# ME 527 Assignment:

# Multi-Objective Optimisation of Expensive Functions

The aim of this assignment was to perform a multi-objective optimisation on a hidden function that relates to the drag coefficient and the cost of production of a vehicle. The aim was to find the minimum values of the drag coefficient and the cost and find the Pareto front representing these values. To achieve this, two methods were employed; the first was to perform a global optimisation of the true function using a multi-objective genetic algorithm with a budget of 50,000 function evaluations, and the second was to create a surrogate model of the function using the Kriging technique and then to perform a global optimisation of the surrogate model using a multi-objective genetic algorithm with a total budget of 600 evaluations of the true function.

## 1. Description of Strategies Used

### 1.1 Genetic Algorithm

The genetic algorithm function used to perform the global optimisation was the *Matlab* inbuilt function ‘*gamultiobj’*. This function uses a variant of the NSGA-II that favours members of the population with best fitness values, but also accepts members with lower fitness that can add diversity to the population, thus increasing the likelihood of convergence upon an optimal Pareto front [1]. Despite this, it is possible, and is likely with an arbitrary initial population, that the algorithm will converge on a sub-optimal Pareto front, as once convergence begins it is hard for the algorithm to explore for dominating solutions.

In order to ensure that the calculation budget was not exceeded, the maximum number of generations for the process had to be set. As each member of the population will have on average one function evaluation per generation, the maximum number of generations was found to be;

*N.B. the population size of 200 is the default for the ‘gamultiobj’ function in Matlab for problems with more than 5 variables.*

The *‘gamultiobj’* function outputs 70 non-dominated members of the final population, the values of the function, as well as the corresponding values of x, were then stored into a matrix comprising of the results from all 20 runs. This allowed for scatter plots of all 20 runs to be produced on the same figure for later comparison and evaluation.

### 1.2 Kriging

The chosen method of creating the surrogate model was Kriging. To create the model, the script provided for Tutorial 9 was adapted to be suitable for a 6 variable problem. As it was desired to be able to run both the non-surrogate and surrogate models within the same script, the Kriging script was changed to be a function. In order to produce reliable results, it was decided that the method of design of experiment should be changed from a Latin-hypercube to a full factorial of the boundaries (*‘xinit.m’*). The full factorial method requires 64 (26) of the available 600 true function evaluations, thus it was decided to be a suitable DOE.

The next element of the script that was required to be edited was the value of θ. As θ is a measure of the search radius for the interpolation of each of the variables, this must be scaled proportionally to each of the variables. This allows for more of the true function to be searched and produce a more accurate surrogate model. These values were chosen arbitrarily to produce an accurate model and further work could be done to optimise the values of θ for this problem.

The final change that had to be made to the function was to increase the number of new members that can be added to the population per calculation. Increasing this value has a double benefit; firstly, it decreases the overall time for the function to produce a final answer, secondly, it means that in the case that a large number of very accurate solutions are found, the majority of them are saved and are not lost as could be the case with a lower number of new members.

## 2. Results and Discussion

### 2.1. Genetic Algorithm

The GA was run twice under different initial conditions and were compared to a result found by running the GA with a very high number of allowable generations, shown as red stars in the graphs. It must be noted that the reference is not perfect as there is a large region of the Pareto front beyond CD = 0.195 that has been missed. The first run was performed using the default setting of having a random initial population. As was to be expected, the algorithm converged on a variety of sub-optimal fronts, as can be seen in Figure 1. This was because there is very little room for exploration built into the *‘gamultiobj’* function on *Matlab*, thus once the algorithm began to converge on a front, there was minimal exploration beyond that front to find a more optimal front.

