# Minimising a Noisy Expensive Function Using Active Learning

**Ross Brown**

Supervisor: Dr. O. Orhobor

Department of Chemical Engineering and Biotechnology
University of Cambridge

This dissertation is submitted for the degree of
*Master of Engineering*

Robinson College September 2021

# Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Ross Brown
September 2021

# Abstract

This is where you write your abstract ...

# Table of contents

# List of figures

# Chapter 1

# Introduction

Finding the global minimum of a function within a set of boundaries is a problem of major import. From optimising a synthetic pathway in drug development, to minimising the error in a neural network, minimisation is vitally important to mathematics. Within the numerical field, the goal is usually two fold: reduce the error, $\varepsilon$, to the true value *and* reduce the processing time. With these goals in mind, the majority of algorithms exploit the commonality of the cheapness of the target function. However, this is not always the case. Take as an example an experimentation of sand grain size, $d$, on the strength of concrete, $\tau$. An underlying function of the form $\tau = f(d)$ exists, but each call to this function takes at least a day, and is labour and material expensive. The target of this paper is to explore how to minimise such a fucntion with the fewest function calls.

## 1.1 Problem Definition

$$y = f(\boldsymbol{x}) \tag{1.1}$$

Given (1.1) where $\boldsymbol{x}$ is a vector with $x_i \in [\alpha_i, \beta_i]$ and y is scalar, find the solution to argmin$[f(\boldsymbol{x})]$. The algorithm will be able to invoke $g(\boldsymbol{x})$ as shown in (1.2) with $\varepsilon$ representing an unknown random error.

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + \varepsilon \tag{1.2}$$

## 1.2 Principles of Active Learning

Active learning involves the intellignent sampling with the intent of reducing the total number of labeling required. Uses of this can be seen in image recognition, where 1000s of images may exist but only a handful of these images have been fully labelled. Each additional image requires the employment of a human to interpret the image. In order to reduce the amount of labelling, [1] suggests there are three methods for reducing the number of required labels.

- Highest Uncertainty
- Competing Hypothesis
- Predicted Model Change

Here, the highest uncetainty value implies knowledge to the probability of the output. Competing hypthesis theories involves several different models, where samples are chosen to test these hypotheses against each other. Finally, the predicted model change takes samples which are expected to have the largest impact on the modelling.

# Chapter 2

# Simple 1-Dimentional Problem

## 2.1 Outlining of Basic Principles

By constraining $\boldsymbol{x}$ to one dimention allows for the problem to be simplified. Suppose $f(x) = \sin(x) + 0.05x^2$ with $x \in [-10, 10]$ as shown in Figure 2.1. In this range, there are multiple minima with only one global minima. The task here is to successfully locate the minima situated at -1.428 (found through ananlytical differentiation and solving $10\cos(x) + x = 0$). Three methods will be used here: fminbound [2], greatest uncertainty active learning, and problem specific active learning (explained in Section 2.1.1). $\varepsilon$ will be chosen to be independent of $x$ and $y$ and fit a normal distribution such that $\varepsilon \sim N(0, 0.2^2)$.

### 2.1.1 Algorithms

#### fminbound

fminbound is a function included in the scipy optimisation library [2]. It uses Brent's method allowing it to be quick in situations where labeling is quick and error is low.

#### Greatest Uncertainty Active Learning

Without a basis for implemementing a certainty model on curve fitting, regions with maximum scarsity of samples will be chosen. In practice, this means for $n$ samples within the range $[\alpha, \beta]$, the sample $s_i$, $i \in \mathbb{Z}$, $i \in [0, n-1]$ will fall chosen will fall according to (2.1). A smoothing spline fit will be used for interpolation identically to problem specific active learning in Section 2.1.1.

