Mark Paluta
AA222 Multidisciplinary Optimization

Project 2: Constrained Optimization

1. Methodology

This optimization project utilized Newton's method. This algorithm uses information from both the derivative and Hessian, and as neither were allowed to be computed symbolically, they were estimated using finite differences.

The constraints for the problem were built into the test function using penalty functions, thus creating an unconstrained problem for optimization. The specific penalty chosen was a quadratic penalty. Usually, this penalty method weights the penalty with a constant weight while iterating several times until convergence, then changes the weighting of the penalty to a heavier value while running a subsequent optimization. A variation on this idea was run. Since Newton's method with the full Hessian calculation is heavy with function calls, this algorithm is rather limited in how many iterations it is allowed. Thus, the weight of the penalty was increased with every single iteration, and only a single optimization path was run. This ran the risk of increasing penalty weight too rapidly, but at the same time, allowed sufficient increase in weight before all function calls were expended.

2. Algorithm Paths for Example Functions

To visualize the algorithm, the process was run on two functions. These functions are explicitly written out in section 3. The first plot below shows the path of three starting points as they make their way toward the optimum point, marked by the red dot. The thin black lines illustrate the constraint functions.

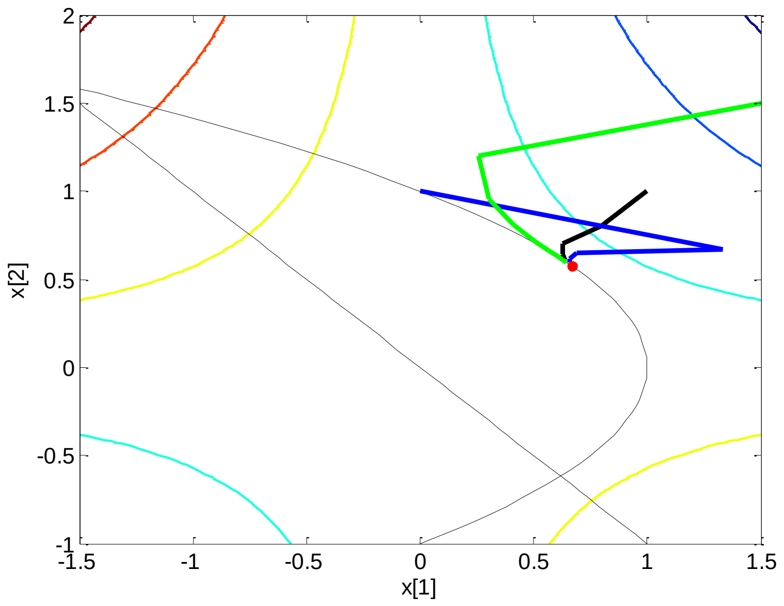

Figure 1. Function 1 and three paths of convergence.

The three points chosen above were (1, 1), (0, 1), and (1.5, 1.5). These paths correspond to approximately 100 function evaluations.

The second function was evaluated at the points (-2, 2), (-1, 2), and (3, 1.5). The plot below shows how they converge given approximately 100 function evaluations.

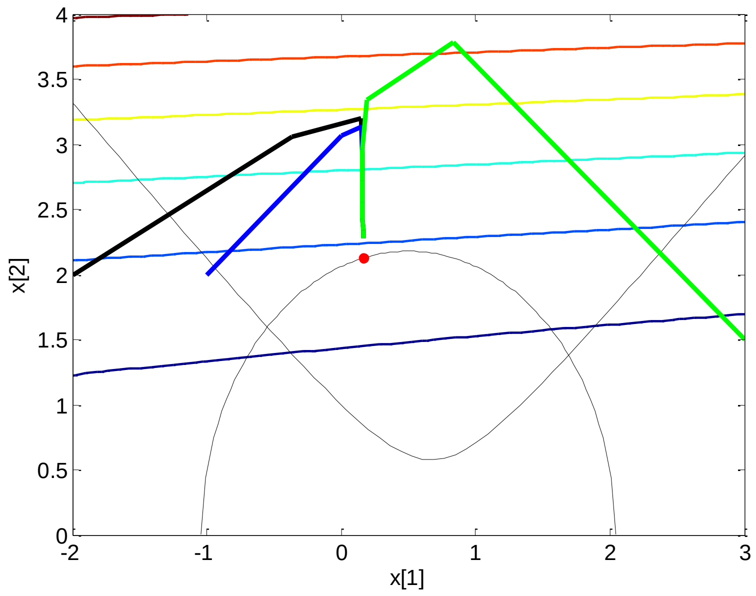

Figure 2. Function 2 and three paths of convergence.