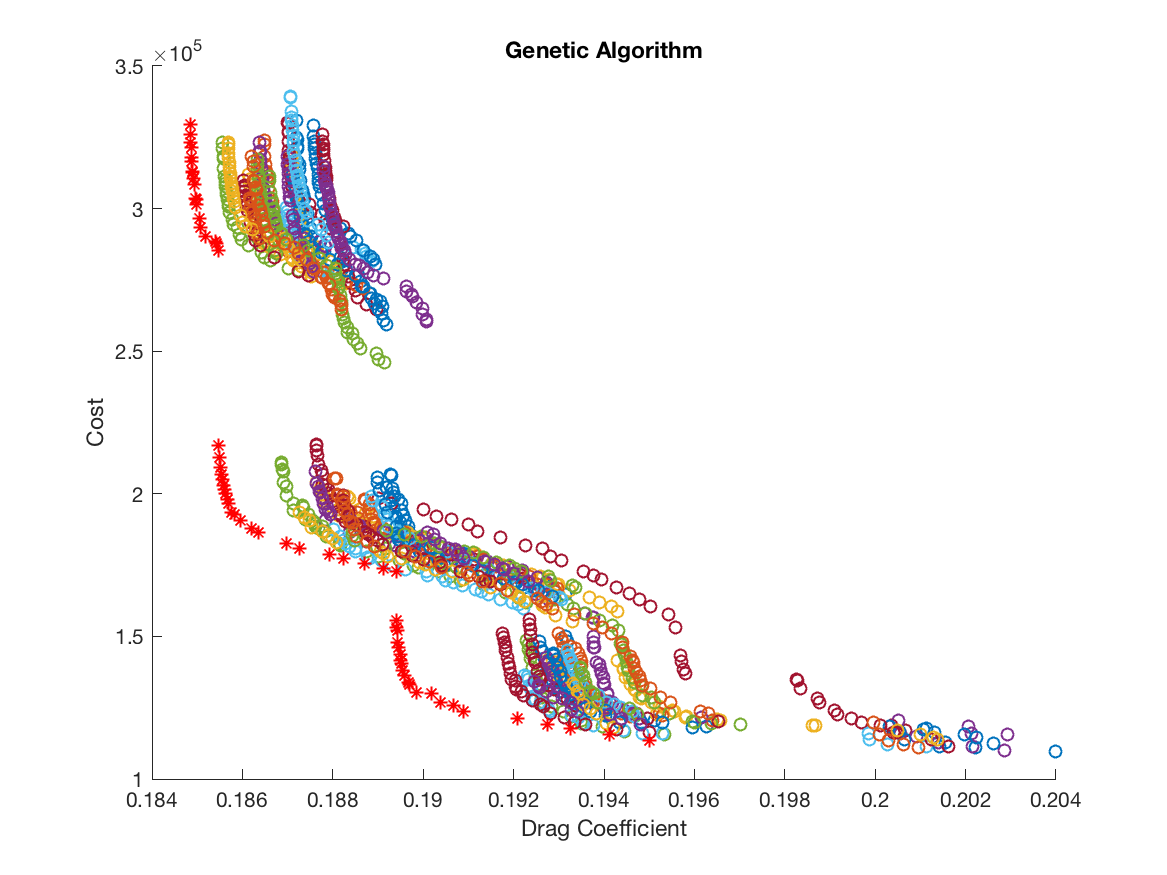


Figure 1 - Genetic Algorithm with Random Initial Population

Figure 2 shows the results obtained with an initial population determined by the full factorial DOE. The values of the initial population are represented by the black pentagrams. As there are three members of the initial population that are very close to the optimal solution, this forces the genetic algorithm to converge towards the optimal solution. Therefore, it can be concluded that it is advisable to use a proportion of the overall budget to perform a DOE before running the genetic algorithm.

