NUMERICAL OPTIMIZATION FINAL PROJECT

LAURA LYMAN

1. Abstract & Problem Statement

The goal of this project is to cover an ellipse with some finite number of rectangles K such that the area between the ellipse and the union of rectangles is minimized. We construct an objective function f and show that this problem can be expressed as a minimization of f subject to linear constraints. Then the optimization problem is solved by Newton's method, steepest descent, and BFGS with the constraints enforced by bounding the step length α in the line-search algorithm. We assume a consistent initial guess in which the x-coordinates of the right rectangular corners correspond to equidistant points on the ellipse in the first quadrant; however, an extension of this project could involve altering this initial guess. Each of the three listed methods is analyzed in terms of accuracy, number of line-search steps, number of iterations, and computation time. We find that Newton's method performs best overall in these categories, which is likely explained by the Hessian being sparse and an accurate second-order approximation of f. Steepest descent and BFGS demonstrate less promising results, probably due to: (1) the slow convergence rate of steepest descent (even when the Hessian is reasonably well-conditioned), and (2) BFGS not necessarily preserving the sparsity of the Hessian when approximating in each iteration.

2. Preliminaries

Suppose we are given an ellipse $\mathcal{E}$ in $\mathbb{R}^2$. Without loss of generality, assume the ellipse is oriented with its major axis along the x-axis — for we can always rotate the ellipse so that it is oriented this way without altering the ratio of rectangle area to ellipse area. That is, we assume that

$$\mathcal{E} = \{(x, y) \in \mathbb{R}^2 : \tfrac{x^2}{a^2} + \tfrac{y^2}{b^2} = 1\}$$

with $a > b$. We are attempting to find rectangles $\mathcal{R}_1, \ldots, \mathcal{R}_K$ such that

$$\mathcal{E} \subseteq \left(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_K\right) \overset{\text{call}}{:=} R$$

and the ratio

$$\frac{\text{area}(R)}{\text{area}(\mathcal{E})} = \frac{\text{area}(R)}{\pi ab} \quad (\geq 1)$$

is minimized. The problem can be further simplified by scaling $\mathcal{E}$ and each $\mathcal{R}_k$ with the map

$$(x, y) \longmapsto (bx, by)$$

so that $\mathcal{E} \longmapsto \{(x, y) \in \mathbb{R}^2 : \frac{x^2}{a^2/b^2} + y^2 = 1\}$, which has the *general* form

$$\mathcal{E} = \{(x, y) \in \mathbb{R}^2 : \tfrac{x^2}{a^2} + y^2 = 1, a > 1\}.$$

The ellipse will be referred to by this formula. Note that the scaling does not change the ratio of areas.

Intuitively, the rectangles should have their sides be parallel to the x and y axes and be symmetric with respect to these axes for optimality. We make this assumption going forward, though a further direction of the project could involve tilted rectangles for comparison. Specifically, each $\mathcal{R}_k$ will have the form depicted in Figure 1 with $0 < y_k \leq x_k$, which we can specify by the corner coordinates

$$(x_k, y_k), (-x_k, y_k), (-x_k, -y_k), (x_k, -y_k).$$

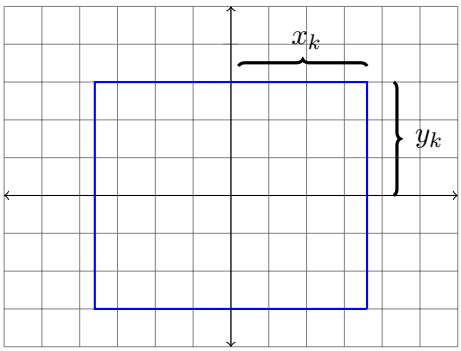


FIGURE 1. Form of rectangle $\mathcal{R}_k$. Symmetric with respect to x and y axes with sides parallel to these axes.

Without loss of generality, order the rectangles such that

$$x_1 \leq \cdots \leq x_K = a$$

for $\mathcal{R}_1, \ldots, \mathcal{R}_K$, where we require $x_K = a$ and $\max_{1\leq k\leq K} y_k = 1$ so that the ellipse is actually covered. Furthermore, if $y_j < y_\ell$ for some $j < \ell$, then

$$\mathcal{R}_j \subset \mathcal{R}_\ell$$

since $j < \ell \Rightarrow x_j \leq x_\ell$ by how we defined our ordering and $y_j < y_\ell$. Since this scenario forces $\mathcal{R}_j$ to be redundant in the covering, we require the y_k to be non-increasing; that is,

$$1 = y_1 \geq \cdots \geq y_K.$$

In addition, we have the following simple result.

