Optimization Assignment 1

Kim Paolo Laberinto

2021-02-xx

Contents

1 Q1. 1D Line Search on Rosenbrock Banana Function

1.1 Set-up

This section of the document discusses some different observations seen in 1D Line Searches on the Rosenbrock Banana Function. The equation for the Rosenbrock 2D Banana Function in shown in Eq. **??**.

$$f_{\text{rosenbrock banana}}(x, y) = (a - x)^2 + 100b(y - x^2)^2 \tag{1}$$

In particular for this report, the 2D objective function examined is the Rosenbrock Banana Function where $a = 1$ and $b = 0.1$. This particular 2D Rosenbrock Banana Function is used to investigate the differences and similarities between the following 1D Line Search Methods:

- Swann's Bracketing Method
- Powell's Bracketing Method

Fig. **??** shows Lines A, B, and C along with the contours of the Rosenbrock Banana Function chosen. Note that Lines A and B pass through the global minimum at $(x, y) = (1, 1)$. The following subsection discusses how Swann's and Powell's Bracketing Methods find a suitable bracketing interval on the 1D line.

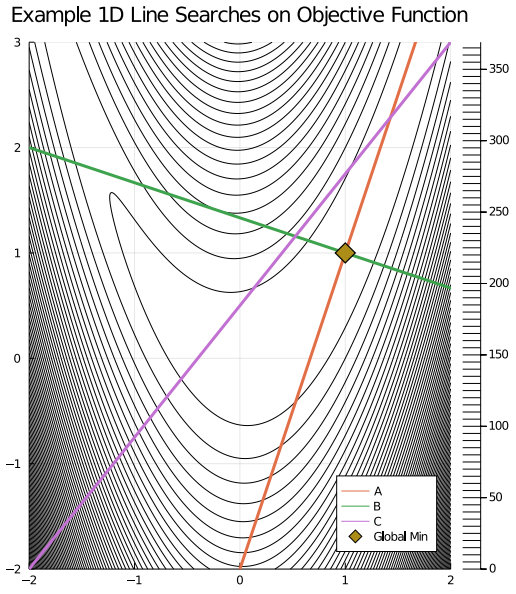


Figure 1: 1D Lines A, B, and C shown on top of the contours of the 2D Rosenbrock Banana Function.

1.2 Swann's vs Powell's for Initial Bracketing

In order to apply a 1D Line Search on an N-D objective function $f : \mathbb{R}^n \to \mathbb{R}$, a parametrization along a 1D line is necessary. This 1D parametrization can be done in the following way in Eq. **??**. An initial point $\vec{x_0} \in \mathbb{R}^n$ along with a direction $\vec{d} \in \mathbb{R}^n$ serve as a way to specify which 1D line to select. The $\alpha \in \mathbb{R}$ parameter then is used to specify where along the 1D line to sample from for the 1D Line Search Methods.

$$f_{1\text{D}} : \alpha \mapsto f\left(\vec{x_0} + \alpha\vec{d}\right) \tag{2}$$

With this parametrization, the 1D Line Search Methods can try various values of α to sample the objective function. The goal of the 1D Line Search is then to determine a suitable bracketing interval in terms of α_{lower} and α_{upper} wherein a minimum exists for the objective function along the 1D line.

Applying Swann's Bracketing Method and Powell's Bracketing Methods on Lines A, B, and C from Fig. **??** result initial bracketing intervals shown in Fig. **??**, Fig. **??**, Fig. **??**.

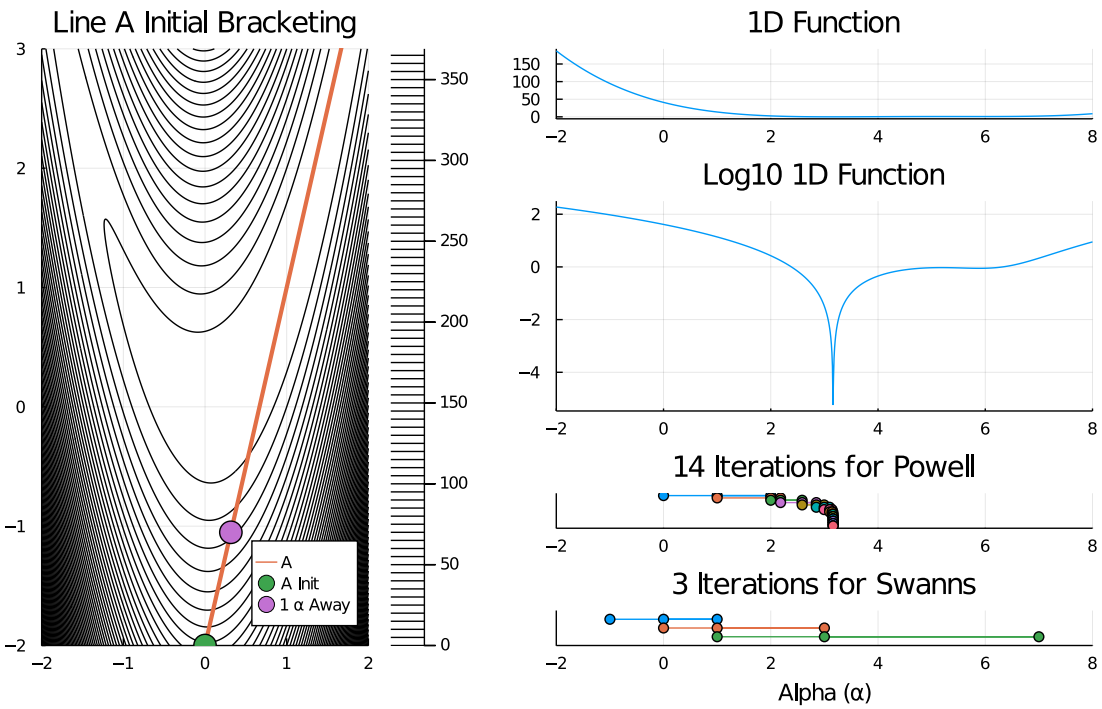


Figure 2: 1D Line Search Intervals on Line A found using Swann's and Powell's Bracketing Methods. Note that the location for global minimum $(1, 1)$ can be found on this line.

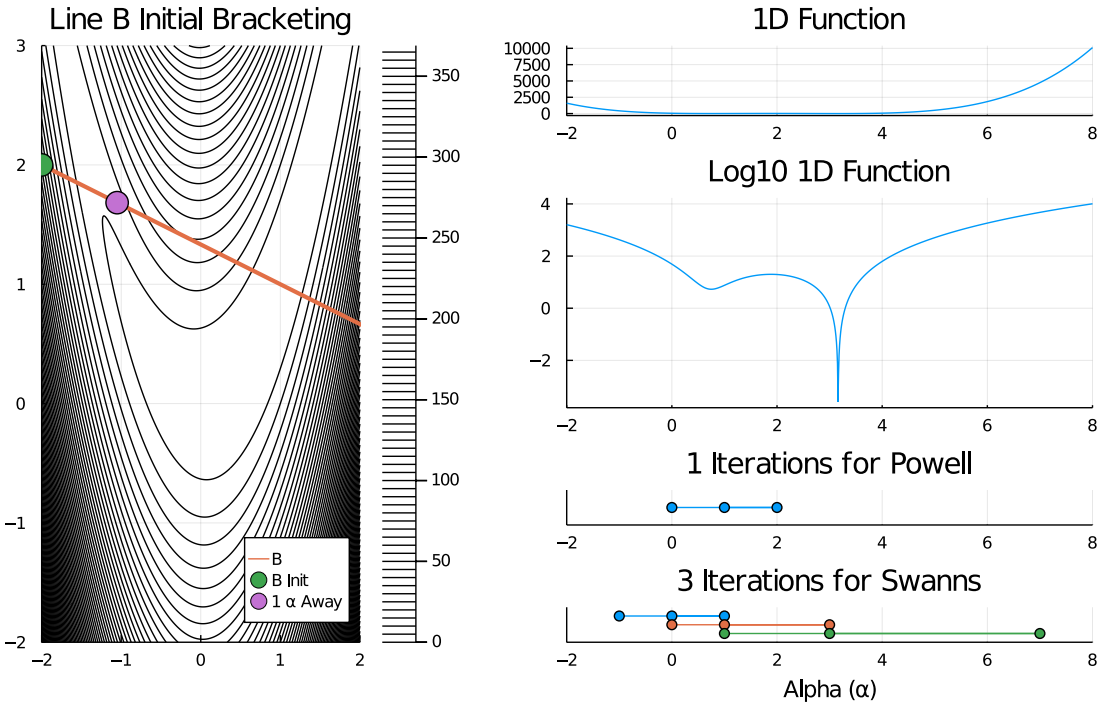


Figure 3: 1D Line Search Intervals on Line B found using Swann's and Powell's Bracketing Methods. Note that the location for global minimum $(1, 1)$ can be found on this line.

