Optimization Homework 6

Dustan Kraus

April 6, 2016

1 Introduction

This assignment had multiple purposes. First, I needed to make more progress on our project. The second purpose was to be exposed to optimization under uncertainty. Finally, I chose the option of learning about surrogate-based optimization rather than optimization framework.

2 Problem 1

2.1 Problem Summary

Finish strong on your project. Focus on trying to finish up all optimization results. Make more progress on the project and begin to finalize the project.

2.2 Problem Solution

During this homework period, we made significant progress on our project. We finished the conversion to a surrogate wake model, and the optimization runs much more quickly on the surrogate than it did off of the CFD data. Upon further investigation, and upon Dr. Ning's suggestion, we changed our constraint function. We now only require the turbines to be 3 radii apart, rather than 8 radii. This is more representative of real wind farms. In addition, our git repository was getting very cluttered with all of our individual files in it. So, we cleaned up the git repository by making more folders, and added a main objective python script so that we could all run optimizations on the same objective function.

Finally, we ran optimizations with more design variables, and with initial turbine positions that were more realistic. During these optimizations, we constrained the turbines to remain inside of the same bounding box as the initial configuration so the optimization was doing more than simply spreading the turbines out.

I ran an optimization with 5 HAWTs and 20 VAWTS with an initial grid configuration as shown below in figure 1. I then optimized this layout using SNOPT with Dr. Ning's pyopt wrapper. The optimized wind turbine locations are shown in figure 2. This optimization resulted in an annual energy production increase from 452.4 GW-hrs to 453.5 GW-hrs. It should be noted that the VAWT wake model does not propagate very far, which may account for the grouping of the VAWTs.

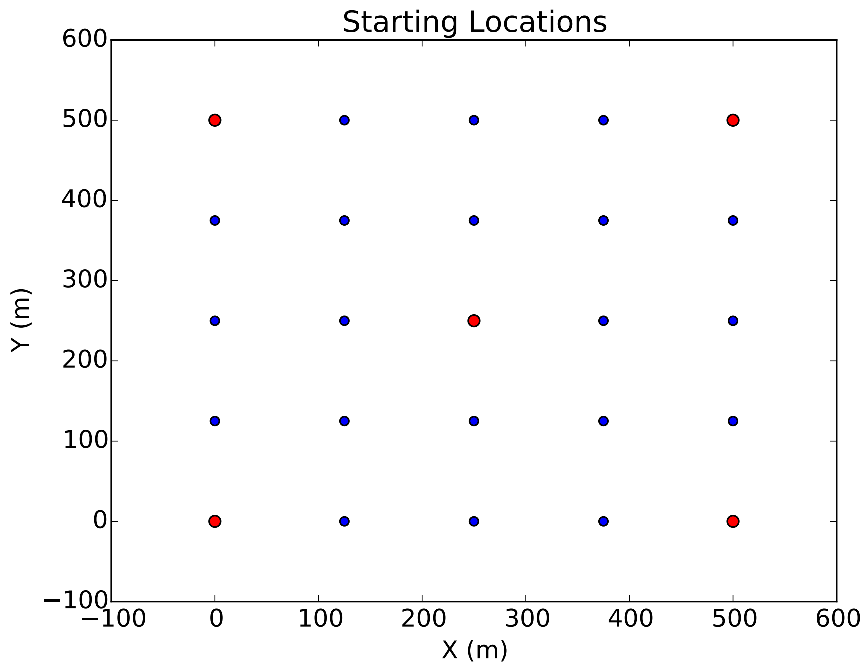

Figure 1: This figure shows the original wind farm layout which was optimized. The red dots are the HAWTs, while the blue dots are the VAWTs.