$$s_i = \alpha + i\frac{\beta - \alpha}{n} \tag{2.1}$$

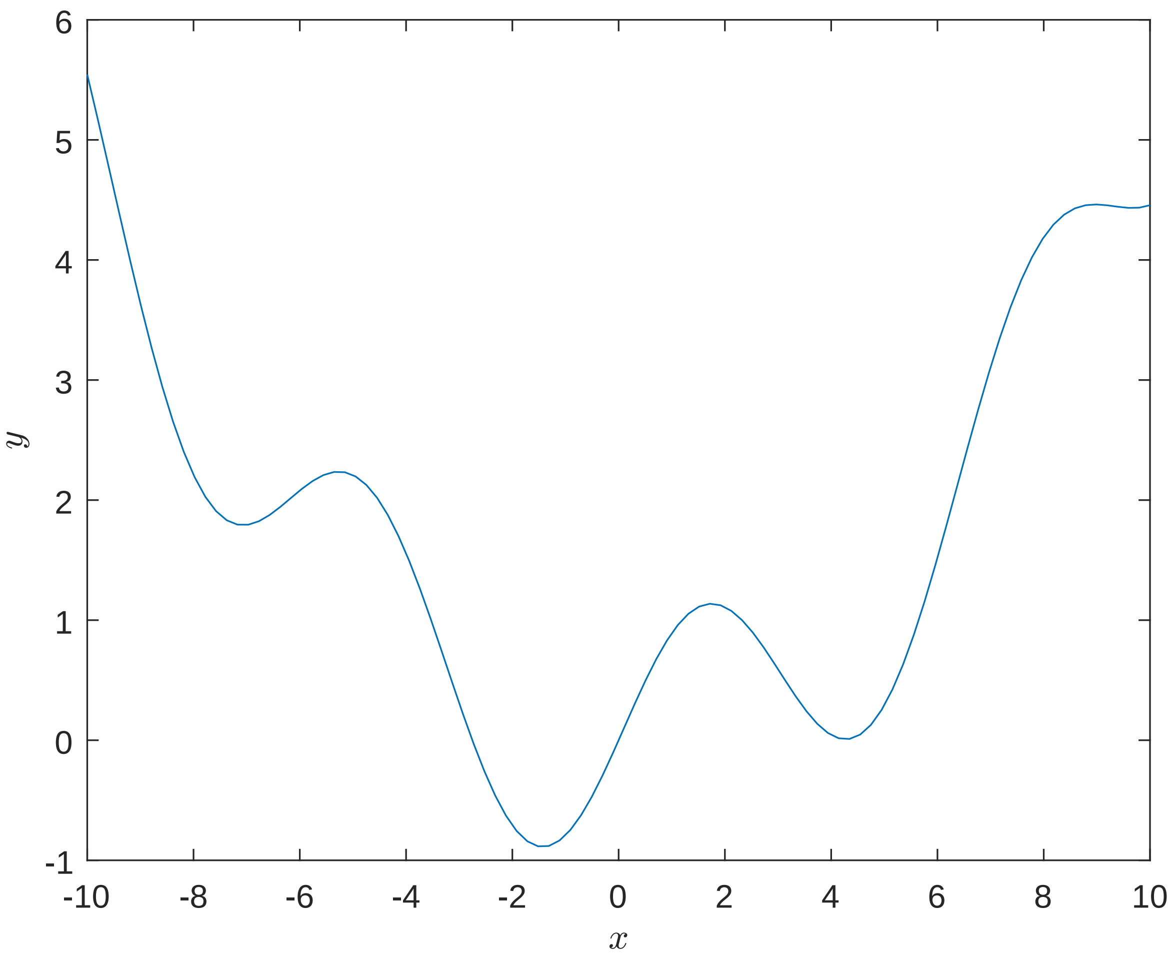

Fig. 2.1 $y = \sin(x) + 0.005x^2$ with $x \in [-10, 10]$

**Problem Specific Active Learning**

This methodolgy has two underlying core principles: sparse areas reveal the most information and minimal areas reveal information to the location of the minima. Combining these allows for better decision making with regards to the next sample to choose. A smoothing spline is used to interpolate providing information for $e(x)$ in (2.2). Functions $h(x)$ and $p(x)$ (see (2.2) and (2.3)) allow for the minimal areas and most sparse areas to be found respectively. The next sample is chosen at $\text{argmax}_x\,[h(x)p(x)]$.