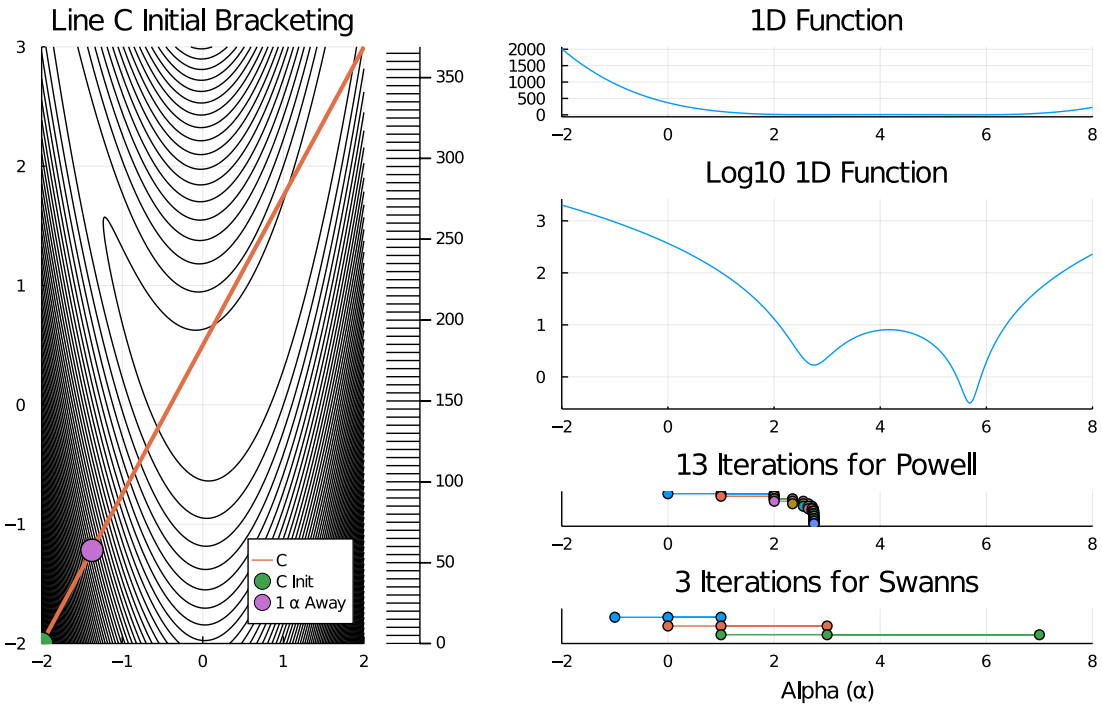


Figure 4: 1D Line Search Intervals on Line C found using Swann's and Powell's Bracketing Methods.

Fig. **??**, Fig. **??**, Fig. **??** all show the 1D line relative to the 2D contours of the Rosenbrock Banana function, as well as the function value along the 1D line in terms of α. Note that that the objective function value is also shown in terms of a log10 scale to help visualize the differences in function value along the 1D line.

There are several interesting observations to note from these results.

- As per the algorithm definition, the 1D Line Searches only stop once it finds a situable interval.
  - A suitable interval is such where the midpoint evaluated has a lower function value
    - An example of this can be seen in Fig. **??** for Line A, where the final iteration for Swann's has a middle evaluation point at $\alpha_{\text{middle}} = 3$ which has a lower function value than the interval endpoints $\alpha_{\text{lower}} = 1$ and $\alpha_{\text{upper}} = 7$
  - If a non-suitable interval is found, then a new iteration for the 1D search is done. Powell's Bracketing Method tests the next point using a quadratic fitting technique and testing the minimum of such a parabola (within bounds). Whereas Swann's Bracketing Method simply tests farther away with a magnification factor and step size.
    - This can be seen by the multiple intervals/iterations shown in the figures.
- For Lines A and B which contained the global minimum there were different results in the Bracketing Methods in whether or not the global minimum was captured.
  - For Line A both Powell's and Swann's Bracketing Intervals captured the global minimum because there was only one minimum in the nearby 1D function as seen on the log10 plot.
  - For Line B, only Swann's Bracketing Interval captured the global minimum inside its final interval. As can be seen for Powell's Bracketing Interval, only 1 interation was done as it formed a suitable interval.
- For Lines A and C, Powell's Bracketing Method made the intervals smaller and smaller. It seems like due to the curvature of the 1D function near the initial starting point, the fitted parabolas formed such that to only move a small amount further along the 1D function.

- For Lines B and C, the two different bracketing methods captured two different local minima.
  - As can be seen with Powell's Bracketing Method, due to the local function behavior near the starting point, only the first nearby minimum was captured by Powell's. Powell's missed the deeper minima.
  - Swann's Method was simple enough with its steps to capture the deeper minima in a large interval.

1.3 Golden Section Search after Swann/Powell

After finding the initial bracketing interval, a Golden Section Search is performed. Fig. **??** below shows how the Powell and Swann's initial bracketing intervals are made smaller to some tolerance level. Note that the figure shows the 1D function values on a log10 scale to help to see the local function behavior.

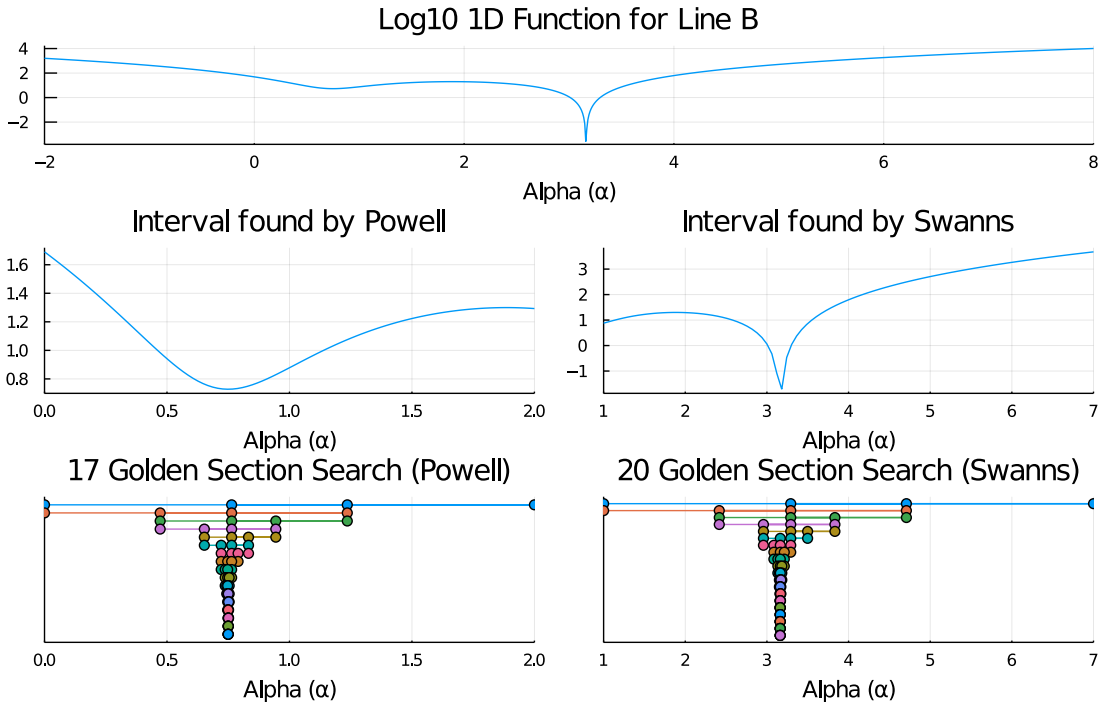


Figure 5: Golden Section Search applied to 1D Line Search Results on Line B. The 1D Function shown in the plots are on a log10 scale. Iterations for Golden Section Search shown where the top is the initial bracketing interval, and towards the bottom is the final bracketing interval.

