Optimization Project

AE 3340

Madeleine Graham

May 1, 2023

Unconstrained Optimization

1. The critical points of the Rosenbrock Valley Function are:

a) The Rosenbrock Valley Function: $f(x_1, x_2) = 100(x_2 - x_1)^2 + (1 - x_1)^2$

b) The first derivative of (a) w.r.t x1: $\frac{\partial f}{\partial x_1} = 202x_1 - 200x_2 - 2$

c) The first derivative of (a) w.r.t x2: $\frac{\partial f}{\partial x_2} = 200x_2 - 200x_1$

d) Setting b) equal to 0 to find critical points: $0 = 202x_1 - 200x_2 - 2$

e) Solving d) for x1: $x_1 = \frac{200x_2+2}{202}$

f) Setting (c) equal to 0: $0 = 200x_2 - 200x_1$

g) solving for x2: $-200x_2 = -200\left(\frac{200x_2+2}{202}\right)$

continued: $-1.98x_2 = -1.98$

continued: $x_2 = 1$

h) plugging (g) into (e) $x_1 = 1$

i) Finding the critical point $f(1,1) = 100(1-1)^2 + (1-1)^2 = 0$

j) finding the derivative of (b) w.r.t x1: $\frac{\partial^2 f}{\partial x_1^2} = 202$

k) The derivative of (c) w.r.t x2: $\frac{\partial^2 f}{\partial x_2^2} = 200$

l) The derivative of (b) or (c) w.r.t x2 or x1: $\frac{\partial^2 f}{\partial x_1 x_2} = -200$

m) The Hessian of (a): $\begin{bmatrix} 202 & -200 \\ -200 & 200 \end{bmatrix}$

n) Finding the determinate of (m): $\det\left(\begin{bmatrix} 202 & -200 \\ -200 & 200 \end{bmatrix}\right) = 40{,}400 - 40{,}000 = 400$

Because the determinate of the Hessian of this function is positive for all x1 and x2, the function (a) is universally concave-up, and therefore **its critical point at (1,1,0)** is a **global minimum.**

2. A strict Newtonian line-search method finds the direction of steepest descent starting at a given set of inputs and finds the minimum in that direction by testing values at arbitrary distances until one can create a polynomial fit based on three points in the slice in the direction of steepest descent. By creating the polynomial, one can approximate minimum value by differentiating at the estimated minimum. Then, a new starting position is established and the process iterates.
   However, one weakness of this method is that it takes several iterations to find the minimum of a function with a long valley, such as Rosenbrock's Valley function. Another weakness is that

this method calculates the Hessian matrix H numerically, which involves a lot of computation. Quasi-Newton methods allow for the H to be estimated using the following equations:

(1) A quadratic model of the problem: $\min_x \frac{1}{2} x^T H x + c^T x + b$

H is the positive definite symmetric Hessian matrix, c is a vector of constants, and b is a constant

(2) Solving for x* by setting the gradient at x* = 0 $\nabla f(x^*) = Hx^* + c = 0$

(3) Solution to (2) $x^* = -H^{-1}c$

Basically the quasi-Newtonian method predicts that it is in the middle of a valley and so it is able to create new iterations basically along a topo curve whilst looking for the global minimum.

3.

a)

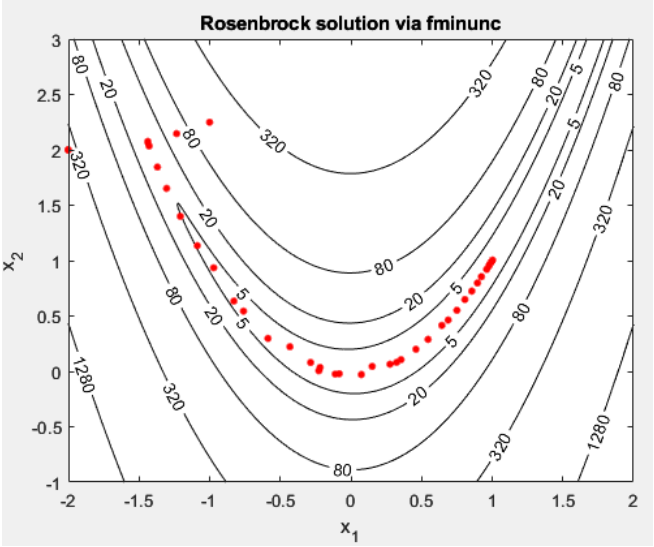


Figure 1: Rosenbrock solution using the Estimated Derivative method starting at [-2,2]

