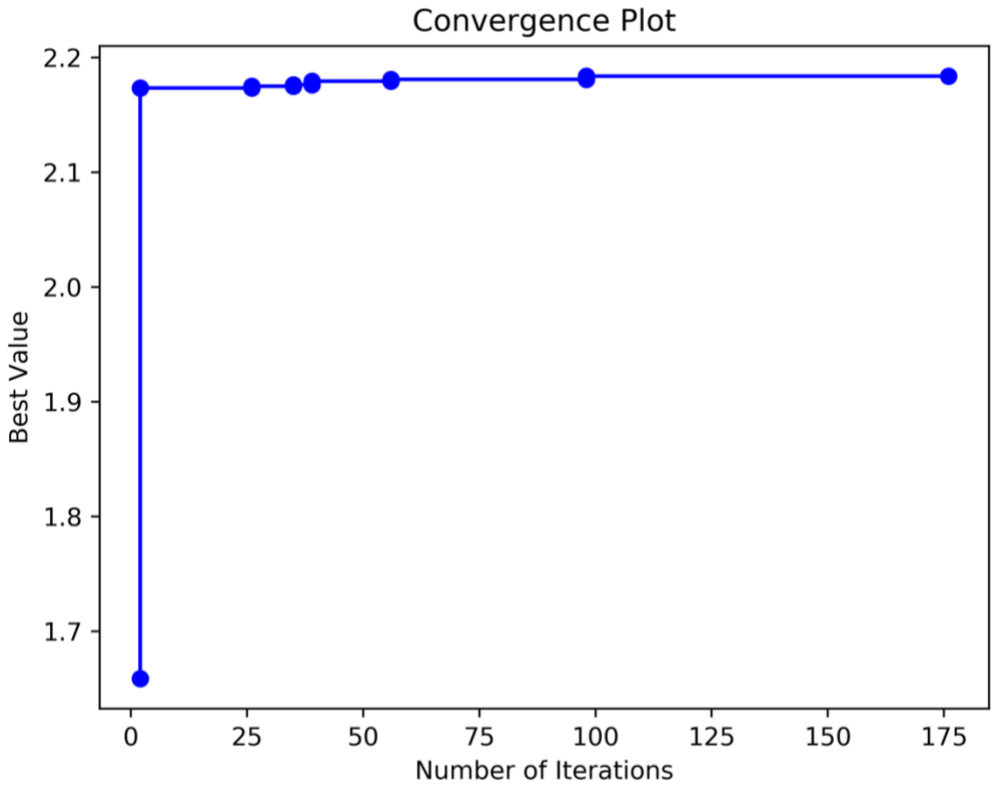

Figure 8: Convergence plot of Bayesian Optimization

Finally, we computed a contour plot of the area as a function of the values of r and t. For this, we used a much smaller canvas with only a total of 15,000 points (res=100) because so many calculations were needed.

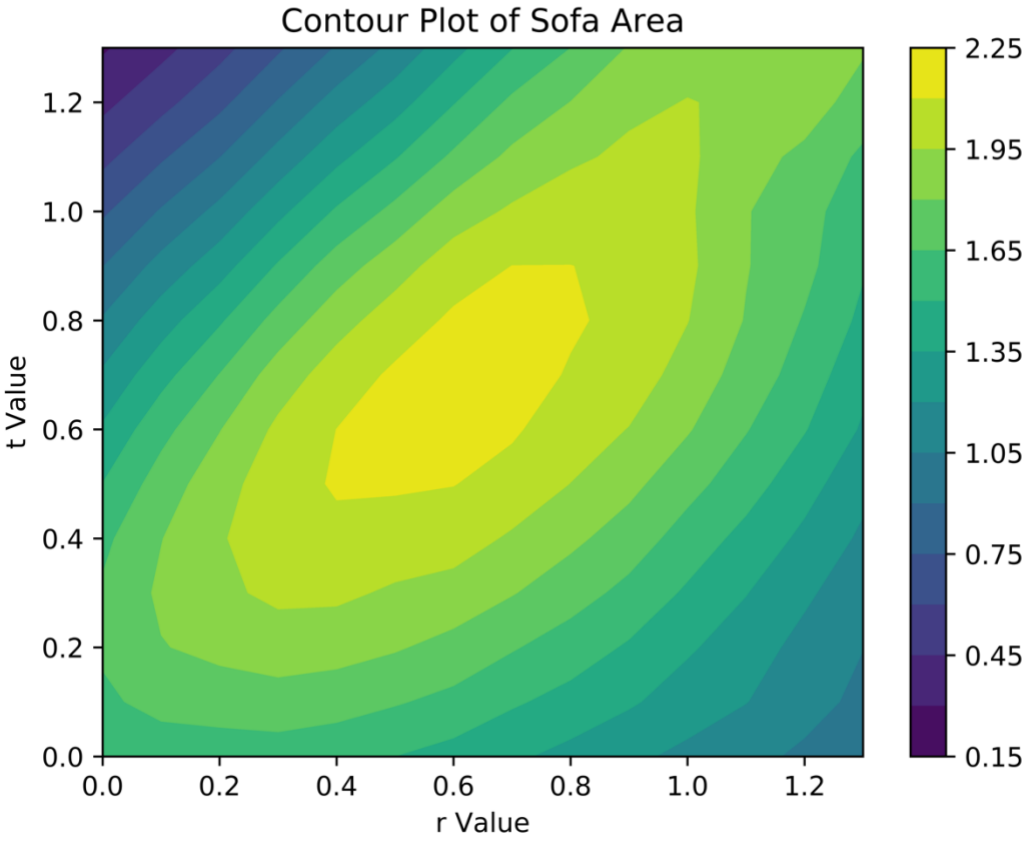

Figure 9: Contour plot of area vs parameters r and t