There are some interesting observations on the results of the Golden Section Search on Line B

- Because Powell and Swann's found different intervals the Golden Section Search refined the intervals into two different minima.
- The global minimum of the 2D function is found near $\alpha \approx 3$. The Golden Section Search from the Swann's Bracketing Method correctly found this global minimum.
- The inner (non-end-point) function evaluations for the Golden Section Search are also shown in the plot. As can be seen, because of the golden ratio the previous functionevaluation points are able to be correctly reused.

Point Label	(x, y) Location
D	(-2, -2)
E	(-1.5, 1.5)
F	(-1, 3)
G	(-0.5, -1.5)
H	(2, 2)
Global Min	(1, 1)

Table 1: Initial Points and Global Minimum Locations for 2D Rosenbrock Objective Function Search

2 Q2. Search on 2D Rosenbrock Banana Function

2.1 Set-up

In this section, a couple of search methods are applied to the full 2D Rosenbrock Banana Function from various different initial points. The two major search methods are:

- Gradient Descent Method
  - Using the gradient (∇f) of the Rosenbrock function find the next direction to search
  - Then using Swann's and Powell's Methods (given some tolerance parameter), the algorithm searches in that 1D direction found using the gradient
  - Once a minimum point in that 1D direction is found, the algorithm loops until the $\|\nabla f\| < 10^{-4}$.
- (EXTRA) Hooke-Jeeves Method

These methods are applied to various initial positions on the 2D parameter search space. The labels for the initial points are found below in Table. **??**. These points relative to the contours of the 2D Rosenbrock Function are also shown in the following section.

2.2 Gradient Descent Results

Each gradient step of the algorithm is recorded and plotted in Fig. **??** from the various different initial points in Table. **??**. As seen in Fig. **??**, various different settings were used to see if there were any differences in the resulting trajectories. The two line search initial bracketing methods used were Powell's and Swann's. Two different settings for the tolerance was also used, 10^{-4} shown on the top row, and 10^{-2} shown on the bottom row.

The values of the objective function for each gradient descent algorithm can be found in Fig. **??**.

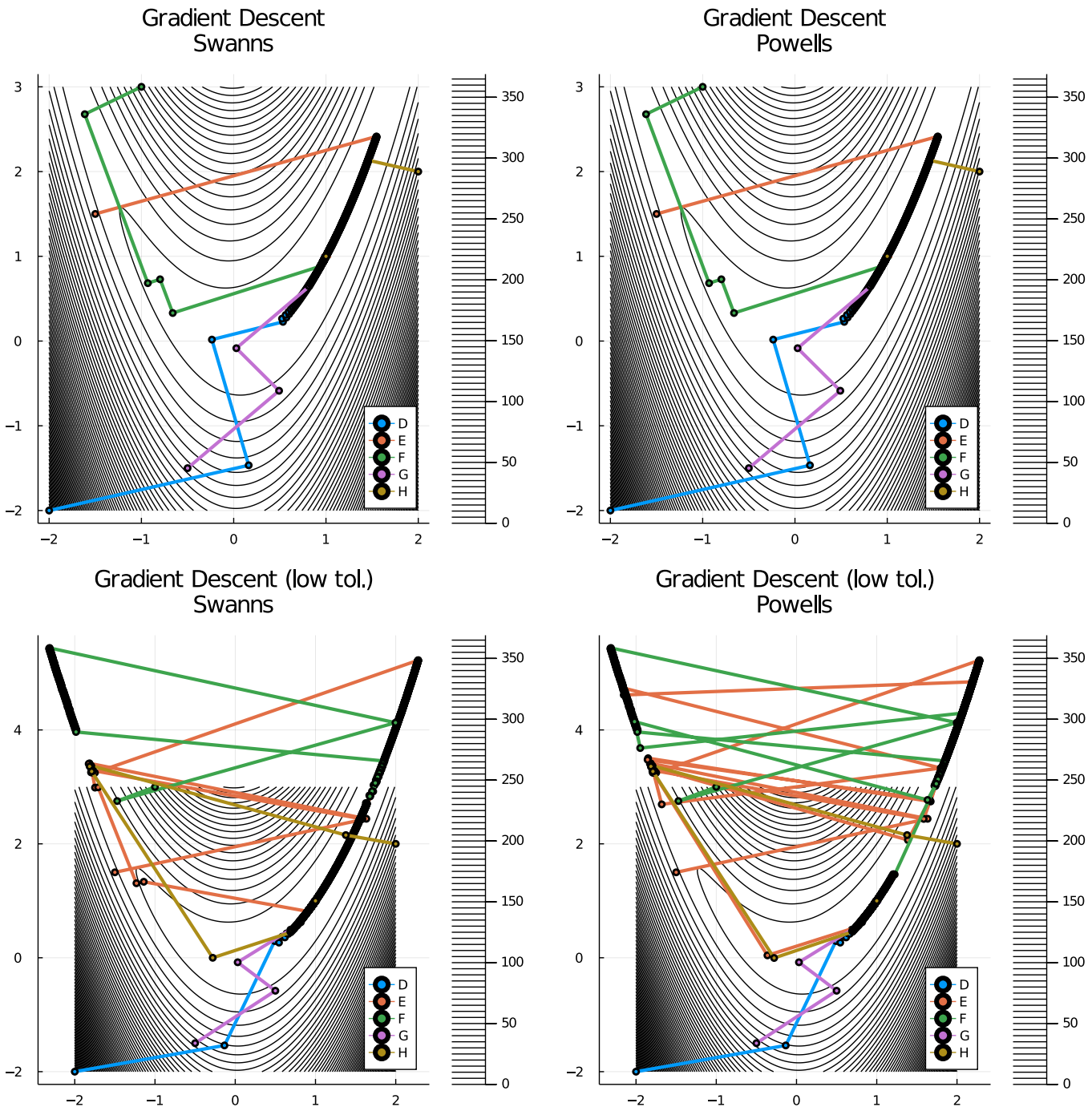


Figure 6: The trajectories of the Gradient Descent on Points D, E, F, G, and H using Powell's and Swann's Bracketing Methods. The top row uses a line search tolerance of 10^{-4} while the bottom row (low tol) uses a line search tolerance of 10^{-2}. Contours only shown in limited region for clarity. Global Minimum located at $(1, 1)$.