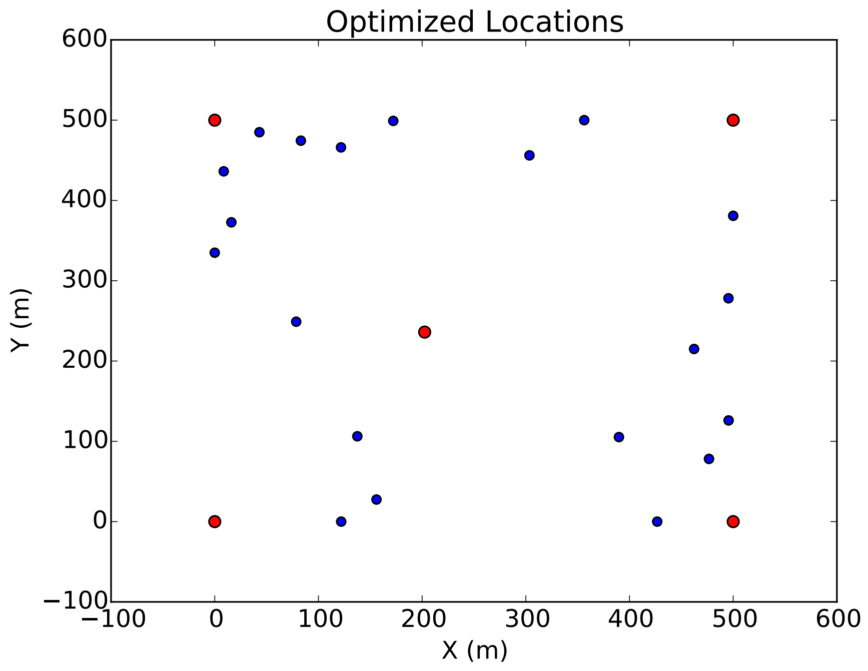

Figure 2: This figure shows the optimized layout of the wind farm shown in Figure 1. The red dots are the HAWTs, while the blue dots are the VAWTs.

3 Problem 2

3.1 Problem Summary

For the following problem:

$$\begin{aligned} \min \quad & f = x_1^2 + 2x_2^2 + 3x_3^2 \\ \text{s.t.} \quad & c_1 = 2x_1 + x_2 + 2x_3 \geq 6 \\ & c_2 = -5x_1 + x_2 + 3x_3 \leq -10 \end{aligned} \tag{1}$$

a Find the deterministic optimum.

b Find the worst-case, reliable optimum where $\Delta\ x_1 = \Delta\ x_2 = \pm 0.1$, $\Delta\ x_3 = \pm 0.05$

c Now, instead of the worst case tolerances, assume the variables are normally distributed with $\sigma_{x1} = \sigma_{x2} = \pm 0.033$,$\sigma_{x3} = \pm 0.0167$. Find the reliable optimum where the target constraint reliability is 99.865% for each constraint individually.

d Compare the total target reliability with a Monte Carlo simulation of reliability for all three approaches (using the normal distributions for the input variations).

e Briefly discuss any lessons learned.

3.2 Problem Solution

a To find the deterministic optimum, I simply coded the objective function defined above into a python function. I also redefined the constraints to be less than or equal to zero (as this is the form that the pyopt wrapper expects). I then passed this function into Dr. Ning's pyopt wrapper using the SNOPT optimizer. My results are below in table 1. As shown, I tested multiple different starting points and arrived at the same optimum.

	[x1, x2, x3]	f(x1, x2, x3)
Original	[1, 2, 3]	36.0
Optimal	[2.35, 0.38, 0.46]	6.45
Original	[6, 2, 7]	191
Optimal	[2.35, 0.38, 0.46]	6.45

Table 1: This table contains the original and optimal values and function values for the deterministic function with two different starting points.

b For this part of the problem, I first found the partial derivatives of the constraints with respect to each design variable and then multiplied by the worst case tolerance as described in the text. I then modified each constraint by adding $\Delta\ c_1$ and $\Delta\ c_2$ to each constraint respectively. I then used the same optimization procedure described in part a. The results I found are below in table 2 once again with multiple different starting points.

	[x1, x2, x3]	f(x1, x2, x3)
Original	[1, 2, 3]	36.0
Optimal	[2.52, 0.40, 0.48]	7.36
Original	[6, 2, 7]	191
Optimal	[2.52, 0.40, 0.48]	7.36

Table 2: This table contains the original and optimal values and function values for the worst case reliable function with two different starting points.

c For this part of the problem, I first calculated σ_{c_1} and σ_{c_2} using the procedure described in the text. I then found the k value corresponding to 99.865% reliability by using python's norm.ppf function (k = norm.ppf(0.99865)). I then added k*σ_{c_1} and k*σ_{c_2} to c_1 and c_2 respectively. I then followed the same optimization procedure described in part a. The results I found are below in table 3 once again with multiple different starting points.

	[x1, x2, x3]	f(x1, x2, x3)
Original	[1, 2, 3]	36.0
Optimal	[2.46, 0.38, 0.47]	7.02
Original	[6, 2, 7]	191
Optimal	[2.46, 0.38, 0.47]	7.02

Table 3: This table contains the original and optimal values and function values for the function with normally distributed variables with two different starting points.

d For this part, I first computed the function and constraint values for 1e6 points which were randomly selected from a normal distribution centered about the optimum (mean), with standard deviations of σ_{x1}, σ_{x2}, and σ_{x3}. I did this for each part (a, b, and c about the optimums in each part). I then counted the number of times neither constraint was violated and divided this by 1e6 to find the total target reliability. The results are below in table 4.