### 2.2. Kriging

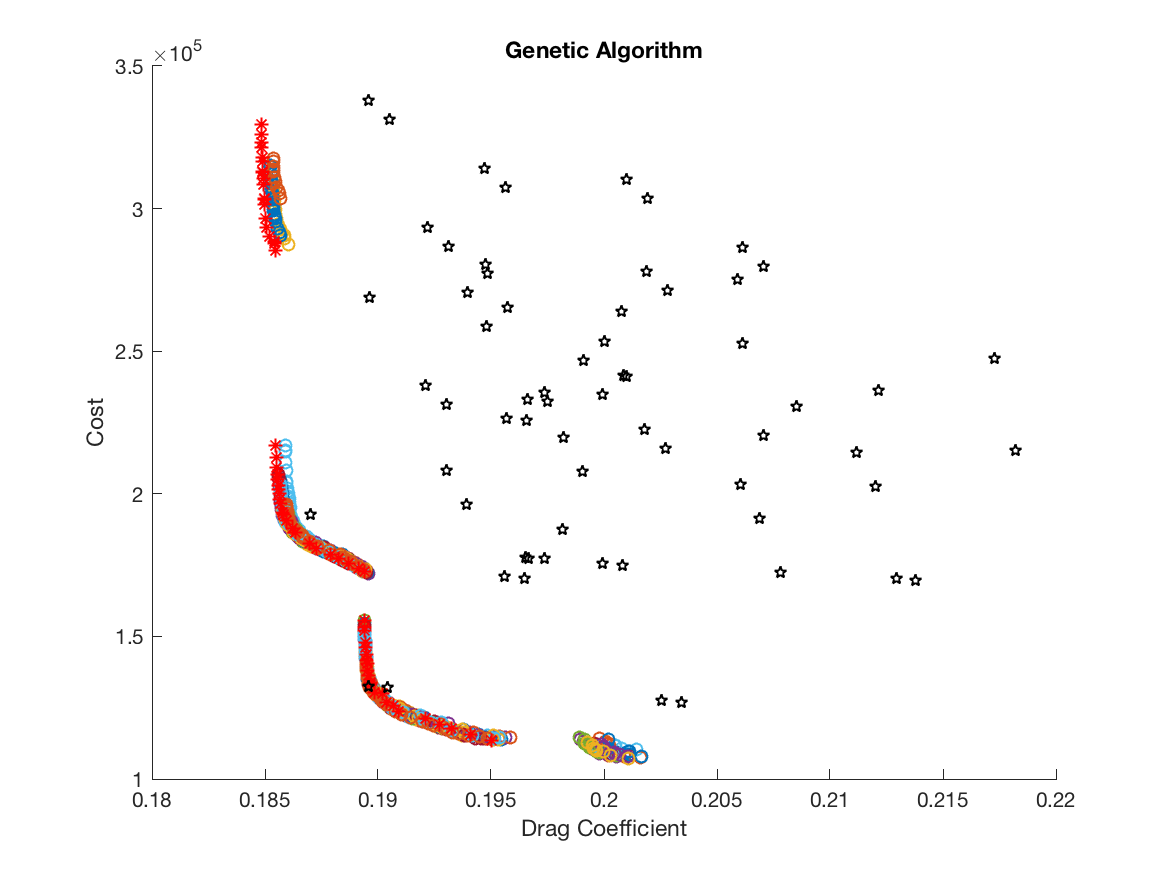


Figure 2 - Genetic Algorithm with Initial Population Determined by Full Factorial

As with the true function, the GA was used to solve the surrogate function both with a random initial population, see Figure 3, and with an initial population determined by a full factorial DOE, Figure 4. As before, the solutions acquired using the initial population were better compared to those acquired using a random initial population. However, it appears that when using an initial population, the algorithm is less likely to explore the area near the top front of the surrogate function than with the true function. Another difference in the results are between the solutions acquired with a random initial population. It can be seen in Figure 1 that despite not being at the ideal solution, the fronts predicted by the GA followed the same general shape as the optimal solution. However, in Figure 3 that there are no obvious individual fronts but rather a group of results near to the optimal solution. This can be accredited to the fact the surrogate model was built using a limited number of true function evaluations, so the finer details of the function are likely to have been missed. It must also be noted that solutions obtained using Kriging to create a surrogate model were more time intensive than using the GA on the true function for this problem, however, had the true function been expensive, then using Kriging to create a surrogate model could become less time expensive and be the preferred method of performing a global optimisation.