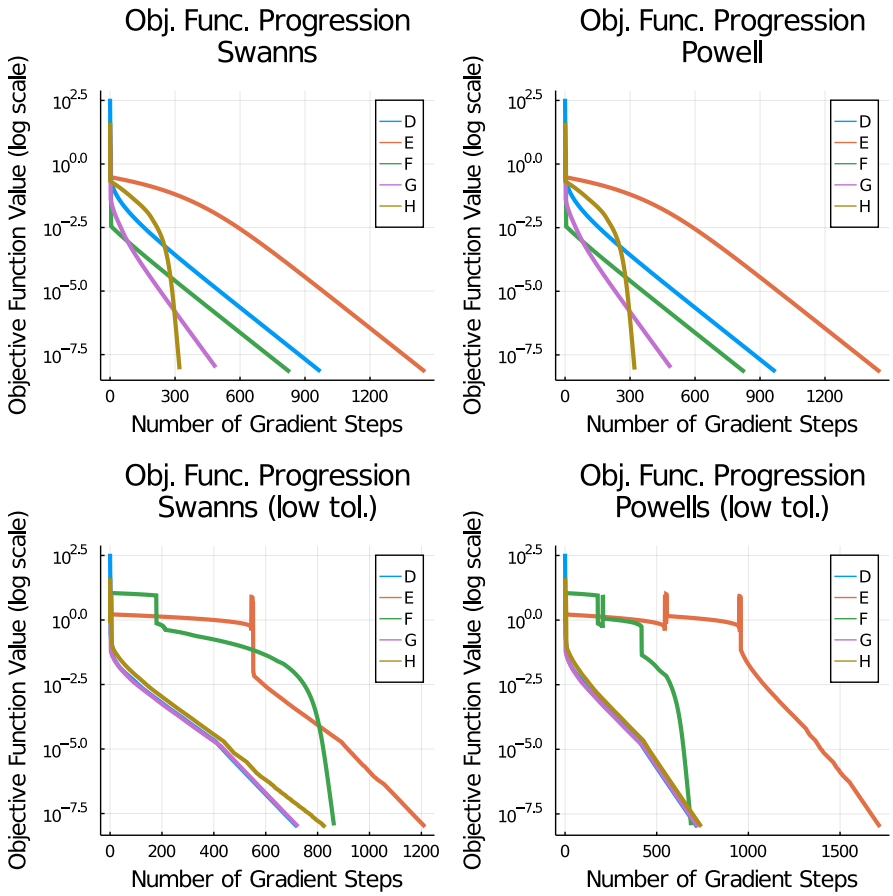


Figure 7: Objective Function Value vs Gradient Descent Steps for Points D, E, F, G, and H using Powell's and Swann's Bracketing Methods. The top row uses a line search tolerance of 10^{-4} while the bottom row (low tol) uses a line search tolerance of 10^{-2}. Global Minimum located at $(1, 1)$.

There are a few interesting observations to note from these Gradient Descent results seen in the contour plots on Fig. **??**

- On the the top row (with the correct amount of tolerance) the algorithm was able to correctly finds the global minimum at $(1, 1)$ from the various initial starting positions.
- There is no noticeable difference between the gradient steps of Swann's and Powell's on the top row.
- As per the course notes, every step of the gradient descent should be orthogonal to the previous step as the gradient. This phenomena can be seen in the top row with the properly calibrated line search tolerance, where each gradient step is at 90 degrees to the previous gradient step.
- All gradient descent trajectories encountered difficulty in the valley of the Rosenbrock Banana Function. The trajectories had to zig-zag down the valley to try to find the minimum.

There are a few interesting observations to note from Fig. **??** with the objective function value plotted against the number the gradient descent steps.

- On the top row, the effect of zig-zag through the valley can be seen with the much slower descent taking much more gradient steps to find the global minimum.
- The curve jumps more erratically in the bottom row with the low tolerance.

2.3 (EXTRA) Hooke-Jeeves Result

The Hooke-Jeeves algorithm (with parameters $\Delta = 0.2$ and $\delta = 0.001$) was applied to the same 2D Rosenbrock Banana Function starting at the same initial values. The results from this can be seen in Fig. **??**.

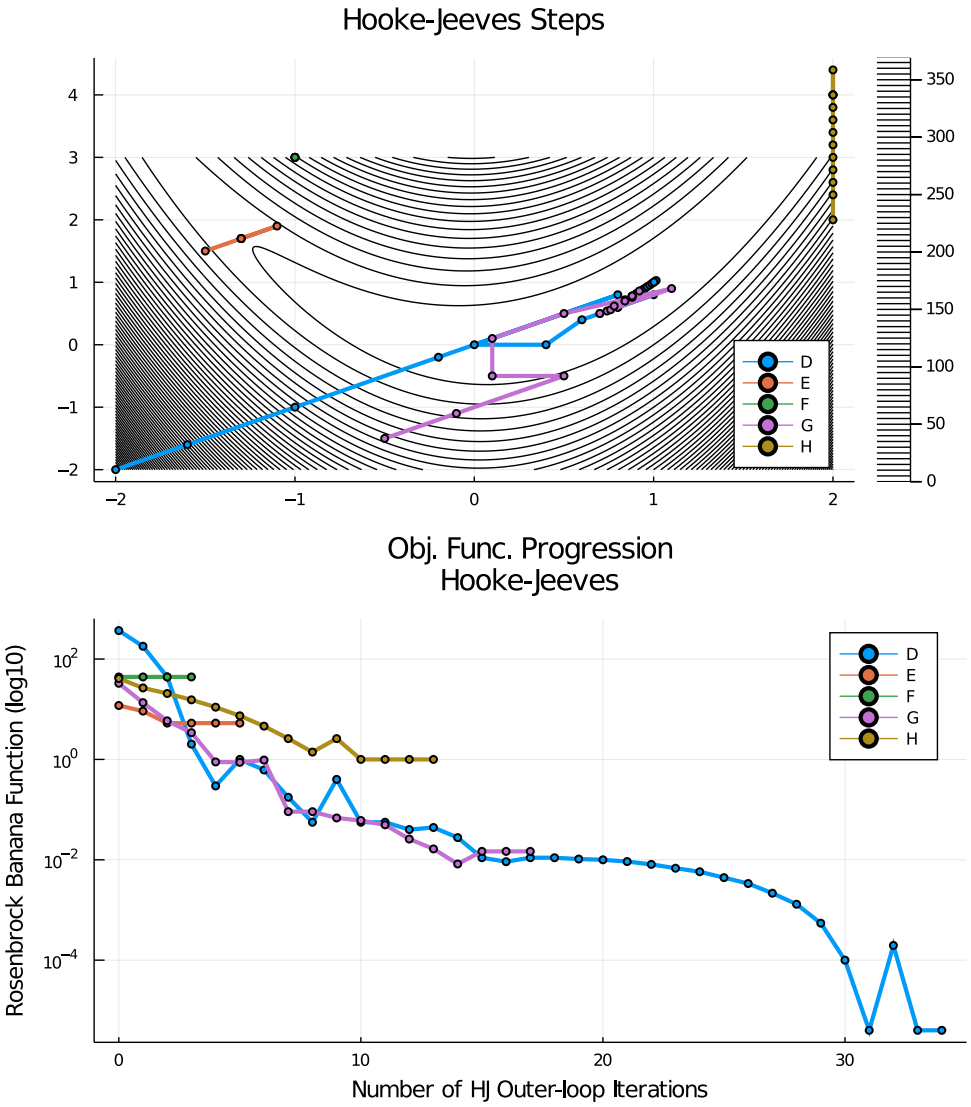


Figure 8: Hooke-Jeeves steps visualized when applied to the 2D Rosenbrock Function.

There are a few interesting observations from this.

- Because Hooke-Jeeves does not have access to the gradient information, the algorithm seems to have a harder time finding the global minimum from points E, F, and H.
- The behavior of bouncing around in the valley can still be seen with D and G.
- The trajectory from point H went towards the valley, but away from the global minimum.

t	0	1	2	3	4	5	6	7	8	9
y(t)	1.75	1.65	1.56	1.49	1.43	1.37	1.33	1.29	1.26	1.23

Table 2: Data used for Model Fitting

3 Q3. Model Fitting using Optimization Techniques

3.1 Set-up

Given a set of data point (also called measurements) and a model, optimization techniques can be applied to find the model parameters which best fit the data.