Theorem 2.1. *The optimal rectangle covering has* $\frac{x_k^2}{a^2} + y_{k+1}^2 = 1$ *for* $1 \leq k \leq K-1$, *meaning the height* y_{k+1} *of the* $(k+1)$*th rectangle is determined by where the* k*th rectangle intersects the ellipse.*

Proof. Let $h(x) = \sqrt{1 - \frac{x^2}{a^2}}$, so that $(x_k, h(x_k))$ is a point on the ellipse intersecting the kth rectangle. If $y_{k+1} > h(x_k)$ then we have

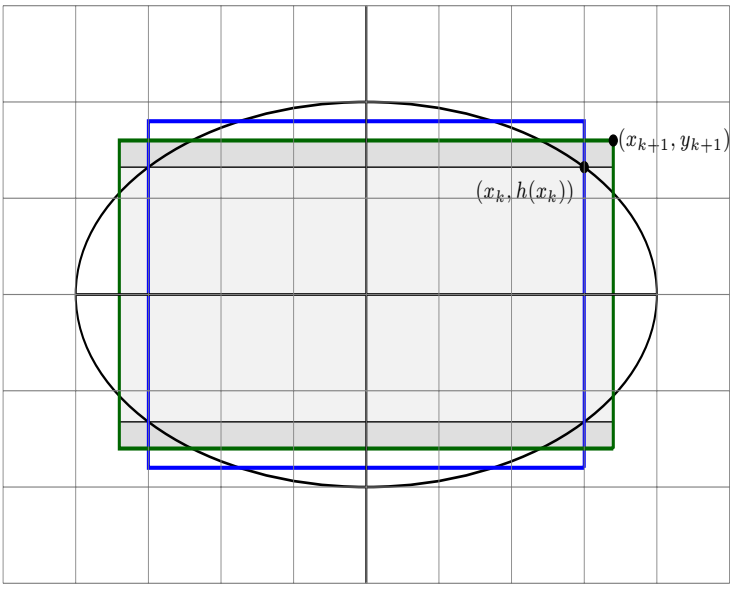


FIGURE 2. Rectangles $\mathcal{R}_k$ (blue) and $\mathcal{R}_{k+1}$ (green) when $y_{k+1} > h(x_k)$. The area of $\mathcal{R}_{k+1}$ can be reduced to the inner light gray box while still covering the same portion of $\mathcal{E}$ not covered by $\mathcal{R}_k$.

From Figure 2, we can see that $\mathcal{R}_{k+1}$ (dark gray with green border) can be made into a tighter fit for the same x_{k+1} by decreasing y_{k+1} (resulting in the light gray rectangle); that is, $\mathcal{R}_{k+1}$ can cover the same portion of $\mathcal{E}$ not covered by $\mathcal{R}_k$ as it did previously. Furthermore, if $y_{k+1} < h(x_k)$ then some area of the ellipse is left uncovered between x_k and x_{k+1}.

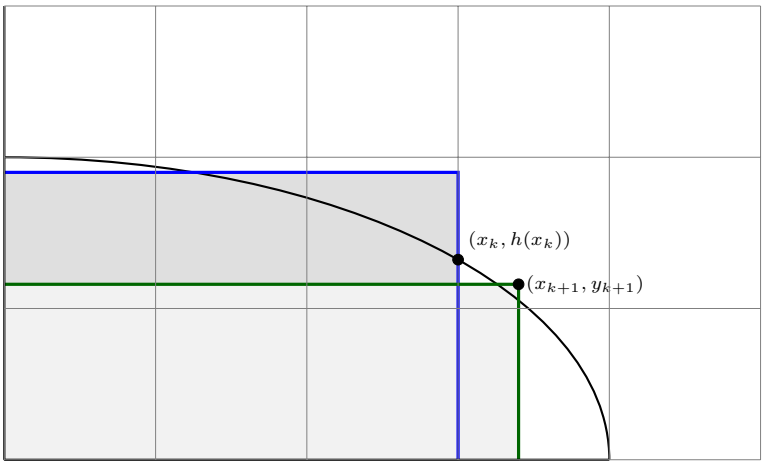


FIGURE 3. Rectangles $\mathcal{R}_k$ (blue) and $\mathcal{R}_{k+1}$ (green) when $y_{k+1} < h(x_k)$. There is area of $\mathcal{E}$ between x_k and x_{k+1} that is not covered.