$$h(x) = \frac{-e(x) + \max[e(x)])}{\max[e(x)] - \min[e(x)]} + 0.01 \tag{2.2}$$

$$p(s_i \leq x \leq s_{i+1}) = \min[x - s_i, s_{i+1} - x] \tag{2.3}$$

The smoothness attached to $e(x)$ is defined differently on each step. Here, all steps have a smoothness factor of $0.01n$ until the final estimation is made. This simply becomes $\frac{n}{30}$. The reasoning behind these steps comes down to tolerance. At low $n$, there simply is not enough data for $\varepsilon$ to be noteworthy. Indeed, more couragous guesses are wanted while far from the maximum number of experimentations. There are two ways to achieve this: weight $p(x)$ higher or allow higher curvature on $e(x)$. As more data appears, this need reduces and the smoothness can be increased. For the final iteration, a smoother curve is wanted so scarse data and uncertainty does not negatively affect the result.

### 2.1.2 Comparison on $\sin(x) + 0.005x^2$

Each method discussed in Section 2.1.1 was executed 50 times for each sample size between 2 and 25. Figure 2.2 shows the mean error, standard deviation, and root mean square of each method. When greater than nine samples were used, fminbound [2] becomes the worst choice with each metric presented. For sample sizes beneath this, the problem specific approach appears worse. This is due to the discovery of the local minima at 4.27. If this were indeed the global maxima, fminsearch would likely have failed to find the global minima in all 50 cases. With respect to the two active learning methods, the problem specific method did slightly better, although this could be attributed to chance based on the curve chosen.