In this lab assignment, the model f used will be the following where the parameter vector is $x \in \mathbb{R}^5$. The data used for fitting is found in Table **??**.

$$f(t;x) = y = x_1 + x_2 e^{x_4 t} + x_3 e^{x_5 t} \tag{3}$$

The objective function used for optimization will be the sum of squared errors, and as such the optimization will try to find the parameters that best fit the data in the least squares sense.

In this section, model fitting will be done using two different optimization techniques:

- Gradient Descent (as described in Q2)
- Hooke-Jeeves

3.2 Gradient Descent Method

3.2.1 Sum Squares Function and Gradient Derivation

In order to apply the Gradient Descent method to the objective function, the gradient function must be obtained. The following steps show the symbolic derivation of the gradient of the objective function.

First, we define the objective function L, which is way to evaluate a model's fit of the data (t_i, y_i) in a least squares sense.

$$L = \sum_i (y_{\text{data}} - y_{\text{model}})^2 \tag{4}$$

$$L = \sum_i (y_i - (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}))^2 \tag{5}$$

We can now derive the gradient ∇L.

$$\nabla L = \nabla \left(\sum_i (y_i - (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}))^2 \right) \tag{6}$$

$$\nabla L = \sum_i \left(\nabla \left((y_i - (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}))^2 \right) \right) \tag{7}$$

$$\nabla L = \sum_i \left(2(y_i - (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i})) \cdot \nabla (y_i - x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}) \right) \tag{8}$$

$$\nabla L = \sum_i \left(2(y_i - (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i})) \cdot \nabla (x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}) \right) \tag{9}$$

Label	$[x_1, x_2, x_3, x_4, x_5]$ **Initial Parameter Vector**
Initial Guess 1	$[0.1, 0.1, 0.1, 0.1, 0.1]$
Initial Guess 2	$[-0.1, -0.1, -0.1, -0.1, -0.1]$
Initial Guess 3	$[0.0, 0.1, -0.1, 0.0, 0.1]$

Table 3: Labels for Initial Parameters used in Model Fitting

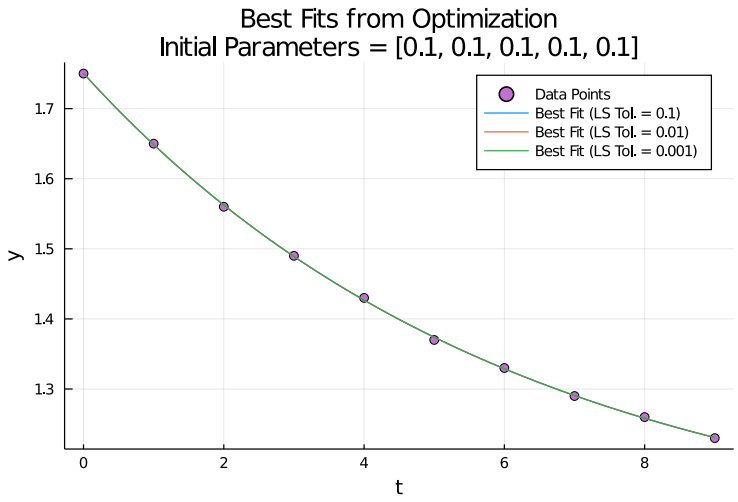
$$\nabla(x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i}) = \begin{bmatrix} 1 \\ e^{x_4 t_i} \\ e^{x_5 t_i} \\ x_2 t_i e^{x_4 t_i} \\ x_3 t_i e^{x_5 t_i} \end{bmatrix} \tag{10}$$

The above expressions for ∇L and $\nabla(x_1 + x_2 e^{x_4 t_i} + x_3 e^{x_5 t_i})$ define the gradient in full given some data and a location in the parameter space to evaluate the gradient.

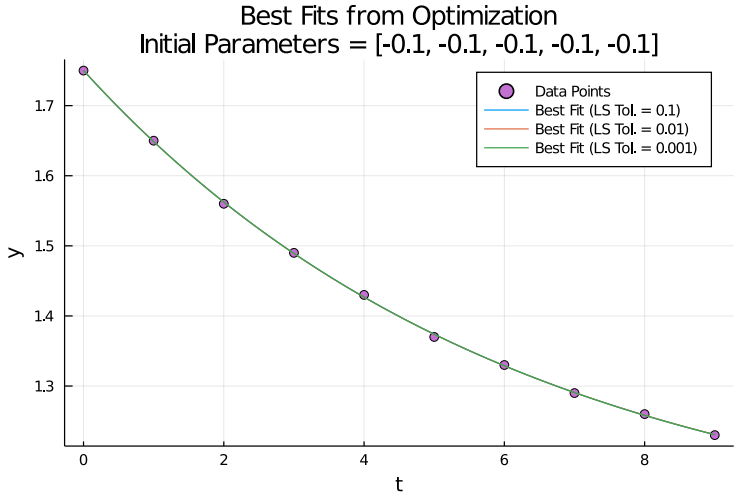
3.2.2 Gradient Descent Results

Using the gradient defined above, the following figures are results from the gradient descent algorithm from various different starting guesses. The starting guesses can be found in Table **??**.

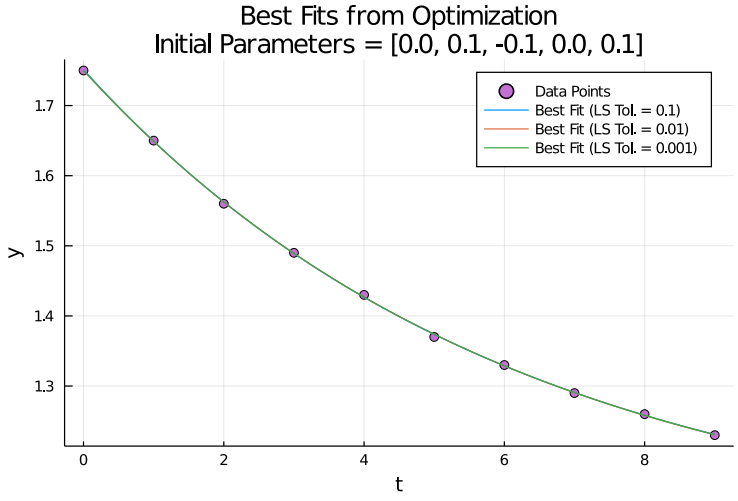
The resulting models are shown in Fig. **??**. They are plotted against the data.



