A COMPARISON BETWEEN GAUSSIAN PROCESS EMULATION AND GENETIC ALGORITHMS FOR OPTIMISING ENERGY USE OF BUILDINGS

Michael Wood, Matthew Eames, and Peter Challenor
College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK

ABSTRACT

Computing speed has increased greatly over recent years. Building designers can now simulate complex building models in a short time. However, even with short simulation times, building optimisation routines can still take too long for some applications.

In this paper, we compare how well genetic algorithms (GAs) and Gaussian process emulation with sequential optimisation (GPESO) optimise a building to minimise the energy use. The GA approach performs a GA routine on an *EnergyPlus* model and the GPESO technique creates a Gaussian Process emulator (GPE) also based on the *EnergyPlus* model. The GPESO uses an *expected improvement* algorithm to sequentially improve the GPE. The results show that the GPESO technique outperforms the GA in terms of minimising the number of simulations required and the solution obtained.

INTRODUCTION

Energy use in buildings accounts for around 40% of all energy use in the European Union (Uihlein and Eder 2010). EU countries must therefore drastically reduce the CO_2 emissions from buildings to meet their carbon budgets. To reduce energy consumption in buildings, it is essential that we design thermally efficient new buildings and effectively retrofit older ones. However, building design is a highly complex problem, which cannot be solved without computational aids.

Over recent decades, the performance of desktop computers has increased greatly. Building designers now routinely use highly sophisticated building simulation software to design energy efficient buildings. However, although more 'accurate' simulation tools have been created, most tools are not capable of evaluating potential design options automatically. Parametric studies are typically completed manually, so it is therefore unlikely that designers will routinely find the best.

Finding the optimum design for a building is computationally demanding. To search all the possible design options, the number of simulations (n) required is,

$$n = l^d,$$

where l is the number of search levels and d is the number of dimensions.

This means that for nine parameter inputs, each with 10 potential values, 9 billion simulations would be required. For a fast simulator taking only 0.001 seconds to run, the search would take three months. If we double the potential values to 20, the analysis time would increase to over 385 million years. The search space also exponentially increases according to the number of dimensions. This problem is commonly know as Bellman's *curse of dimensionality* (Bellman 1957).

Methods to overcome the curse of dimensionality have been widely researched across many engineering disciplines (Jones, Schonlau, and Welch 1998). Commonly researched techniques, in building simulation at least, include evolutionary algorithms, multi-start, and simulated annealing (Bull 2011; Jones D.R. 2001). However, although these methods have their advantages, they are often best suited to objective functions that are relatively quick to evaluate (Bull 2011).

In this research, we compare two optimisation techniques. The first technique uses a Genetic Algorithm (GA), which is used widely in building research. The second technique, Gaussian Process emulation with sequential optimisation (GPESO), although common in other area of engineering, has rarely been applied to buildings.

Due to the large number of different building types, climates and variables of interest, it is difficult to definitively compare GAs and GPESOs, since the 'shape' of the output can vary greatly. Instead, we apply both tools to a sufficiently complex building problem and qualitatively compare the results. We hypothesise that the GPESO method is a more efficient global optimisation method, because it provides full interpolation between the training points and can cover the full range of inputs. It can also provide an estimate of its own uncertainty, which we use to identify and correct areas of poor emulation performance.

METHOD: THE BUILDING MODEL

The building model used in the assessment is a single story medium office with a corridor to the north façade and windows to the south, east and west, each with a brise soleil (Figure 1).

