Report 4

Rob Rau

March 12, 2014

First, the Easy Stuff

Given the function

$$f(x) = x_1^2 - x_2 \tag{1.1}$$

and constraints

$$c_1(x) = 2x_1 - x_2 \geq 0 \tag{1.2}$$
$$c_2(x) = x_1 + x_2 - 1 \geq 0 \tag{1.3}$$

the Lagrangian takes the form of

$$\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2, s_1, s_2) = f(x_1, x_2) - \left(\lambda_1 \left[c_1(x_1, x_2) - s_1^2\right] + \lambda_2 \left[c_2(x_1, x_2) - s_2^2\right]\right) \tag{1.4}$$

which can be expanded to

$$\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2, s_1, s_2) = x_1^2 - x_2 - \left(\lambda_1 \left[2x_1 - x_2 - s_1^2\right] + \lambda_2 \left[x_1 + x_2 - 1 - s_2^2\right]\right) \tag{1.5}$$

The following 6 equations can be set up to solve for all 6 unknowns:

$$\frac{\partial \mathcal{L}}{\partial x_1} = 2x_1 - 2\lambda_1 - \lambda_2 = 0 \tag{1.6}$$
$$\frac{\partial \mathcal{L}}{\partial x_2} = \lambda_1 - \lambda_2 - 1 = 0 \tag{1.7}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_1} = 2x_1 - x_2 - s_1^2 = 0 \tag{1.8}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_2} = x_1 + x_2 - 1 - s_2^2 = 0 \tag{1.9}$$
$$\frac{\partial \mathcal{L}}{\partial s_1} = \lambda_1 s_1 = 0 \tag{1.10}$$
$$\frac{\partial \mathcal{L}}{\partial s_2} = \lambda_2 s_2 = 0 \tag{1.11}$$

When solved for, there are 5 possible solutions

	x_1	x_2	λ_1	λ_2	s_1	s_2
1	1	2	1	0	0	$\sqrt{2}$
2	1	2	1	0	0	$-\sqrt{2}$
3	1/3	2/3	5/9	-4/9	0	0
4	-1/2	3/2	0	-1	$\frac{i\sqrt{10}}{2}$	0
5	-1/2	3/2	0	-1	$-\frac{i\sqrt{10}}{2}$	0

To fully satisfy the first order KKT conditions $\lambda_k \geq 0$. This leaves only the first and second solution as possible optimal solutions. Just by observation we see both solutions are identical as s_2 is squared in the Lagrangian.

To determine positive definiteness of the Hessian of the Lagrangian, the active constraint must satisfy $\nabla c_k^T(x^*)y = 0$ for some $y \neq 0$. For this solution the only active constraint is c_1, so

$$\nabla c_1^T(x^*)y = \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 0 \tag{1.12}$$

Just by inspection, we can see that $y_1 = 1$ and $y_2 = 2$, thus satisfying the second order conditions for a constrained problem.

Therefore the optimal point of the constrained problem is $x^* = \begin{bmatrix} 1 & 2 \end{bmatrix}$.

Now for the Hard Stuff

Logarithmic Interior Penalty

The logarithmic penalty function, conceptually isn't to difficult to implement. The core of my implementation is the BFGS optimizer I wrote for assignment 3, completely unmodified. The line search that BFGS uses did need to be modified somewhat to better handle the infinite values seen with the logarithmic penalty. This modification simply made it backtrack if infinity was found.

The other issue I had was when the start point happened to be outside of the feasible region. This would cause the line search to choke, as all it saw were infinite values, it didn't have anyplace to go. To solve this, I simply performed a line search on the constraint function in the direction of steepest ascent. This would move the start point into the feasible region, and the BFGS optimizer would handle it from there. I do not know how this method would work when there are multiple constraints. I did not have time to study this. If I were to hazard a guess, I would try averaging the gradients of each constraint and use that as the search direction instead.

When toying around with this optimizer I found two things had the largest effect on the convergence rate. The initial size of μ and how fast μ is decreased. The larger the initial μ more major iterations are needed to converge. On the surface this doesn't sound ideal, but

the fact of the matter is that the interior penalty method is an algorithm layered on top of two more algorithms. I found that some of the larger initial μ values were actually better for the underlying BFGS implementation. I imagine this is problem dependent, and would probably need to be tuned for different problem spaces.

Changing how much μ was decreased seemed to have the largest effect. I found that decreasing μ a large amount each iteration, in general, would allow for much faster convergence. If, however, μ got to small to fast, it would start to cause problems for the BFGS optimizer. Another thing I noticed was that this method has a constant number of major iterations regardless of start point.

The interior penalty method ended up converging on the point $AR = 13.60185$ and $S_{ref} = 12.54455$ with a drag value of $D = 258.74594N$.