	Reliability (%)
Deterministic Optimum	34.73
Worst-Case reliable Optimum	99.9995
Normally distributed variable reliable optimum	99.74

Table 4: This table contains the total target reliability for the optimums found in parts a, b, and c of this problem. These reliabilities were found by running the Monte Carlo simulation multiple times with 1e6 samples, and then averaging the reliabilities found in each simulation.

e I learned quite a few lessons during this simulation. First, if I run an optimization assuming the design variables (or other variables) are deterministic, and they actually have variability, I will get a very low reliability. This is because with a deterministic optimum, I may be right on the edge of constraints. Which means a slight variability will mean a constraint violation.

Second, running an optimization with the extremes of the tolerances on every variable (situation b.) will result in a very reliable optimum; however, this also results in a worse objective function value because the optimum is too conservative.

Finally, running an optimization with a target reliability (as in part c.) given a distribution of the design variables results in a better objective function value than part b, while still maintaining the desired reliability.

4 Problem 3

4.1 Problem Summary

Minimize the drag of a supersonic body of revolution using a global polynomial surrogate model. The provided analysis code is somewhat noisy, similar to what would exist with experimental data or with some grid-based simulations, hence the use of a surrogate. In addition to presenting and discussing your results, discuss lessons learned and whether or not any of the concepts are relevant to your project.

4.2 Problem Solution

This problem really walked me through the process of surrogate based optimization. The CFD data takes a while to load and evaluate making a surrogate useful in this scenario. I used latin hypercube sampling with 20 samples to get an initial sample and construct my initial model parameters. I then ran through a loop 20 times where I optimized my parameters by comparing the error of my model (fhat) to the actual data (f). Once I had a satisfactory model, I ran an optimization to minimize the drag of the supersonic body using my surrogate model (which was much quicker than using the actual data).

The error when comparing the shape optimization using my surrogate model vs. the CFD data was 0.916%. The error of the optimization of the surrogate model to the theoretical solution was only 1.4%. The optimal solution of the surrogate model is shown below on the same plot as the theoretical solution (the Sears-Haack body) in figure 3.

During this part of the homework, I learned that even though a surrogate model is lower fidelity than actual data. The trade-off between computational efficiency and accuracy is often in favor of a surrogate model. In many cases, the accuracy is often so close that the computation advantages clearly outweigh the loss of accuracy.

These concepts most definitely apply to our project. In fact, we actually use a surrogate model to run our optimization. We were originally using CFD vorticity data to calculate the wake behind the VAWTs in our wind farm simulation, but this took a large amount of time to run. We have since moved to a simpler surrogate model of this vorticity data, and the speed of our simulation has drastically increased.

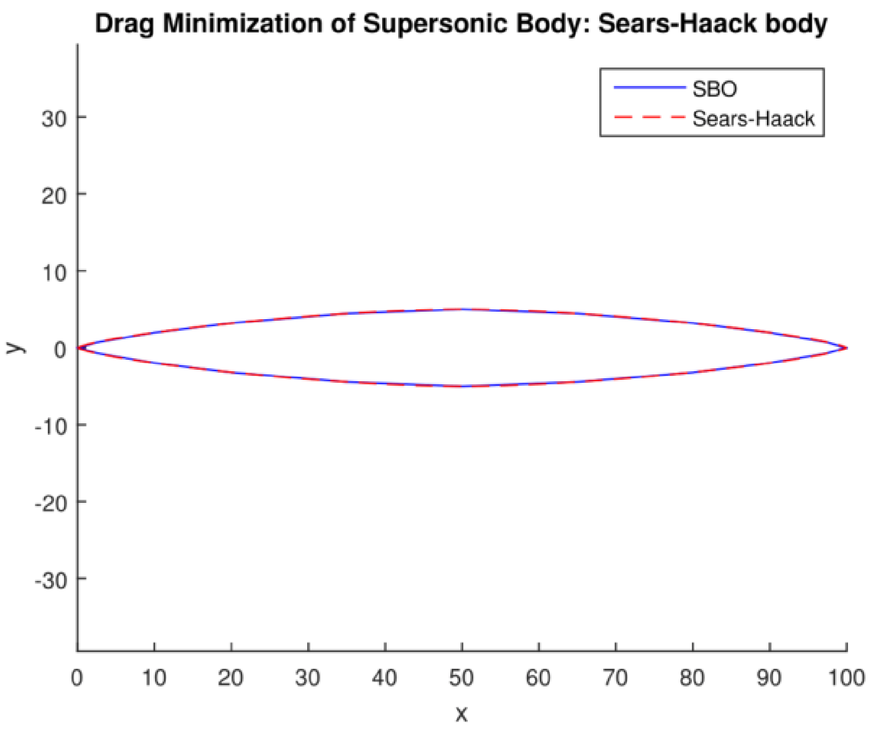

Figure 3: Optimal shape of a revolved body for supersonic drag reduction.