Technical Milestone Report: Structural optimisation using statistical inference techniques

Zoe Tan

January 20, 2021

Abstract

Often, engineers wish to find optimal designs of structural products. In this project, Bayesian optimisation using Gaussian Process models is being applied to structural optimisation problems. This report details the initial testing of the technique and potential improvements using synthetic examples. Some experimentation on structural optimisation has been completed and will be the focus of the remaining project time.

1 Introduction

This project sets out to apply a statistical inference technique, Bayesian Optimisation, on structural design problems. Through testing and research so far, a key difficulty found is that obtaining a data point is expensive, and Bayesian Optimisation applied to structural optimisation needs many data points. Multi-fidelity Gaussian Process models improve the efficiency of the process, reducing the number of high-cost data points required, compensating by including low-cost data into the model. Sparse Gaussian Process models reduce the time needed to compute the predictive distribution when lots of data is supplied. The project is on track and the remaining time will be used to apply Bayesian optimisation on a more complex structural optimisation problem, including implementing the improvements discussed.

2 Motivation and Aims

Structural optimisation seeks to find designs for structures that fulfill some given specifications, aiming to find a optimum solution. For example, a beam design may need to sustain a load while remaining under a deflection limit. For any design, engineers also aim to minimise the structure's weight and therefore material usage and cost. This type of design problem can be seen as a constrained multi-variate optimisation of a 'black-box' function.

Conventionally, structural optimisation has been done using derivative based approaches. However, the problems are often constrained or includes discrete variables and the structure complex, leading to non-linear or intractable objective functions. The core goal of this project is to test a data-driven approach instead. The data obtained in a structural optimisation problems are from Finite Element Models (FEMs) or (scale) prototypes. Imperfections in

these models lead to noisy data. An estimate of the objective function can be obtained from this noisy data and an estimated minimum found by applying Bayesian optimisation using Gaussian Processes.

The project aims to highlight the possible use of Bayesian optimisation in industrial-scale structural optimisation, the difficulties and techniques to improve the optimisation process. I will do this by adapting Bayesian optimisation tools to create a structural optimisation toolkit and apply it on relatively simple structural problems. I will also utilise various techniques to improve on the optimisation process to require less time and data.

3 Plan

- Implement basic Bayesian Optimisation using Gaussian Process models ✓
  - Use various functions with known minima to confirm core functionality ✓
  - Compare performance of various hyperparameters settings ✓
  - Compare performance difference with different test functions - low or high dimensional, convexity etc. ✓
- Explore ways to reduce optimisation time ✓
  - Implement sparse Gaussian Process ✓
  - Implement multi-fidelity Gaussian Process ✓
  - Evaluate techniques and effect on time and accuracy ✓
- Implement FEM on trusses and simple 3D models ✓
- Test Bayesian Optimisation on basic structures
  - Test basic Bayesian Optimisation ✓
  - Include improvements made, test and compare
- Apply Bayesian Optimisation on test structure
  - Define structural problem - design, variables and constraints
  - Create FEM of test structure
  - Optimise structure

The project is broken down into an initial testing phase of each code component using known functions and simple structures. After this, a more complex structure will be selected for modelling and optimisation applied on it. At the end, I will analyse the Bayesian Optimisation process applied to structural optimisation for its efficiency and accuracy.

4 Progress

4.1 Mathematics of Bayesian optimisation

Bayesian optimisation uses Gaussian processes (GPs). Informally, GPs can be thought of as probability distributions for functions. They are the generalisation of multivariate Gaussians to infinitely many variables. A GP is fully defined by its mean function $m(x)$ and covariance function $k(x, x')$ - $f \sim \mathcal{GP}(m, k)$.

For training data, we assume $\mathbf{y} = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I})$. The covariance function determines form of the functions. By maximising marginal likelihood with respect to the kernel hyperparameters, the GP model can be optimised. The GP log marginal likelihood has a closed form which applies Occam's razor, penalising complexity while rewarding data fit.

The GP is "trained" on data to give a posterior distribution which can then be used to predict a minimum (see Rasmussen & Williams (2006) for detailed mathematics):

$$p(f|\mathbf{x}, \mathbf{y}, \mathcal{M}_i) \sim \mathcal{GP}(m_{\text{post}}, k_{\text{post}})$$

$$m_{\text{post}}(x) = k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}$$

$$k_{\text{post}}(x, x') = k(x, x') - k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}]^{-1} k(\mathbf{x}, x')$$

So far, these equations are simply performing regression on the training data. Bayesian optimisation sequentially selects new test points to improve the model and thus find an estimate of the min/max of the underlying function. It does so with an acquisition function $u(x)$. The acquisition function balances exploration, sampling from unexplored high uncertainty areas, and exploitation, sampling around the current optimum to find the local optimum. Some common acquisition functions are Maximum Probability of Improvement (MPI), Expected Improvement (EI) and Lower Confidence Bound (LCB) (see Brochu et al. (2010)).