3. Convergence Data for Example Functions

The two functions used above are explicitly outlined below, as well as a third. All three were used to run convergence tests, as described in this section.

Function 1:

$$f = -x_1 x_2$$

$$g_1 = x_1 + x_2^2 - 1 \leq 0$$

$$g_2 = -x_1 - x_2 \leq 0$$

$$x^* = (\frac{2}{3}, \frac{1}{\sqrt{3}})^T, f = -\frac{2}{3\sqrt{3}}$$

Function 2:

$$f = 4x_1^2 - x_1 - x_2 - 2.5$$

$$g_1 = -x_2^2 + 1.5x_1^2 - 2x_1 + 1 \leq 0$$

$$g_2 = x_2^2 + 2x_1^2 - 2x_1 - 4.25 \leq 0$$

$$x^* \approx (0.164439, 2.12716)^T, f \approx -4.68344$$

Function 3:

$$f = x_1 - 2x_2 + x_3$$

$$g_1 = x_1^2 + x_2^2 + x_3^2 - 1 \leq 0$$

$$x^* = (-\frac{1}{\sqrt{6}}, \sqrt{\frac{2}{3}}, -\frac{1}{\sqrt{6}})^T, f = -\sqrt{6}$$

For function 1, a starting point of (1, 1) was used. The convergence is shown in Figure 3 below.

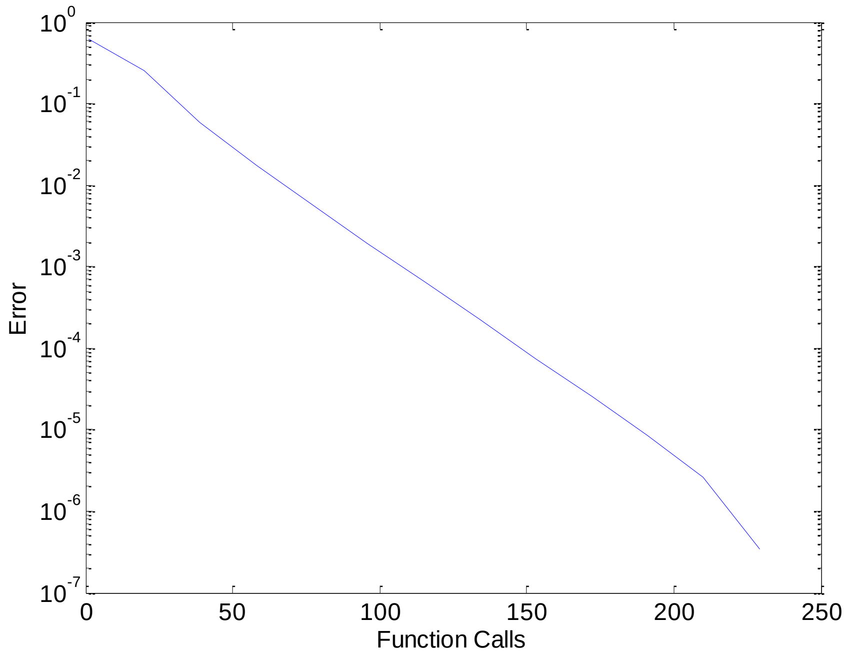

Figure 3. Convergence of Function 1.

This convergence is nearly linear in a semi-log space and is converging at a rate of approximately one decade per 40 function calls.

For function 2, (-2, 2) was used as the test point. Figure 4 shows the convergence data.