Furthermore, this area will *not* be covered by any of the other rectangles since doing so would require an x-coordinate between x_k and x_{k+1}. Therefore, the tightest covering necessarily has $\frac{x_k^2}{a^2} + y_{k+1}^2 = 1$ for $1 \leq k \leq K-1$. □

Note that Theorem 2.1 does not address *how* to pick $x_1 \leq \cdots \leq x_K$, but instead specifies the corresponding heights y_k given this ordering of x-coordinates. Now $\text{area}(\mathcal{R}_1) = x_1 y_1 = x_1$ since $y_1 = 1$. Using the height function from Theorem 2.1

$$h(x) = \sqrt{1 - \frac{x^2}{a^2}}$$

the kth rectangle contributes uncovered area $2(x_k - x_{k-1})h(x_{k-1})$ for $k > 1$. So defining $x_0 = 0$ (and recalling that $x_K = a$), the optimization problem in question becomes

$$\begin{aligned} \min_{x_1,\cdots,x_K \in \mathbb{R}} \quad & \sum_{k=0}^{K-1} (x_{k+1} - x_k) h(x_k) \overset{\text{call}}{:=} f(\vec{x}) \\ \text{such that} \quad & 0 = x_0 < x_1 \leq \cdots \leq x_{K-1} \leq x_K = a. \end{aligned}$$

where f denotes our objective function.

3. Gradient and Hessian Calculations

To perform the line search algorithm as provided by Nocedal et. al ([3]), we need to calculate the Hessian and gradient of the objective function. In particular, we have for Newton's method that at each iteration

$$p_k = -\mathcal{H}_k^{-1} \nabla f_k$$

where p_k is the computed descent direction and $\mathcal{H}_k = \nabla^2 f(x_k)$ is the Hessian evaluated at x_k. BFGS is quite similar; the key difference is that the approximate Hessian is used in place of the true Hessian. Instead of computing $\mathcal{H}_k$ at every iteration, the Hessian is updated in a simple manner to account for the curvature measured during the most recent step ([2], [3]). For steepest descent, we simply have that

$$p_k = -\nabla f_k.$$

Compute that

$$\frac{\partial f(x)}{\partial x_j} = [h(x_{j-1}) - h(x_j)] + (x_{j+1} - x_j)h'(x_j),$$

where $h'(x) = \frac{\frac{x}{a^2}}{\sqrt{1-\frac{x^2}{a^2}}}$. Consequently, we see that $\frac{\partial^2 f(x)}{\partial x_j \partial x_i} = 0$ if $i, (i-1), (i+1) \neq j$, so the Hessian is tri-diagonal. That is, we have

$$\mathcal{H}(x) = \nabla f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & 0 & \cdots & \cdots & 0 \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ \vdots & & & \ddots & & \frac{\partial^2 f(x)}{\partial x_{K-1} \partial x_K} \\ 0 & \cdots & \cdots & & \frac{\partial^2 f(x)}{\partial x_K \partial x_{K-1}} & \frac{\partial^2 f(x)}{\partial x_K^2} \end{pmatrix}.$$

More explicitly, we can compute that off the diagonal (noting that the matrix is symmetric)

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_{i+1}} = h'(x_i)$$

and on the diagonal

$$\begin{aligned} \frac{\partial^2 f(x)}{\partial x_j^2} &= -h'(x_j) - h'(x_j) + (x_{j+1} - x_j)h''(x_j) \\ &= -2h'(x_j) + (x_{j+1} - x_j)h''(x_j) \end{aligned}$$

where $h''(x) = \frac{a^2\sqrt{1-\frac{x^2}{a^2}}}{(a^2-x^2)^2}$. This completes the necessary calculations.