4.2 Implementation and Experimenting with Bayesian Optimisation

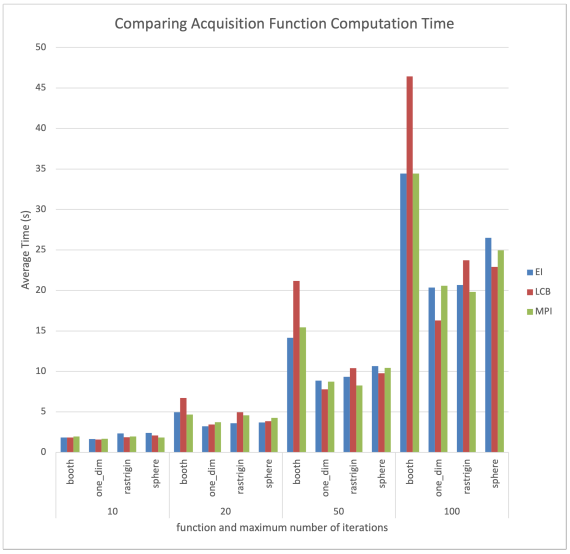
As the optimisation process is mostly based on closed form mathematics, the implementation of code would be fairly trivial with scientific Python. Spending time optimising matrix calculations is beyond the scope of this project, so I decided to make use of Python libraries. This also allows for more time spent on experimentation. The `GPyOpt` library based on `GPy` was chosen as it is well documented and has good functionality.

I experimented with 4 functions with known minima to explore the characteristics of the Bayesian optimisation process. These functions were a 1-dimensional simple test function (multimodal, non-convex), Booth (2-dimensional, unimodal, convex), Sphere (n-dimensional, unimodal, convex) and Rastrigin (n-dimensional, multimodal, non-convex). I ran the optimisation on these functions with different acquisition functions, number of iterations and noise variance.

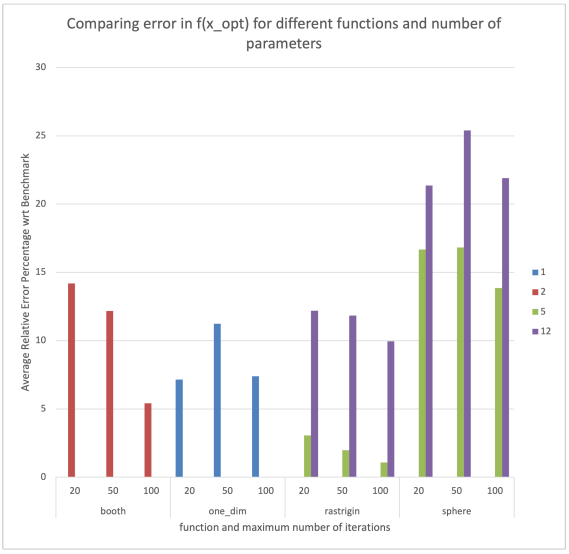
I found that between the acquisition functions, there was not one that was clearly more efficient for all test functions, as seen from Figure 1a. From literature, EI seems to be the

most commonly used acquisition function and does not contain a hyperparameter which would require tuning. From Figure 1c, we can also see that as the number of iterations increases, the time per iteration taken for the optimisation to run also increases. This is because with a larger dataset, the covariance matrix $\mathbf{K}$ also gets larger. Since the calculation of the GP posterior requires finding the inverse of $[\mathbf{K} + \sigma_n^2 \mathbf{I}]$, and finding the inverse of a matrix is a $\mathcal{O}(N^3)$ calculation, this greatly increases the time taken per iteration.

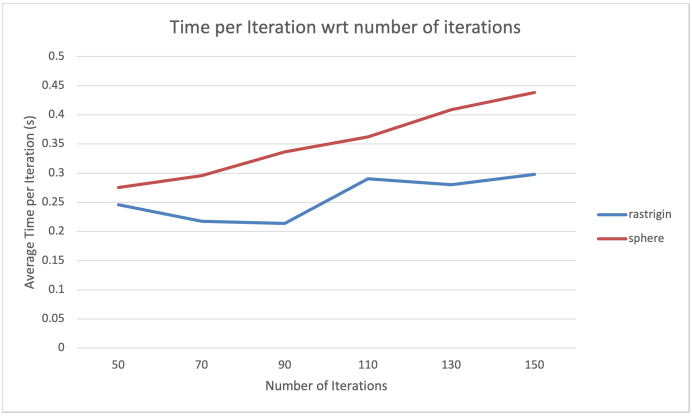
As the different functions had different levels of difficulty with regard to optimisation, I used a relative error measure to compare the optimisations, found with respect to a benchmark error. From Figure 1b we can see that, as expected, the error in the predicted f_{opt} tends to decrease with more iterations. Notably, the actual error is still quite large for the higher dimensional functions, especially Rastrigin. As we expect a structural optimisation problem to be similarly high dimensional and likely non-convex, many iterations will need to be performed to obtain a satisfactory result using simple Bayesian optimisation.