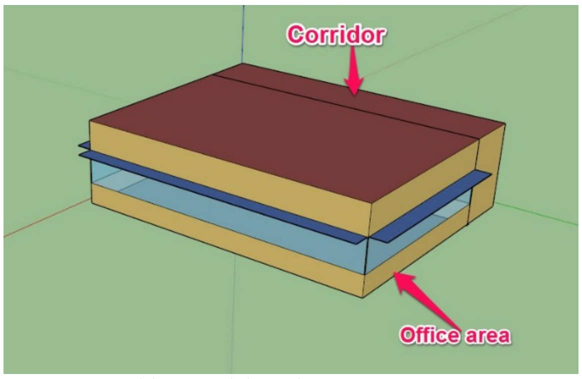

Figure 1: Building model used

The construction of the building is based on the *Medium Office* of ASHRAE 189.1-2009 (ASHRAE and US Green Building Council 2014):

Table 1: Constructions used in the building model

Element	**Construction**
External walls	25 mm Stucco 200 mm Concrete (heavyweight) Wall insulation 40 mm 12.5 mm Gypsum
Windows	ASHRAE 189.1-2009 *ExtWindow* ClimateZone 4-5
Ceiling	M11 100mm lightweight concrete F05 Ceiling air space resistance F16 Acoustic tile
Roof	Roof Membrane Roof Insulation [21] Metal Decking
Internal wall	19mm gypsum board air gap (m^2K/W) 19 mm gypsum board

Table 2 shows the parameters varied in the assessment to determine the lowest annual energy use.

Table 2: Model input values

VARIABLE	**LOWER RANGE**	**UPPER RANGE**
Building floor area	200 m^2	1000 m^2
Aspect ratio	1	10
S window to wall ratio	0.02	0.9
E window to wall ratio	0.02	0.9
W window to wall ratio	0.02	0.9
S shade projection factor	0.05	1
E shade projection factor	0.05	1
W shade projection factor	0.05	1
Orientation	-90°	90°

The annual energy use of each building configuration was predicted using *EnergyPlus* (US Department of Energy 2013) using weather data from the UKCIP09 test reference year for Cardiff, UK.

METHOD: OPTIMISATION PROCEDURES

Genetic Algorithms

Genetic algorithms (GAs) are a class of evolutionary algorithms which were made popular in the 1970s (Holland 1975) and have become popular for optimising complex models (Scrucca 2013; Goldberg 1989; Sivanandam and Deepa 2007). In a GA, the input parameters are encoded onto a 'chromosome'. In this case, each chromosome represents a particular building configuration.

The first step of the GA is the creation of an initial *population* of input chromosomes. These chromosomes are then evaluated, using *EnergyPlus*, according to a *fitness function*, which defines the 'fittest' or 'best' output. In a GA, the fitness function should be *maximised,* but since we want to *minimise* the annual energy use, we express the fitness function as

$$fitness = -f(x).$$

After the initial population has been evaluated, a new population is generated using *selection, crossover* and *mutation*, which are defined as follows:

1. *Selection*: From the current population, the *fittest* are then chosen for crossover and mutation
2. *Crossover*: Creates new offspring from two parent chromosomes by combining part of the 'genetic' information from each
3. *Mutation*: Mutation randomly alters the values of genes in a parent chromosome

After each generation, the best 5% of buildings survive. This is referred to as *elitism* and ensures that the best outputs are not lost over time.

The selection, crossover and mutation process is repeated for a set number of iterations. The best output from the final population is the 'optimum' solution. The GA is implemented using the *R* package *GA* (Scrucca 2013).

Gaussian Process emulation

Gaussian Process emulation is a spatial interpolation method (Roustant, Ginsbourger, and Deville 2012). which has its origins in *geostatistics* (Krige 1951), but is now a widely used technique for exploring complex high-dimensional models in climate and engineering (Drignei 2009; Holdaway 1996; Tokmakian and Challenor 2013). GP emulation is different from other emulation methods because it fully interpolates between each of the known outputs and makes estimates of its own uncertainty. This gives GP emulation an advantage over fitted quadratic surfaces for example, which do not always capture the shape of the objective function well and provide no estimate of their own accuracy (Jones D.R. 2001).