4. Optimization Techniques & Line Search Results

The minimization problem will be solved through three unrestricted optimization algorithms: Newton's method, BFGS, and steepest descent, where we ensure that the necessary linear constraints are satisfied by choosing steps that do not violate these restrictions. For the line search algorithm, we implement the algorithm as detailed by Nocedal ([3] Section 3.5) so that the Wolfe conditions are in place. Specifically, we require the *Armijo condition*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^\intercal p_k \quad c_1 \in (0,1)$$

and the *curvature condition*

$$\nabla f(x_k + \alpha_k p_k)^\intercal p_k \geq c_2 \nabla f_k^\intercal p_k \qquad c_2 \in (c_1, 1)$$

for the step length α_k at each iteration. As suggested by this text, common values for c_1 and c_2 in practice are $c_1 = 10^{-4}$ and $c_2 = 0.9$ (for Newton/Quasi-Newton methods). The stopping conditions in play will be that the change in the objective function is less than 10^{-15} or the norm of the gradient is less than 10^{-8} ([2],[3]).

In order to preserve the ordering $0 \leq x_1 \leq \cdots \leq x_{K-1} \leq a$, we set a bound $\alpha_k \leq \alpha_{\max}$ so that the x_k stay within their intended domain $[0, a]$. Finally, the initial guess will be that the x_k correspond to equidistant arc lengths on $\mathcal{E}$ from 0 to $\frac{\pi}{2}$. For example, when $K = 3$ the x_k would be chosen as in Figure 4 below.

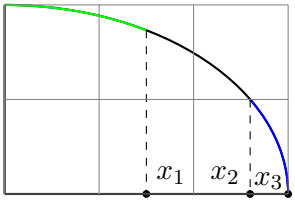
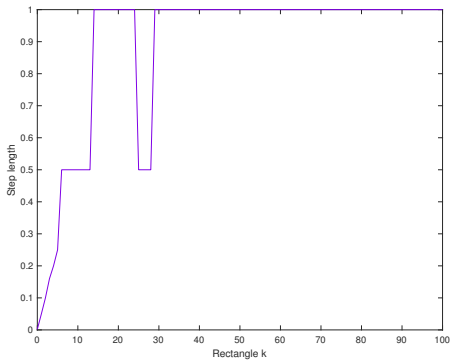


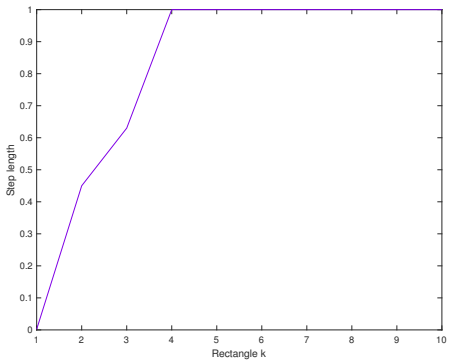
FIGURE 4. Initial guess has the x_k spaced corresponding to equidistant arc lengths on $\mathcal{E}$ in the first quadrant.

An extension of this project could involve an initial guess in which the x_k are spaced non-uniformly.

We begin comparing Newton's method, steepest descent, and BFGS by examining the step length for each found through the line search algorithms (which is the first step that satisfies the Wolfe conditions). Note that $\alpha_0 = 1$ ([3]). Both BFGS and Newton's method were run with $K = 10^4$ with $a = 2$ (where our set-up ensures $b = 1$) and plotted below; the x-axes of the plots were truncated when convergence was clear. As predicted by theory, we see that Newton's method accepted a step of $\alpha = 1$ (and even did so in under 5 iterations). BFGS also showed convergence to $\alpha = 1$, though required a larger number (> 30) of iterations to do so. The delay was probably due to several iterations being used to approximate the Hessian well — though once the Hessian was successfully approximated, $\alpha = 1$ was accepted ([3] Section 6.1).



(A) BFGS

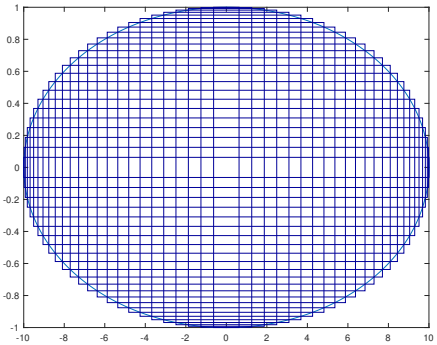


(B) Newton's Method

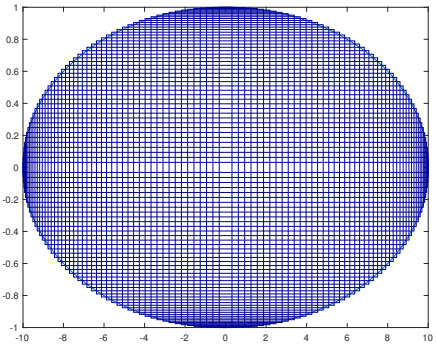
FIGURE 5. Selected step length α_k for BFGS and Newton's method.