(a) Initial Guess 1

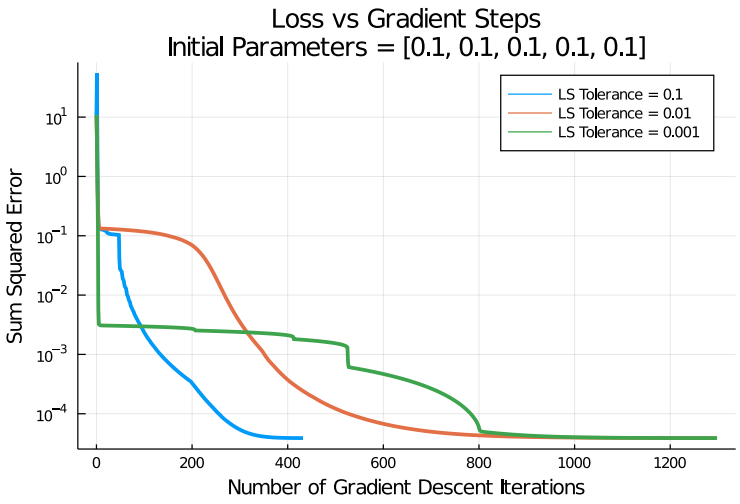


(b) Initial Guess 2

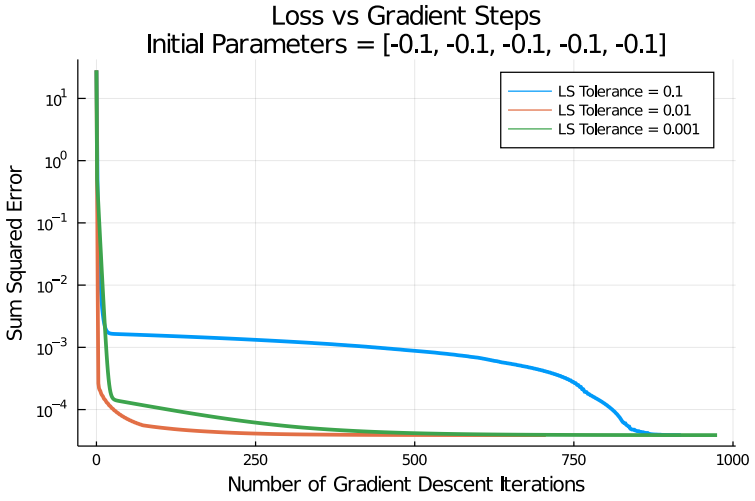


(c) Initial Guess 3

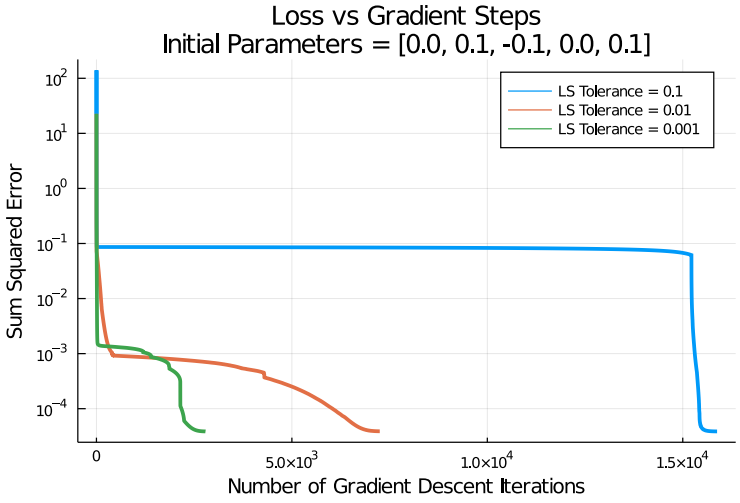
Figure 9: Result of optimization of model parameters using various initial guesses. Fitted model plotted against data. Various Line Search tolerances were used to do the gradient descent as shown in the legend of the plots.



(a) Initial Guess 1



(b) Initial Guess 2



(c) Initial Guess 3

Figure 10: Loss vs Gradient Descent Iterations plot for various guesses. Different line search tolerances were used as shown in legend of plots.

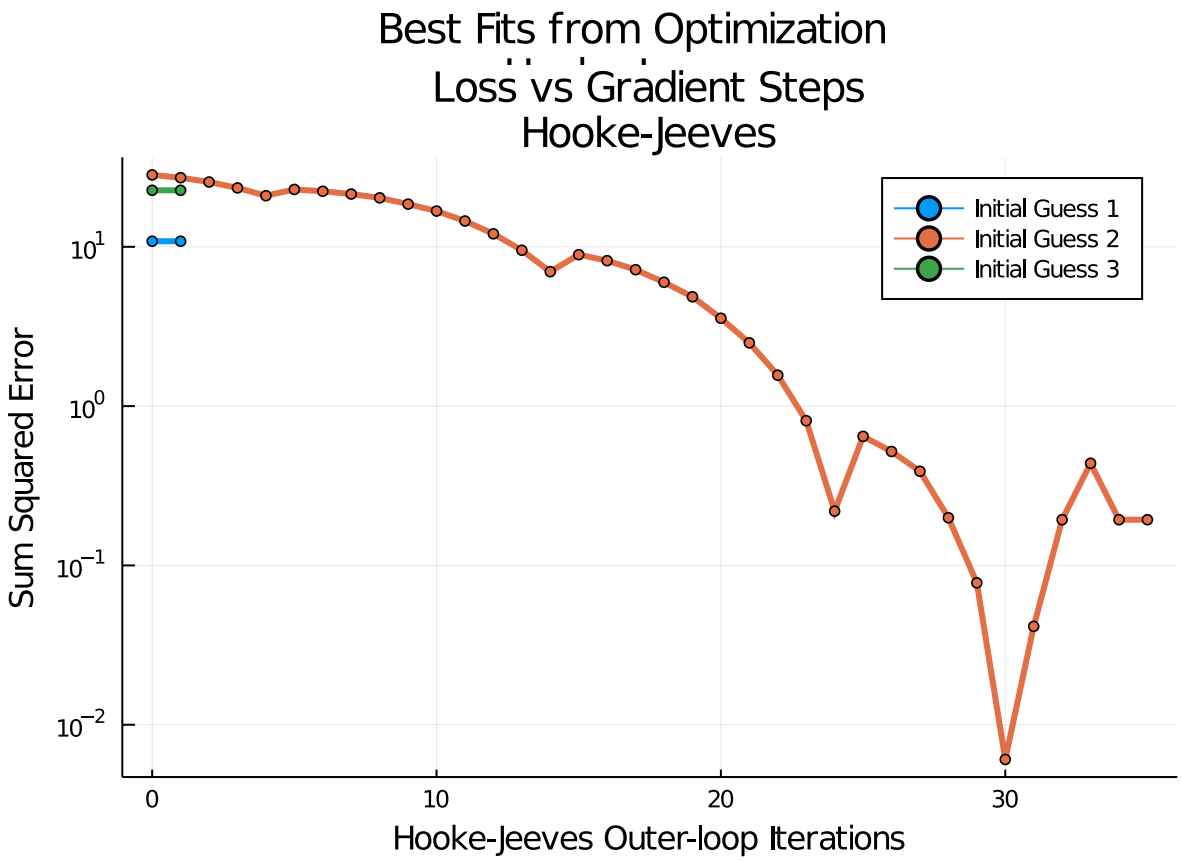
Initial Guess	Line Search Tolerance	Number of Function Evals	Number of Gradient Steps	Final Parameter Vector [x1, x2, x3, x4, x5] (rounded)
Initial Guess 1	0.1	6113	433	[1.076, 0.337, 0.337, -0.163, -0.163]
	0.01	24643	1298	[1.076, 0.337, 0.337, -0.163, -0.163]
	0.001	29904	1299	[1.074, 0.338, 0.338, -0.162, -0.163]
Initial Guess 2	0.1	12918	919	[1.074, 0.338, 0.338, -0.163, -0.163]
	0.01	13416	707	[1.076, 0.337, 0.337, -0.163, -0.163]
	0.001	22425	976	[1.074, 0.338, 0.338, -0.163, -0.163]
Initial Guess 3	0.1	222340	15879	[1.078, -0.002, 0.674, -51.631, -0.165]
	0.01	137726	7249	[1.074, 0.000, 0.676, -1.693, -0.163]
	0.001	64230	2791	[1.075, -0.001, 0.675, -4.498, -0.163]

There are a few interesting observations to make from these results.

- The different initial guesses and line search tolerances used did not change the final fit of the model parameter.
- The different initial guesses and line search tolerences did make a difference into the exact final parameter vector that the technique converged to. This seems to indicate multiple local minima that all have various similar fits (least squares sense). In fact as seen with some of the final parameter vectors having $x_2 \approx 0$ term (from Initial Guess 3), a single constant term summed up with an exponential term may be another suitable model instead of two different exponential terms as given in the assignment. Recall that the original model given the assignment is $f(t;x) = x_1 + x_2 e^{x_4 t} + x_3 e^{x_5 t}$.
- Not shown here, however comparing the results between using Swann's vs Powell's for the line search bracketing did not make a difference between the final fit of the model.
- All the trials was able to reduce the sum of squared error to 10^{-1} very quickly in a short amount of gradient steps. The rest of the gradient steps were used to reduce that error down lower than 10^{-4}.
- The run using Initial Guess 3 and Line Search tolerance of 0.1 was stuck for a large amount of steps as seen in the plot and the number of function evaluations and gradient descent evaluations.