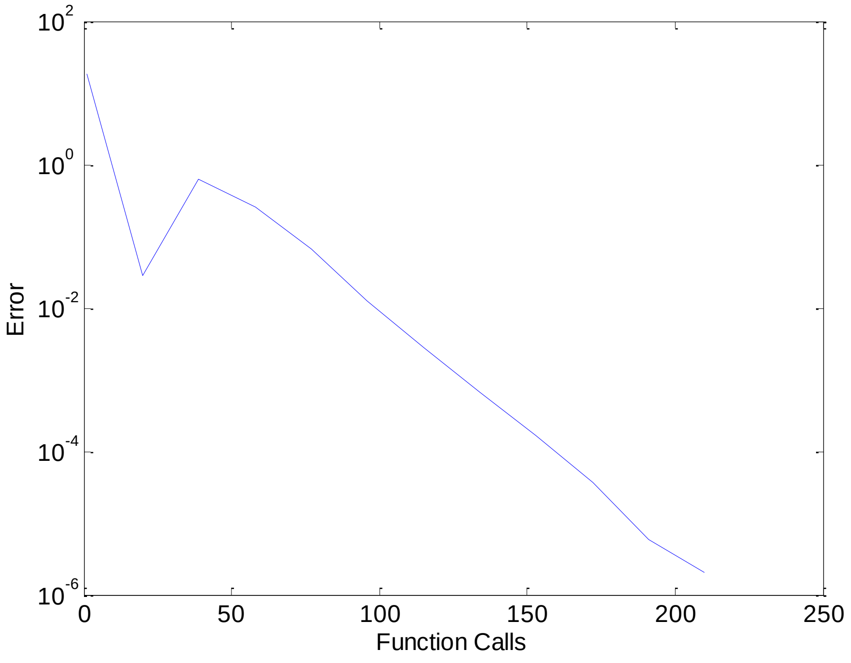

Figure 4. Convergence of Function 2.

This function converges at a rate of approximately one decade per 30 function calls. It is interesting in that it travels in the wrong direction early on. This can be attributed to the penalty function. Early on the constraint penalization had low weighting, allowing a point with small error in f(x), but this point was unfeasible. As the penalty was given more weight, it then prioritized feasibility over functional error, so it jumped to a larger error before it continued on its path toward the optimum.

Function 3 was tested at (-0.5, 1, -1). Its convergence plot is shown below in Figure 5.

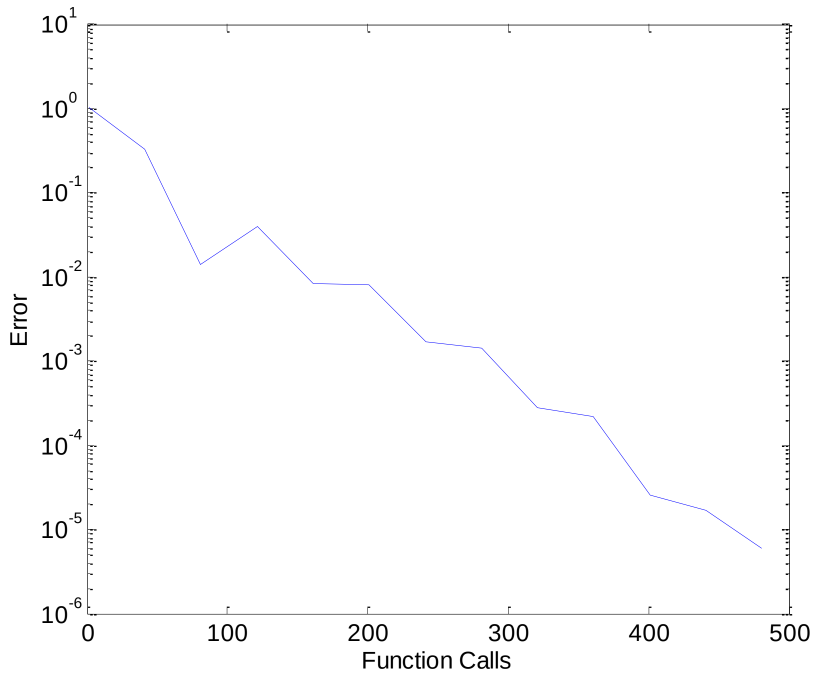

Figure 5. Convergence of Function 3.

This function also exhibits some “wrong direction” behavior, explained by the need to satisfy the penalty function which incurred more weight in later iterations. Its convergence is approximately one decade per 90 function calls.

4. Conclusion

This algorithm is a useful solution for these constrained optimization problems. Its rate of convergence was more impressive in 2 dimensional cases than in the 3 dimensional case. It would be interesting to use a more efficient algorithm than Newton’s method while still using the strategy of increasing the weight of the penalty function every iteration. This would allow for an algorithm that was less taxing on function calls but still may exhibit a more rapid convergence behavior than the standard process of optimizing with a single penalty weight, then changing the weight and optimizing again.