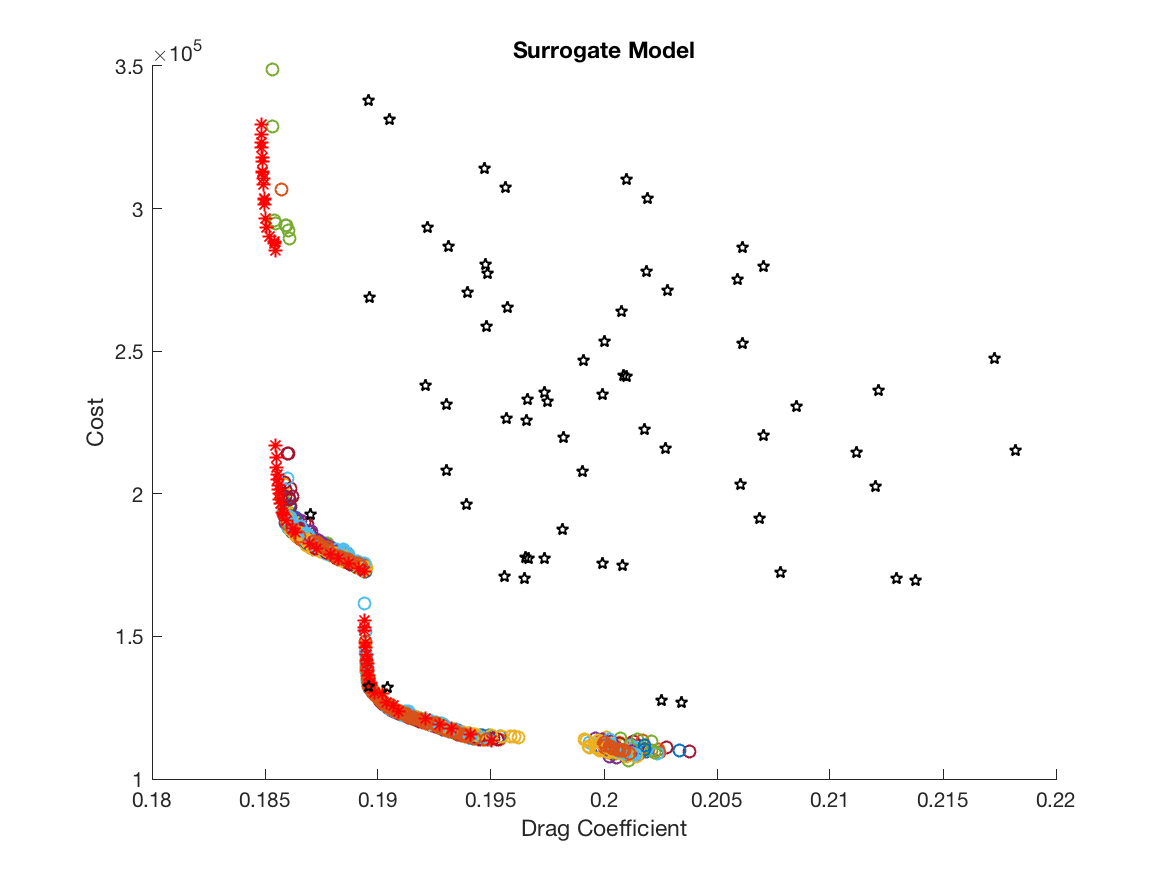


Figure 4 - Kriging with Initial Population Determined by Full Factorial

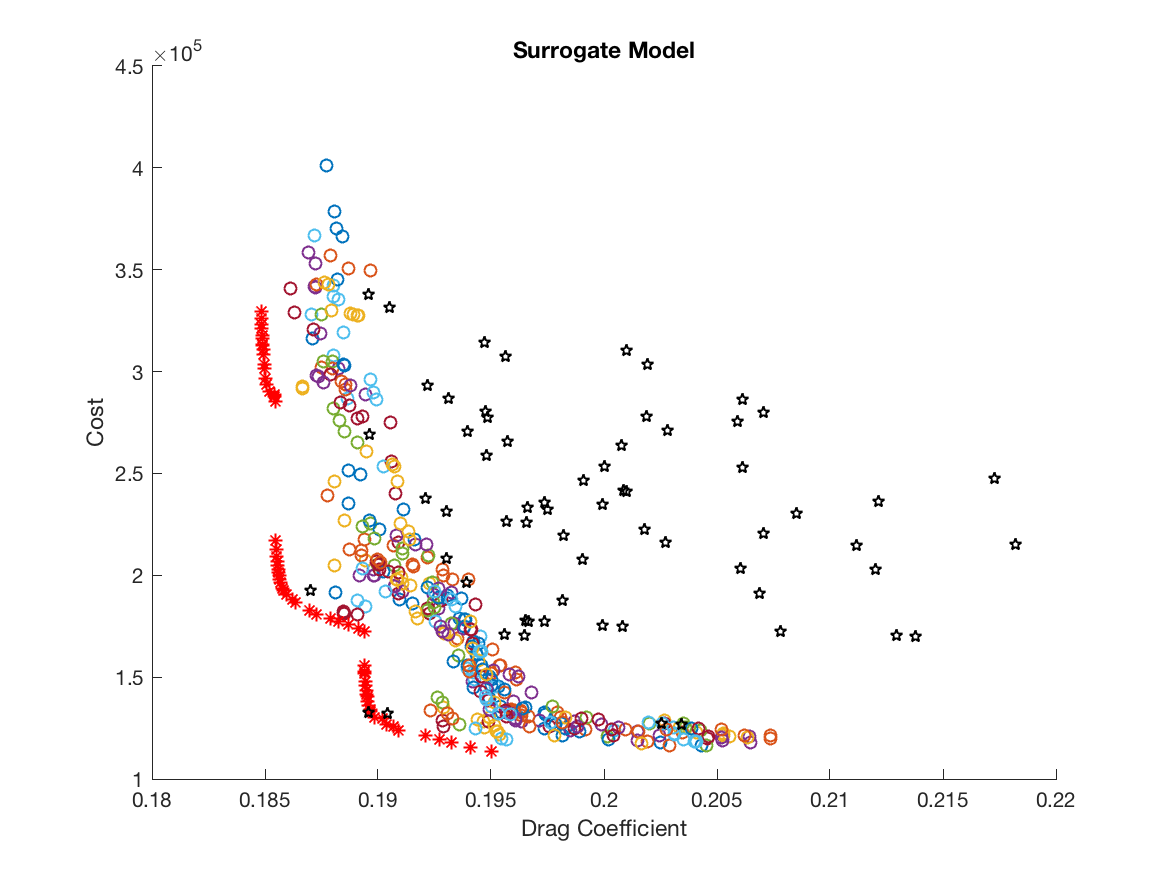


Figure 3 - Kriging with Random Initial Population

## References

1. Mathworks. *gamultiobj.* [*https://uk.mathworks.com/help/gads/gamultiobj.html?s\_tid=doc\_ta*](https://uk.mathworks.com/help/gads/gamultiobj.html?s_tid=doc_ta)*. (Last Accessed 27/03/18)*