Sequential Quadratic Programming

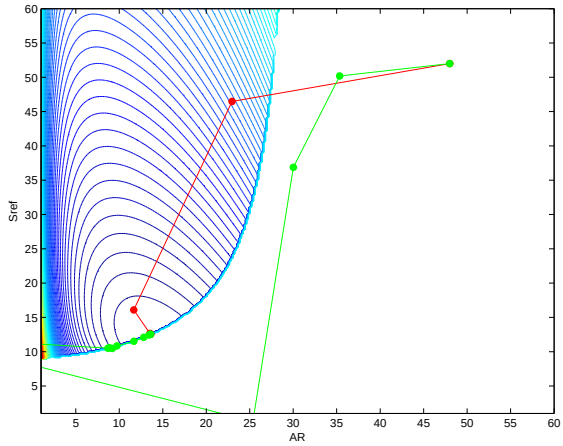
We can see from the result given by the interior penalty method, that the constraint was active. Because of this, we can treat that inequality constraint as an equality constraint to use with the SQP method.

I found SQP to be more robust than the interior penalty method. It took less fussing around with parameters to achieve good results. It is also much faster than the interior penalty method. I found that on average it would take more major iterations to converge, it still spent much less computational time to find the result. The reason for this is because the SQP algorithm does not spend nearly as much time in minor iterations. I found that, for this problem at least, the line search in SQP was almost never run. This will most likely not be the case with a different problem set, but for this one it seems to be.

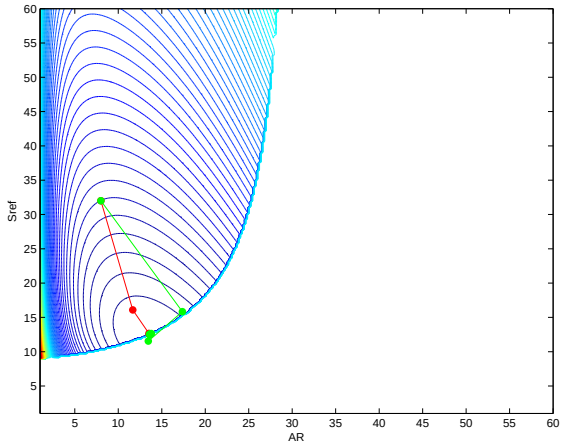
The obvious disadvantage SQP has is that it is only built to handle equality constraints without major modifications. This also means that it does not have the same problem that the interior penalty method has when starting outside of the feasible region.

The following plots show the paths of travel for both the interior penalty method as well as SQP. As fig. 1 shows, SQP first couple of moves take it to the equality constraint. From there it follows the constraint until it finds the minimum point.

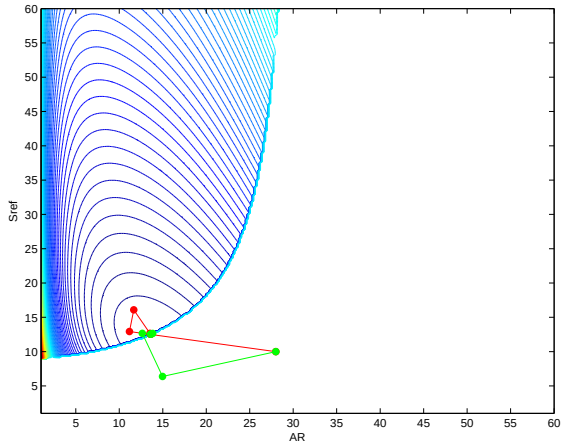
While SQP is faster than the interior penalty method, it has some major drawbacks. The fact that it can't handle inequality constraints was already mentioned, but further than that theres the fact that it does spend more time outside of the feasible region than the interior penalty method. This means that if the simulation is needed to be stopped early, the solution may be infeasible.



(a) Drag minimization with start point of $AR = 48$, $S_{ref} = 52$



(b) Drag minimization with start point of $AR = 8$, $S_{ref} = 32$



(c) Drag minimization with start point of $AR = 28$, $S_{ref} = 10$

Figure 1: Comparison logarithmic interior penalty and SQP

Final Thoughts

Both methods have their merits. SQP is fast, but only for equality constraints, while interior penalties are slower, but work with inequalities and guaranty feasibility.

I also learned an import lesson about the complex step derivative method. When trying to debug why my optimizers would go off in the weeds, I discovered that when they would travel into negative values for AR and S_{ref} the drag function would produce imaginary numbers due to the square roots involved in the computing the span of the wing. This caused the complex step method to produce garbage results and thus threw the optimizers off. One needs to be careful when using the complex step method as to not fall into the same trap I did.