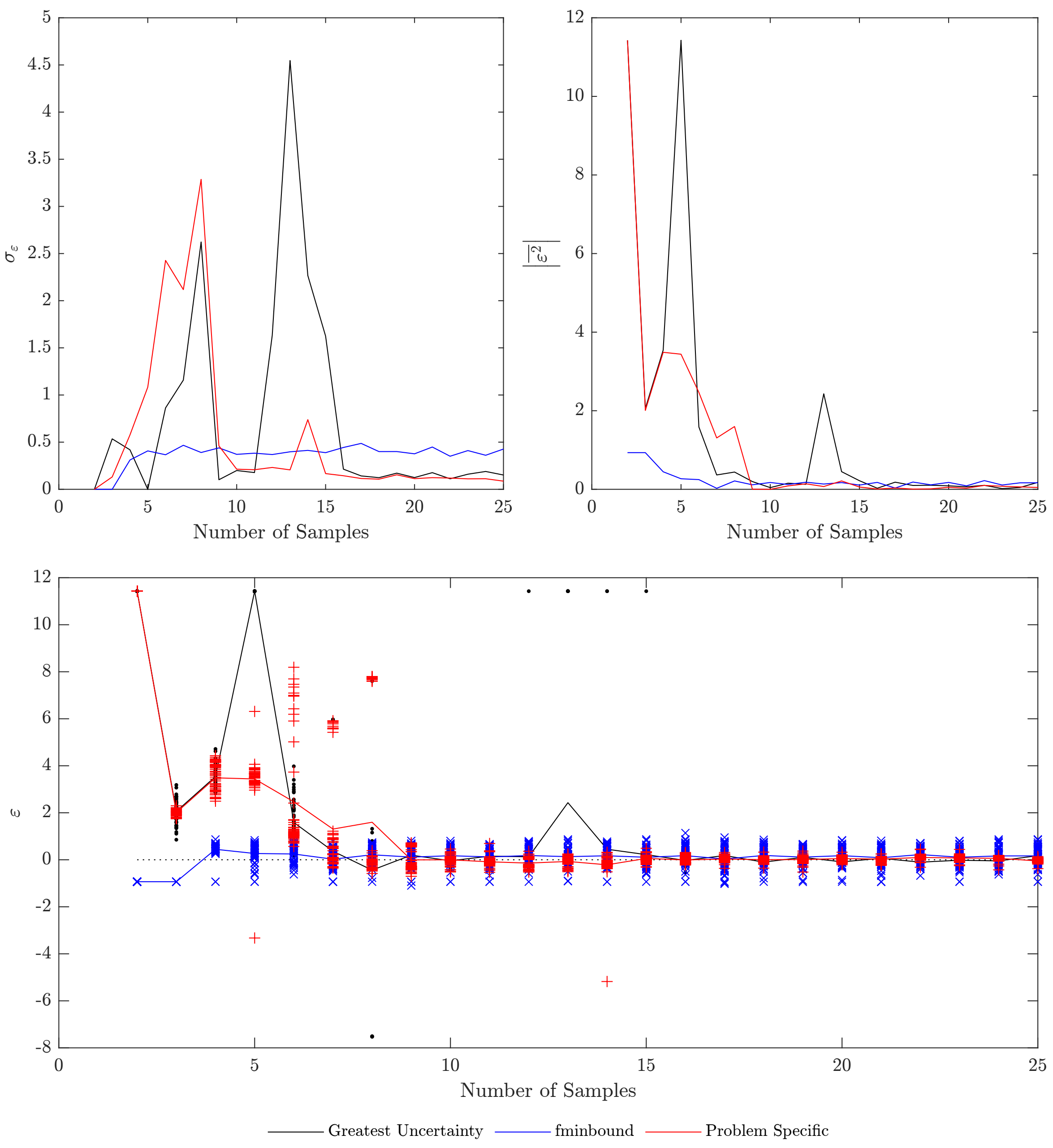

Fig. 2.2 Comparison of the three methods discussed. The top left shows the standard deviation of each method, the top right shows the absolute mean error, and the bottom shows the true error of each sample with a mean of the error to guide the eye.

## 2.2 Deeper Analysis

### 2.2.1 Random Function

To fully test these methods, random functions should be used. For this, serveral basic assumptions are made:

- The function is continuous.
- $x \in [-10, 10]$.
- $f(-10) = 0$.
- $f'(x) \sim N(0, 0.5^2)$.

A simple script for this is in Listing 2.1, which has been designed to allow for easy production of a curve that is differentiable to the $n^{th}$ degree.

Listing 2.1 Funcion to generate a random continuous function.

```
def randFunc():
    lims = [-10, 10]
    derivatives = 1
    x = np.linspace(*lims, 5000)
    f = np.zeros([derivatives + 1, len(x)])
    f[-1, :] = np.random.normal(0, 0.5, np.shape(x))

    for i in range(1, derivatives):
        f[i, 0] = np.random.uniform(-1, 1)

    for i in range(1, len(x)):
        for j in range(derivatives):
            f[j, i] = f[j, i - 1] + f[j + 1, i - 1] * (x[1] - x[0])

return interp1d(x, f[0, :])
```