The Estimated Derivative method evaluated the function 156 times, used 40 solver iterations, and had an exit flag of 1. An exit flag of 1 means that the magnitude of the gradient is smaller than the OptimalityTolerance value, which means that an optimum was successfully found. From its starting point at [-2,2], it found points along a slice in the upper-right direction and determined the local minimum to be at a point on the valley contour. Then it was able to navigate through the "valley" to finally arrive at the solution. This is possible because it was iterating through local Hessian matrices, which showed there to be a curve, and so it was able to predict semi-accurately where the next lower points were going to be.

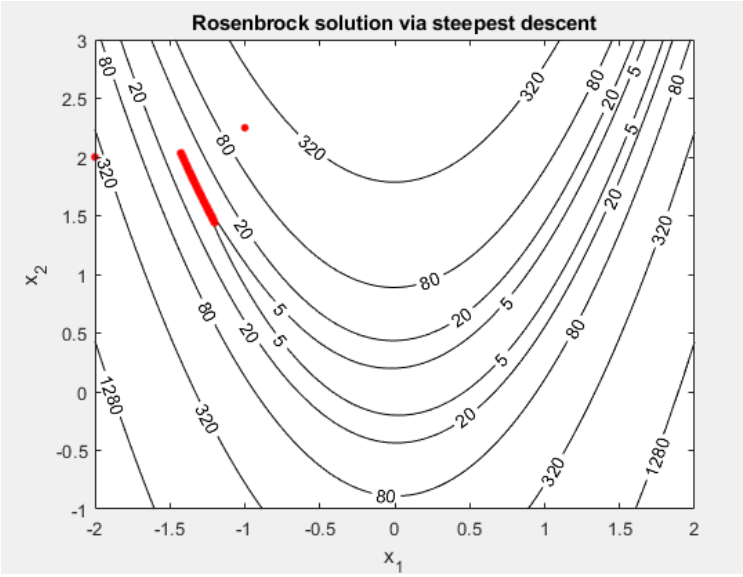


Figure 2: The Rosenbrock solution via the Steepest Descent Method starting at [-2,2]

The Steepest Descent Method evaluated the function 600 times, iterated the solver 45 times, and had an exit flag of 0. An exit flag of 0 means that the number of iterations exceeded the maximum allowable number of iterations. For the first step, the method was able to find the minimum in the same way as the previous method, but then what happened was it did not have enough information about the surroundings to try to iterate further along the valley – it kept running into more and more minimums, so it had to go very slowly and timed itself out, not even making it to the solution.

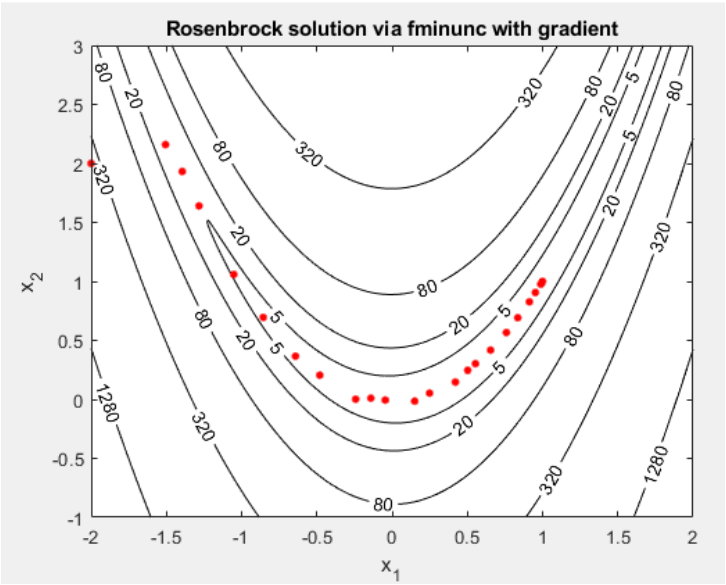


Figure 3: The Rosenbrock Solution via the Analytic Gradient method starting at [-2,2]