However, for steepest descent the results do not show this convergence. In particular, α_k oscillated dramatically for $1 \leq k \leq K = 10^4$ (with an average of 0.0244), and the value $\alpha = 1$ was hardly ever met. By not predicting a valid step, the algorithm uses more time searching for one — which is clearly problematic.

Examples of the optimal solutions for various K are provided below (with $a = 10$ in Figure 6, $a = 2$ in Figure 7).

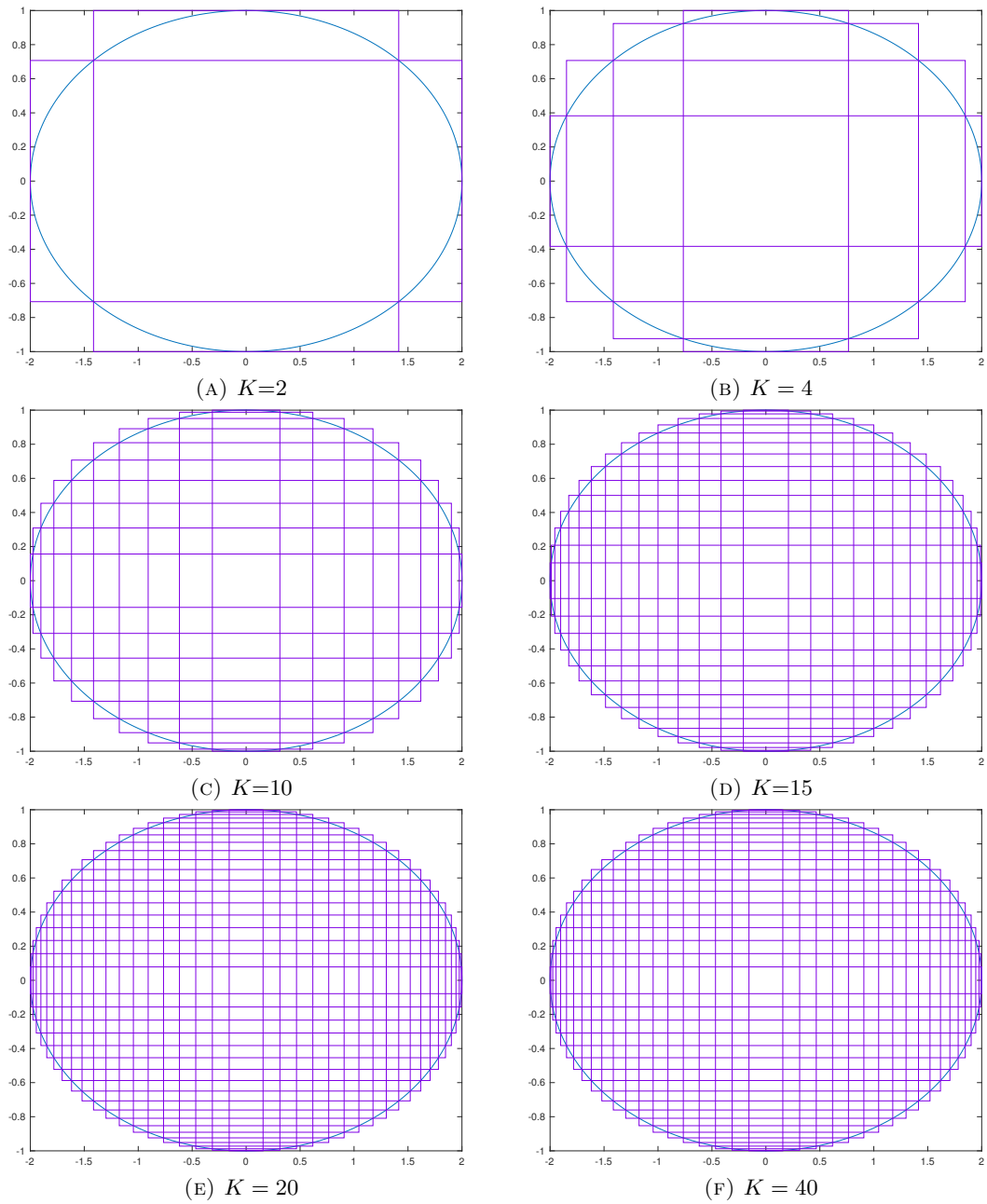


(A) K=25



(B) $K = 50$

FIGURE 6. Covering of $\mathcal{E}$ for $K = 25$ and $K = 50$ with $a = 10$.



(A) K=2 (B) $K = 4$

(C) K=10 (D) K=15

(E) $K = 20$ (F) $K = 40$

FIGURE 7. Covering of $\mathcal{E}$ for various numbers of rectangles K with $a = 2$.

Note that the area of the optimal covering is bounded below by the actual area of the ellipse (which is πa in our case) — and convergence to this lower bound increases with K. In the next section, we present some data on the accuracy of the three algorithms along with corresponding iteration counts/computation times for assorted values of K.

5. Accuracy & Speed of Different Methods