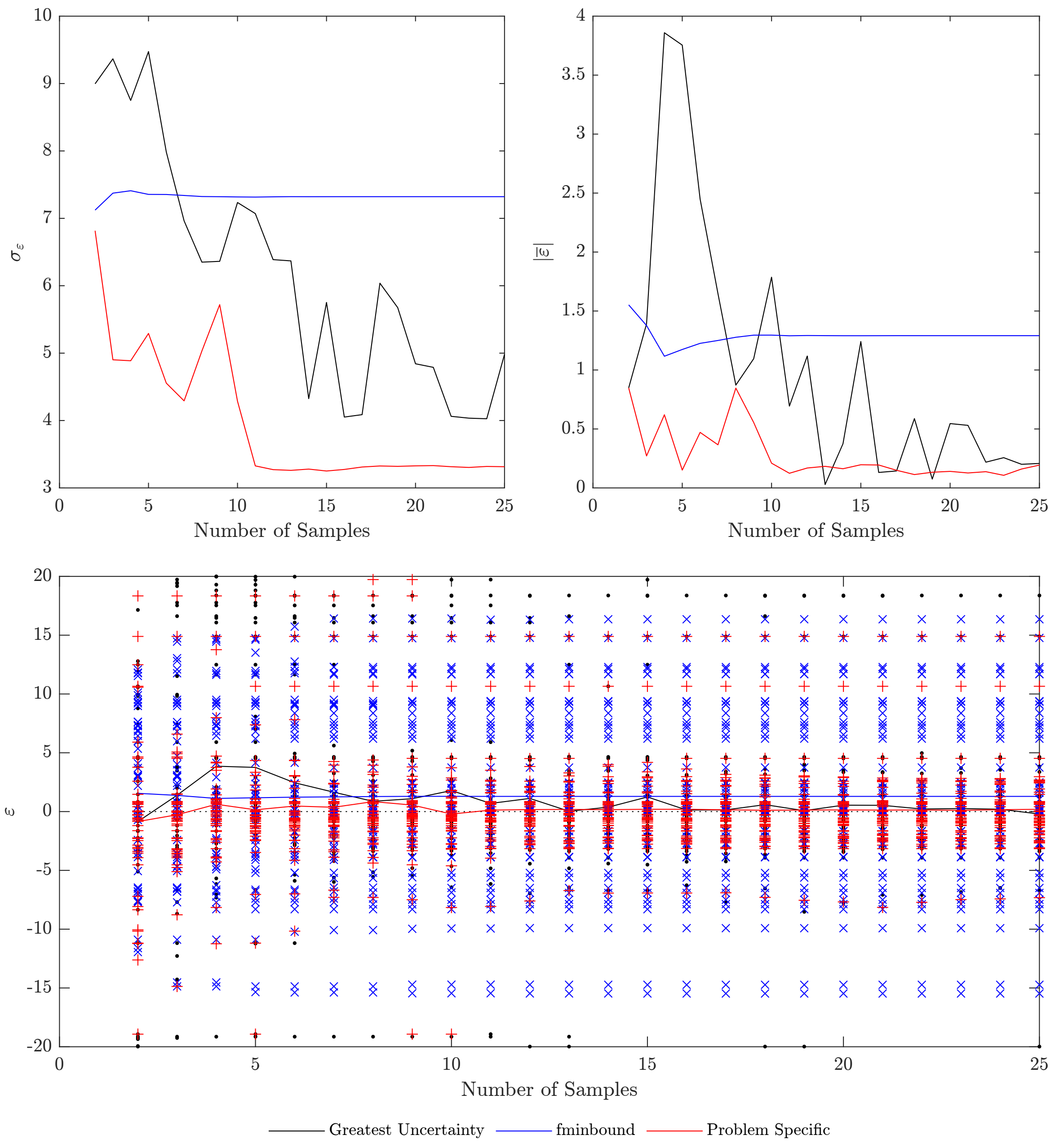

Fig. 2.3 Comparison of the three methods discussed against random functions. The top left shows the standard deviation of each method, the top right shows the absolute mean error, and the bottom shows the true error of each sample with a mean of the error to guide the eye.

# Chapter 3

# Final Discussion

## 3.1 Further Work

Allowing for $n$-dimentional minimisation follows routine procedures. Indeed, simply using splprep from the scipy.optimize package allows for this to be modified to accept n-dimentions. All other procedures remain the same. This was not evaluated within this report as $n$-dimentional functions to evaluate against is difficult to naturally produce. Further work was taken to attempt to converge upon suitable smoothing factors as a function of sample size, although this proved to be unreliable.

Investigation into different non-parametric curve fitting may be made. It is expected that a combination of locally estimated scatterplot smoothing (LOESS) and smoothing splines may be used: LOESS for heavily sampled regions and smoothing splines for scarser areas. Alternatively, regression bins may be used under the assumption that the minima can be approximated to quadratic. Upon estimation of the width of the minima, a suitable bin size may be determined.

Care is needed when choosing values minima with low data representation. This is the cause for many of the extreme deviations seen within the problem specific active learning methodology. To combat this, a coupling of estimated minima and certainty could be tailored.

## 3.2 Conclusion

# References

[1] Settles, B. (2009). Active learning literature survey. pages 1–47.

[2] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.