For the Analytic Gradient method, the number of function evaluations was only 32, with the number of solver iterations at 31 and an exit flag of 1. This method had the same exit flag as the Estimated Derivative method. This method plows right into the "valley" and has no trouble at all finding the curves. For this method, a gradient is provided, allowing a trust-gradient algorithm to be used, which is faster than the previous method by far.

b)

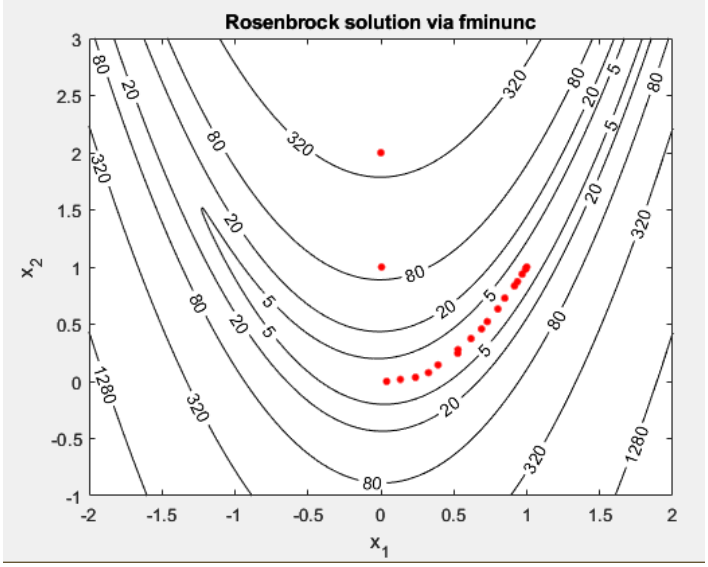


Figure 4: Estimated Derivative method starting at [0,2]

The Estimated Derivative method starting from x_0 = [0,2] this time resulted in 78 function evaluations, 20 iterations, with an exit flag of 1. This time the function didn't "overshoot" the minimum as in the same method starting from [-2,2]. It went straight down into the valley and stopped when it found the lowest point, then successfully followed the curve to reach the optimum solution. I think that it didn't overshoot to the other ridge because the "5" contour is fairly wide in this case. Starting at the new location was more efficient than starting in the old location, causing much fewer function evaluations in comparison.

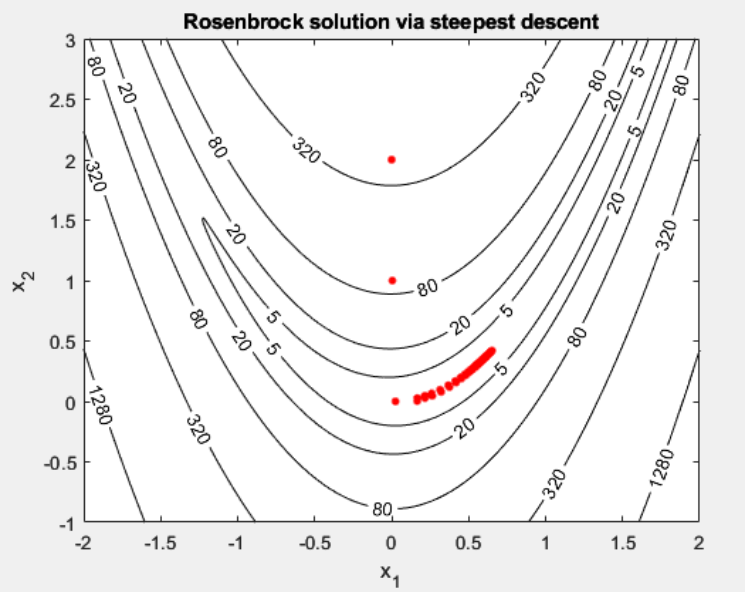


Figure 5: Steepest Descent method starting at [0,2]

The Steepest Descent method strikes again! At 600 function calls and 57 solver iterations, it still gave an exit flag of 0, and no optimum found. However, it is much closer to the optimal point than previously. It just has a difficult time with the curved valley as previously observed.