All three optimization methods (Newton's method, steepest descent, and BFGS) were applied at first for small values of K. The plot in Figure 8 shows the logarithmic error for each over 50 iterations for when $a = 2$ and $K = 10$.

From this plot, we can see that Newton's method converges in a small number of iterations (< 5), while BFGS and steepest descent require more to do so. As depicted, the convergence of steepest descent is linear, while Newton and BFGS demonstrate super-linear convergence when close to the optimum, as predicted by theory.

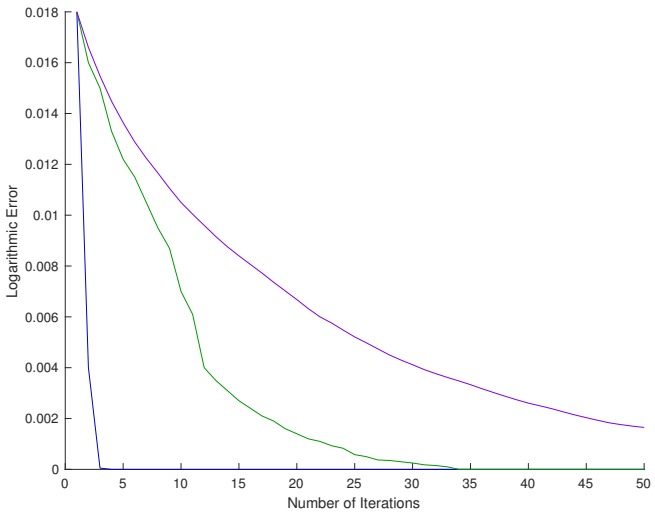


FIGURE 8. Logarithmic error for Newton's method (blue), steepest descent (purple), and BFGS (green) for $K = 10$.

For larger values of K, the error of Newton's method continued to decrease quickly; for example, with $K = 100$, the error dropped to under $\frac{1}{2} \times 10^{-4}$ in under 5 iterations. This is likely explained by having a sparse (tri-diagonal) Hessian and an accurate second-order approximation of f. However, BFGS and steepest descent were not as successful for larger problems. They are plotted below ($K = 100$) for comparison. Note that the y-axis scaled is by 10^{-4}.

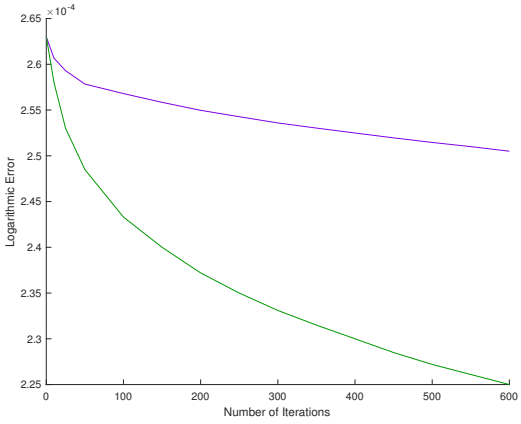


FIGURE 9. Logarithmic error for steepest descent (purple) and BFGS (green) for $K = 100$.