3.3 Hooke-Jeeves Direct Search Method Results

The Hooke-Jeeves algorithm (with parameters $\Delta = 0.1$ and $\delta = 0.01$) was applied using the same initial guesses as in Table **??** to the Model Fitting optimization problem. A visualization of the results can be found on Fig. **??**. A table summarizing the number of function evaluations and the final parameters can be seen in Table ???. (Note zero gradient function evaluations were used because the Hooke-Jeeves algorithm does not use the gradient function.)



(
I .
n(d)
N Loss
el vs
fo found
us-
ing
Hooke-
Jeeves
ch Hooke-
pa Jeeves
with
data.

Figure 11: Results from Hooke-Jeeves on Model Fitting

As seen compared with the results from the Gradient Descent algorith...

3.4 (EXTRA) Automatic Differentiation Gradient Descent

4 Q4. (Graduate Student)

Let V be the vector space of polynomial functions from $\mathbb{R} \to \mathbb{R}$ which have degree less than or equal to 2. That is, for all $a, b, c \in \mathbb{R}$ the following polynomial $p(x)$ is in V

$$p(x) = a + bx + cx^2$$

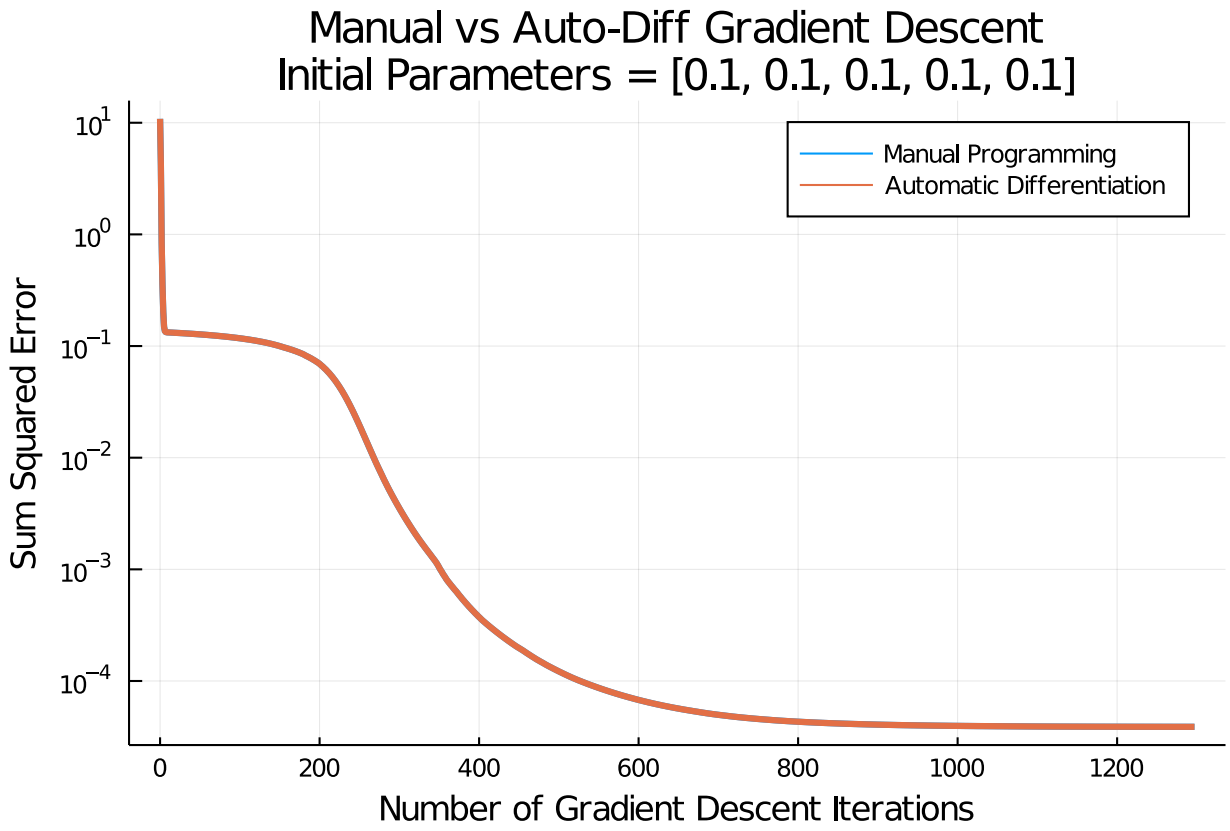


Figure 12: Caption

4.1 Proving B_1 and B_1 are both a basis of V

4.1.1 Proof: B_1 is Linearly Independent

- Implication of only way to get 0 vector.

4.1.2 Proof: B_1 spans V

- Show all $v \in V$ can be represented in basis

4.1.3 Proof: B_2 is Linearly Independent

4.1.4 Proof: B_2 spans V

4.2 Transformation between B_1 and B_2

4.3 D Derivative Operator

4.3.1 Proof: D is linear

4.3.2 Matrix Representation of D in the bases B_1 and B_2

5 Q5. (Graduate Student)

5.1 x in full $\mathbb{R}^3$ standard basis

5.2 Coordinates of x in B_X

5.3 Coordinates of y in B_Y using L ($c_y = Lc_x$)

5.4 y in full $\mathbb{R}^4$ standard basis

6 Q6. (Graduate Student)

6.1 Df derivation

Let $f : \mathbb{R}^4 \to \mathbb{R}$ be defined as $f(x) = \|Ax - b\| + \lambda\|Cx\| + \gamma\|Eb\|$ where $A, C, E \in \mathbb{R}^{4\times4}$, $x, b \in \mathbb{R}^4$ and $\lambda, \gamma \in \mathbb{R}$.

Let D denote the differential operator.

$$
\begin{aligned}
Df(x) &= D\left(\|Ax - b\| + \lambda\|Cx\| + \gamma\|Eb\|\right) \\
&= D\left(\|Ax - b\|\right) + \lambda D\left(\|Cx\|\right) + \gamma D\left(\|Eb\|\right) && \text{[using linearity of D]} \\
&= D\left(\|Ax - b\|\right) + \lambda D\left(\|Cx\|\right) + 0 \\
&= D\left(\|Ax - b\|\right) + \lambda D\left(\|Cx\|\right) \\
&= \frac{(Ax - b)^T A}{\|Ax - b\|} + \lambda\frac{(Cx)^T C}{\|Cx\|} && \text{[using derivations from next sections]}
\end{aligned}
$$

The full derivation of $D\left(\|Ax - b\|\right)$ and $D\left(\|Cx\|\right)$ can be found in the following sections.

6.2 $D\left(\|Cx\|\right)$ Derivation

$$
\begin{aligned}
D\left(\|Cx\|\right) &= D\left(\sqrt{(Cx)^T(Cx)}\right) \\
&= D\left(\left((Cx)^T(Cx)\right)^{\frac{1}{2}}\right) \\
&= \frac{1}{2}\left((Cx)^T(Cx)\right)^{\frac{-1}{2}} D\left((Cx)^T(Cx)\right) && \text{[using Chain Rule]} \\
&= \frac{1}{2\sqrt{(Cx)^T(Cx)}} D\left((Cx)^T(Cx)\right) \\
&= \frac{1}{2\|Cx\|} D\left((Cx)^T(Cx)\right) \\
&= \frac{\left((Cx)^T D(Cx)\right) + \left((Cx)^T D(Cx)\right)}{2\|Cx\|} && \text{[using Product Rule]} \\
&= \frac{2\left((Cx)^T D(Cx)\right)}{2\|Cx\|} \\
&= \frac{(Cx)^T C}{\|Cx\|}
\end{aligned}
$$