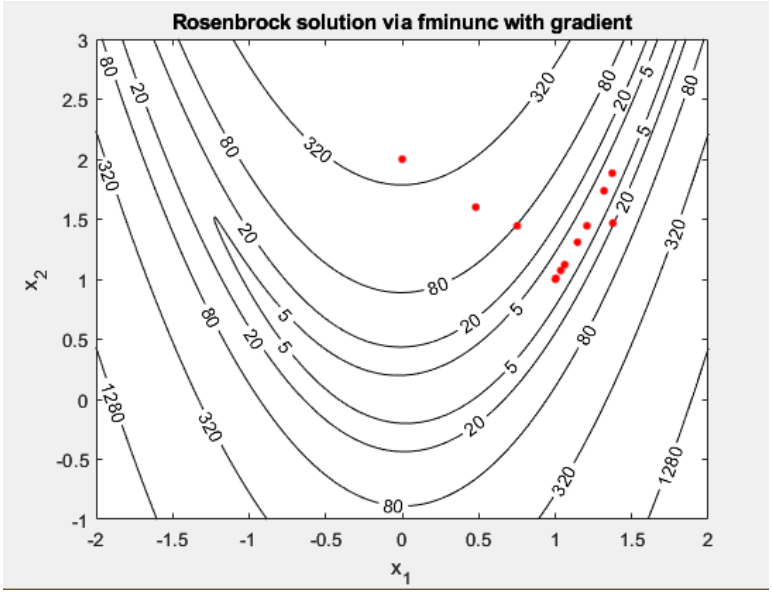


Figure 6: Analytic Gradient method starting at [0,2]

The Analytic Gradient method used 17 function evaluations, 16 solver iterations, with an exit flag of 3. The exit flag of 3 value means that the change in the objective function was less than the FunctionTolerance tolerance – this means that there was a satisfying optimum found. When this method was used from [-2,2], the exit flag was 1. An exit flag of 1 means that the magnitude of the

gradient is smaller than the OptimalityTolerance value. Perhaps the exit flag differed because the magnitude of the gradient was still larger than the OptimalityTolerance value, while the change in objective function was small enough to satisfy the exit requirements for exit flag 3.

The function calls did not head straight Southwards using this method. Instead, they veered off in the direction of the minimum, and therefore took way less time to get there. How is this possible? I think that at the topmost point, the Hessian was calculated and therefore the uneven curvature of the function could be known. The method allowed for function calls in the likely direction of the optimum solution. This is by far the best performance of fminunc.

4.

The objective function is to maximum the following:

$$f = \frac{L}{D}$$

Therefore, we need to minimize:

$$f = -\frac{L}{D}$$

Constraint equations are as follows:

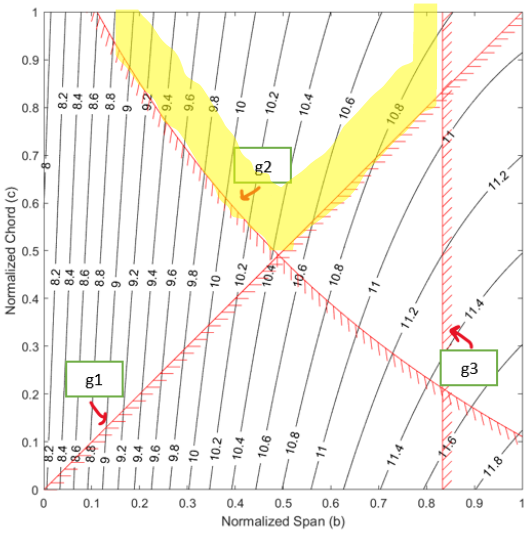
$$g_1: \ b - 10c \leq 0$$

$$g_2: \frac{W}{bc} - 40 \leq 0$$

$$g_3: b - 55 \leq 0$$

5.

The plot below is from LonDplot.m (provided). The zone highlighted in yellow is the boundary of feasibility and the lines hatched lines g1, g2, and g3 correspond to the above constraint equations where g = 0.