In this plot, we see BFGS no longer exemplifies super-linear convergence as it did previously. This is perhaps justified by the Hessian being better approximated for smaller values of K.

In addition to comparing the number of iterations, we also compare computation time for the three methods for various numbers of rectangles (with $a = 2$). Recall that the initial guess has the x_k spaced equidistantly along the ellipse in the first quadrant. We use the stopping conditions mentioned in Section 4; that is, we terminate when the change in the objective function less than 10^{-15} or norm of the gradient is less than 10^{-8}.

TABLE 1. Computational Cost for Different Methods

K	**Newton (s)**	**BFGS (s)**	**Steepest Descent (s)**
5	0.038	0.068	0.255
10	0.039	0.241	0.742
25	0.049	1.198	9.311
50	0.051	4.193	56.87

Again, Newton's method performs best in terms of computational cost — which is reasonable given that the linear system being solved is sparse.

Unfortunately, the computation time (as with the number of iterations) for BFGS and steepest descent increased rapidly, suggesting that these methods do not scale as well for large K. The slow convergence rate of steepest descent can occur even when the Hessian is reasonably well-conditioned (as formalized in [3] Section 3.3 Theorem 3.4). Furthermore, BFGS involves solving a large linear system for K large — at first glance, this seems to be mitigated by the Hessian being sparse. However, while BFGS preserves positive definiteness, the same does not hold for sparsity; that is, the approximation obtained at every step is *not* necessarily sparse ([3] Sections 6.1 - 6.2).

Consequently, Newton's method was used for very large K (> 100); some results are provided in the table below for $a = 4$ as an example. We can see that the optimal area is bounded below by the actual area of the ellipse (which is 4π in this case).

TABLE 2. Optimal Area for Various K and Computational Cost ($a = 4$) with Newton's Method

K	**Optimal Area**	**Run Time (s)**	**Number of Iterations**
10	13.0656	0.021	6
10^2	12.6104	0.06	7
10^4	12.5680	0.26	8
10^8	12.5664	158	8

We therefore conclude that Newton's method remains robust for minimizing the objective function even when K is large.

6. Conclusion

The aim of this project was to cover an ellipse $\mathcal{E}$ with K rectangles such that the area between the rectangles and $\mathcal{E}$ was minimized. The problem was formulated as

$$\begin{aligned} \min_{x_1,\cdots,x_K\in\mathbb{R}} \quad & \sum_{k=0}^{K-1}(x_{k+1}-x_k)h(x_k) \overset{\text{call}}{:=} f(\vec{x}) \\ \text{such that} \quad & 0 = x_0 < x_1 \leq \cdots \leq x_{K-1} \leq x_K = a \end{aligned}$$

where the rectangles $\mathcal{R}_1, \ldots, \mathcal{R}_K$ were assumed to have corners of the form

$$(x_k, y_k), (-x_k, y_k), (-x_k, -y_k), (x_k, -y_k).$$

Theorem 2.1 provided a necessary condition on the y_k given the ordering

$$0 = x_0 < x_1 \leq \cdots \leq x_{K-1} \leq x_K = a$$

which was implemented in the line-search algorithm. The initial guess was that the x_k were spaced corresponding to equidistant arc lengths on the ellipse in the first quadrant, as depicted in Figure 4; however, further work could explore altering this initial guess. The optimization problem was solved by Newton's method, steepest descent, and BFGS with the linear constraints enforced by bounding the step length α in the line-search algorithm. Each of these three methods was examined in terms of accuracy, the number of line-search steps, the number of iterations, and computation time. Steepest descent and BFGS overall performed less successfully than Newton's method did; this is probably explained by the slow convergence rate of steepest descent (even when the Hessian is reasonably well-conditioned) and BFGS not necessarily preserving the sparsity of the Hessian when approximating in each iteration. The accuracy and efficiency of Newton's method likely results from the sparse Hessian and accurate second-order approximation of the objective function.

7. References

(1) Gill, P., Murray, W. & Wright, M. (1982) *Practical Optimization.* Emerald Group Publishing Limited.
(2) Griva, I., Nash, S. & Sofer, A. (2009) *Linear and Nonlinear Optimization.* SIAM Press.
(3) Nocedal, J. & Wright, S. J. (2006). *Numerical Optimization.* Springer.