(a) Comparing acquisition functions



(b) Relative error change over number of iterations



(c) Time per iteration change

Figure 1: Base Bayesian optimisation test results

function	no. data points	f_{opt} error
1D	10	0.13
1D, 2 fidelities	10, 20	0.00039
Rastrigin, 12D	50	128.93
Rastrigin, 12D, 2 fidelities	50, 500	97.46

Table 1: Regression results for some function tests, using $\sigma_n = 0.1$

4.3 Improving on optimisation time

I first attempted to tackle the issue of time per iteration. I tested optimisation using sparse GP models, which aim to combat the costly matrix inverse calculation with large datasets. There are various sparse methods, of which we will focus on Variational Free Energy (VFE) as outlined by Titsias (2009). The VFE method directly approximates the posterior GP means and covariance functions.

This model is a `GPyOpt` library built-in. With 1000 data points, regression on a normal GP model took 32 seconds to run, while it only took 20 seconds with a sparse GP model. The complexity for sparse GP is reduced to $\mathcal{O}(NM^2)$ where M is number of psuedo-data and $M << N$.

After testing however, I realised that this may not have great impact for the project. The bottleneck for the project will likely be running FEM models rather than in the regression. Therefore, efforts should be concentrated in minimising the number of data points necessary for good optimisation.

I then moved on to testing multi-fidelity GPs. These models take data points from a lower fidelity approximation to the target function and incorporates this additional information into the model. The model can be expressed as:

$$u_1 \sim \mathcal{GP}(m_1, k_1(x, x')), u_2 \sim \mathcal{GP}(m_2, k_2(x, x'))$$

$$f_L(x) = u_1(x), f_H(x) = \rho u_1(x) + u_2(x)$$

A predictive posterior for f_H is given (closed form) using data points from both f_L and f_H.

Using modified example code from `GPTutorial`, I tested regression on multi-fidelity GPs on the functions outlined previously, visualising the posteriors when possible. Figure 2 shows the posterior distribution of the one-dimensional function. Table 1 shows how the accuracy improves with the use of multifidelity GPs.

To utilise this in Bayesian optimisation, Zhang et al. (2019) outlines an algorithm which iterates data sampling between the two fidelities. This process can be formalised with multi-fidelity acquisition functions, as in Kandasamy et al. (2019) and others. As comparing these multi-fidelity options is outside the scope of the project, I will attempt the simplest option from Zhang et al. (2019) for the final experimentation phase first. If time allows, I hope to explore the multi-fidelity acquisition functions.

4.4 Structural optimisation

As FEMs are a basic concept within the structures group, and there are many commercial products that can efficiently model structures, I will be using the `PyNite` library to run FEMs.

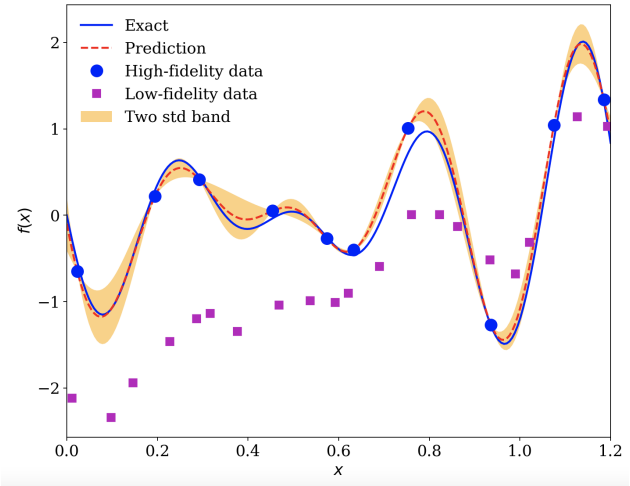


Figure 2: 1-dimensional multi-fidelity GP regression

I performed some Bayesian optimisation tests (without improvements) on a basic truss to confirm functionality and compatibility between the libraries. For the objective function, I performed a weighted sum on weight and maximum deflection to give one output. By varying the weights, the Pareto frontier can be found through the optimisation process.

For the final structural optimisation test, I will base the problem on the spaceframe structure from Lim et al. (2020). The paper will provide a validation for the Pareto frontier my optimisations find and allow comparison of the data cost of different optimisation approaches. I will estimate the behaviour under load using a truss structure to obtain low-fidelity data, and model using 3D elements to obtain high-fidelity data.

References

Brochu, E., Cora, V. M. & de Freitas, N. (2010), 'A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning'.

Kandasamy, K., Dasarathy, G., Oliva, J. B., Schneider, J. & Poczos, B. (2019), 'Multi-fidelity Gaussian Process Bandit Optimisation'.

Lim, J., You, C. & Dayyani, I. (2020), 'Multi-objective topology optimization and structural analysis of periodic spaceframe structures', *Materials & Design* **190**, 108552.

Rasmussen, C. E. & Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, MIT Press, chapter 2.

Titsias, M. (2009), Variational Learning of Inducing Variables in Sparse Gaussian Processes, *in* D. van Dyk & M. Welling, eds, 'Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics', Vol. 5 of *Proceedings of Machine Learning Research*, PMLR, pp. 567–574.

Zhang, S., Lyu, W., Yang, F., Yan, C., Zhou, D., Zeng, X. & Hu, X. (2019), 'An Efficient Multi-fidelity Bayesian Optimization Approach for Analog Circuit Synthesis', *Proceedings of the 56th Annual Design Automation Conference 2019* .