6.

Table 1: Number of iterations and function calls using different methods in the MATLAB function fmincon.m

Method	interior-point	trust-region-reflective	sqp	sqp-legacy	active-set
Number of Iterations	14	n/a	5	5	3
Number of Function evaluations	53	n/a	16	18	9
Exit Flag	1	n/a	1	1	1

```
>> LonDoptim
Error using fmincon (line 478)
Option Algorithm = 'trust-region-reflective' but this algorithm does not solve problems with the constraints you have
specified. Run a different algorithm by setting the Algorithm option. For information on applicable algorithms, see
Choosing the Algorithm in the documentation.

Error in LonDoptim (line 57)
[xopt,fopt,exitflag,output,lambda,gradf,hessianf] = fmincon(fun,x0,A,rhs,[],[],lb,ub,nonlincon,options)
```

From Table 1 it is obvious that with this kind of function, the 'active-set' method is the best way to go. For the 'trust-region-reflective' method, I received the error above in red.

From reading the documentation, the reason that the 'trust-region-reflective' method did not work was because there was not a gradient supplied in the objective function. I can surmise from this that the method works by analyzing the gradient supplied and establishing a "trust region," where the function knows what the direction all around it is.

The 'active-set' method was the fastest and most effective in this case because it took larger leaps than the 'sqp' method.

7.

a)
xopt = [55, 5.5]

$$g_1: \ 55 - 10(5.5) \leq 0$$
$$g_1: 0 \leq 0$$

$$g_2: \frac{8000}{302.5} - 40 \leq 0$$
$$g_2: -13.55 \leq 0$$

$$g_3: 55 - 55 \leq 0$$

$$g_3: 0 \leq 0$$

All the above are satisfied.

b) g1 and g3 are active because they are equal o at the optimum point, xopt.

c)

λ_1 = 0.0272

λ_2 = 0

λ_3 = 0.0146

All are positive or zero.

Lamda should be positive for an optimal solution, so they are appropriate.

d)

Complementary slackness condition:

$$\lambda_j g_j = 0$$

Check to see that this condition is satisfied for all λ:

$$\lambda_1 g_1 = (0.0272)(0) = 0$$
$$\lambda_2 g_2 = (0)(-13.55) = 0$$
$$\lambda_3 g_3 = (0.0146)(0) = 0$$

e) Compute the gradient of the active constraint(s) at "xopt".

The active constraints are g1 and g3. The gradients are:

$$\nabla g_1(55,5.5) = [1, -10]$$

$$\nabla g_3(55,5.5) = [1,0]$$

f) Compute the gradient of the Lagrangian function by using your answer to part (e) and the "gradf" result returned by MATLAB.

KKT Condition 3:

$$\nabla_x \mathcal{L}(x^*, \lambda) = "gradf" + \lambda_1 \nabla_x g_1(x^*) + \lambda_2 \nabla_x g_2(x^*) + \lambda_3 \nabla_x g_3(x^*) = 0$$

$$\nabla_x \mathcal{L}([55{,}5.5], \lambda) = [-0.0418, 0.2723] + (0.0272) * [1, -10] + (0) + (0.0146) * [1{,}0] = 0$$

$$[-0.0000391, 0.0002769] \approx 0$$

g)

There are 3 KKT conditions: "xopt" is feasible, the criterion for complimentary slackness is met, and that the gradient of the Lagrangian function is equal to zero. A) proves that "xopt" is feasible. D) proves that the complimentary slackness condition is met. F) proves that the gradient of the Lagrangian function is approximately equal to zero.

The answer to f is so close to zero that I would say that all 3 KKT conditions are met.

8.

Graphical interpretation of the gradient of the Lagrangian:

The figure below shows the two active constraint vectors in red and the gradient vector in blue. In order for "xopt" to be a constrained minimum, the blue vector must fall within the "convex cone" defined by the active constraint vectors. In this 2-d representation, a "convex cone" is more like triangle. As you can see clearly in the figure below, the blue gradient vector falls between within the small angle created by the two active constraint vectors. This illustrates that "xopt" satisfies all the conditions to be the constrained minimum solution of our problem.