6.3 $D\left(\|Ax-b\|\right)$ Derivation

$$
\begin{aligned}
D\left(\|Ax-b\|\right) &= D\left(\sqrt{(Ax-b)^T(Ax-b)}\right) \\
&= D\left(\left((Ax-b)^T(Ax-b)\right)^{\frac{1}{2}}\right) \\
&= \frac{1}{2}\left((Ax-b)^T(Ax-b)\right)^{\frac{-1}{2}} D\left((Ax-b)^T(Ax-b)\right) \quad \text{[using Chain Rule]} \\
&= \frac{1}{2\sqrt{(Ax-b)^T(Ax-b)}} D\left((Ax-b)^T(Ax-b)\right) \\
&= \frac{1}{2\|Ax-b\|} D\left((Ax-b)^T(Ax-b)\right) \\
&= \frac{\left((Ax-b)^T D(Ax-b)\right) + \left((Ax-b)^T D(Ax-b)\right)}{2\|Ax-b\|} \quad \text{[using Product Rule]} \\
&= \frac{2(Ax-b)^T D(Ax-b)}{2\|Ax-b\|} \\
&= \frac{2(Ax-b)^T A}{2\|Ax-b\|} \\
&= \frac{(Ax-b)^T A}{\|Ax-b\|}
\end{aligned}
$$

A Extra Plots

A.1 Q1 Extra Plots

B Source Code

B.1 All Optimization Algorithms (A1Module.jl)

B.2 Plot Generation Script (makeplots.jl)

B.3 (EXTRA) Automatic Differentiation for Gradient

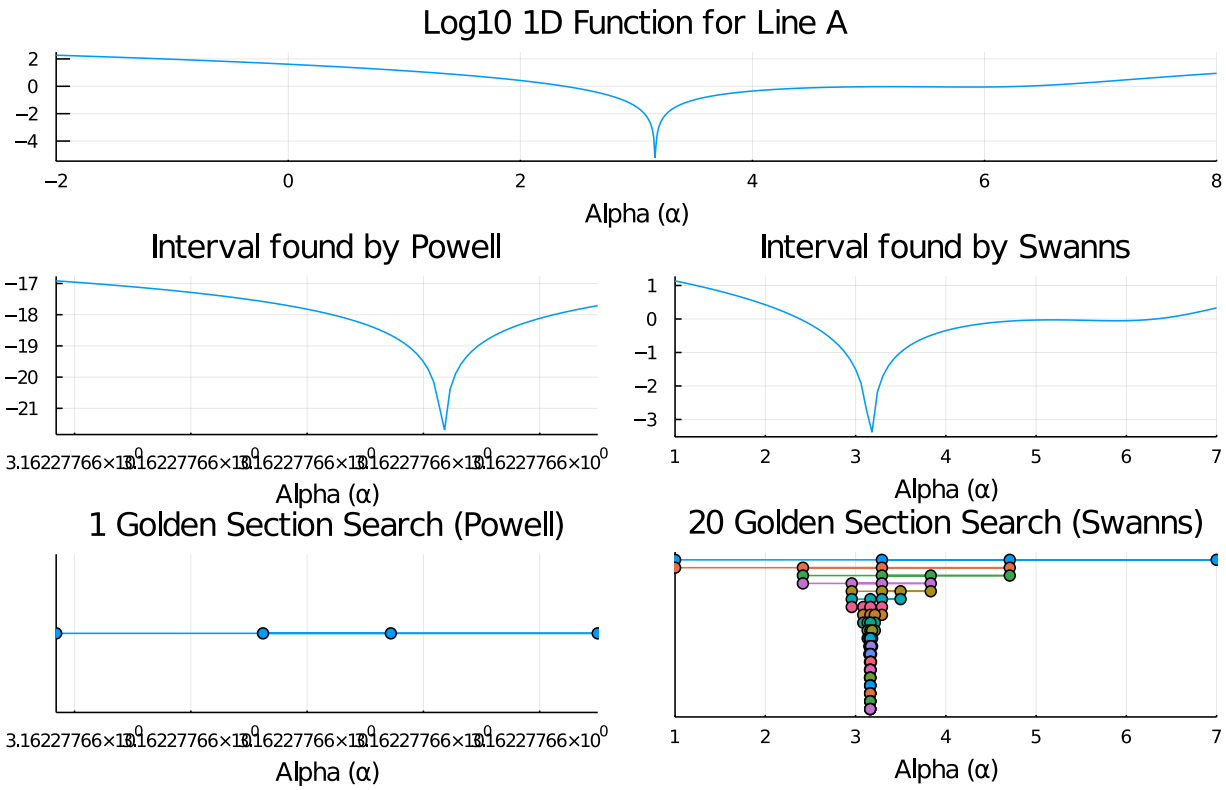


Figure 13: Caption

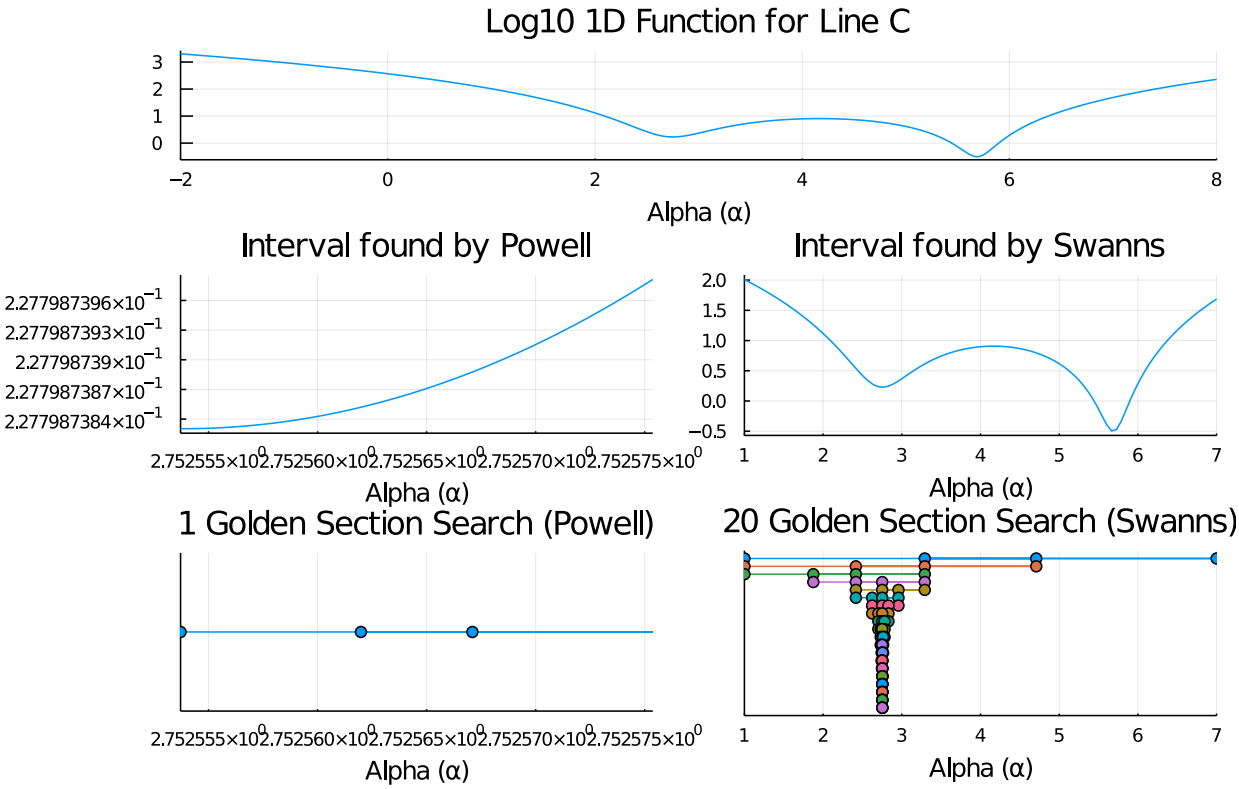


Figure 14: Caption