In a GP analysis, sample data is used to estimate the output of a model for untested input combinations. We notate the output of the simulator (i.e. *EnergyPlus*) as $f(x)$, and the output of the GP emulator as $\hat{f}(x)$, which is defined as multivariate Normal distribution:

$$\hat{f}(x) = \mathcal{N}(m(x), v(x,x)),$$

where $m(x)$ is the mean function of the emulator and $v(x,x)$ is the covariance function.

Defining the emulator in this way does not mean that we view the output of the *simulator* as stochastic. The mean and variance are only used to quantify the uncertainty in the output for untested areas of the simulator. Figure 2 illustrates the mean and variance functions for a simple GP emulator based on a one-dimensional function, $y = \frac{\sin(4\pi x^2)}{2x^2} + (x-1)^2$.

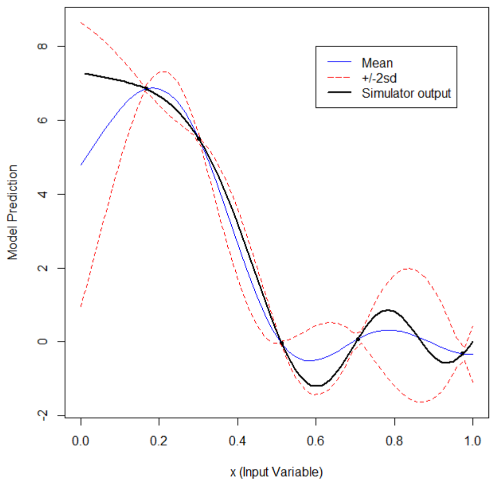

Figure 2: Simple example showing a 1-dimensional emulator.

Figure 2 shows how the emulator makes both predictions for the mean function (in blue) and the 95% confidence intervals (red dotted lines). Figure 3 shows how an additional five training points allows the emulator to rapidly converge on the objective function, $f(x)$.

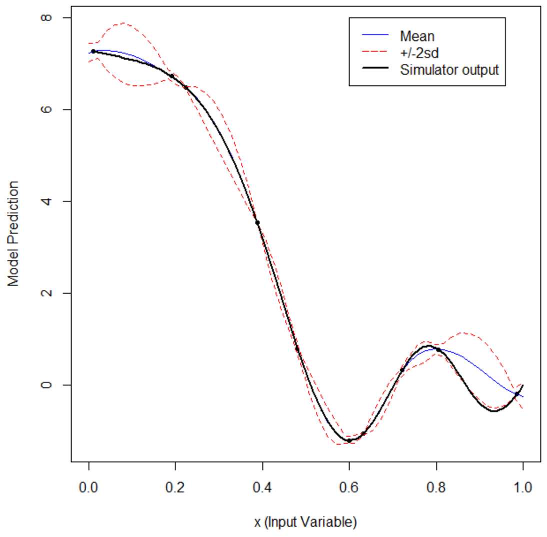

Figure 3: 1D emulator based on 10 training simulations

Gaussian Process Emulation with Sequential Optimisation (GPESO)

One approach to optimisation using GP emulators is to optimise the emulator directly, since each run of the emulator is several orders of magnitude faster than the simulator. However, this approach is not efficient (Jones D.R. 2001) because the emulator can create false local minima, due to limited information about the objective function. This is solved using the expected improvement criterion (EI), which takes advantage of the emulator's uncertainty predictions (Jones, Schonlau, and Welch 1998; Brochu, Cora, and de Freitas 2010). The GPESO method is a 6-step process that:

1. Creates a set of design input points, $\boldsymbol{D}$ using a maxi-min Latin Hypercube design;
2. Uses the *EnergyPlus* simulator to determine $f(\boldsymbol{D})$ for each of the inputs;
3. Builds a GP emulator using $\boldsymbol{D}$ and $f(\boldsymbol{D})$;
4. Maximises the expected improvement criterion (EI) to determine the best input configuration for the next simulation;
5. Uses $\boldsymbol{D_{opt}}$ from step 4 to re-estimate the GP model (including covariance parameters re-estimation) based on the $\boldsymbol{D}$ and $\boldsymbol{D_{opt}}$;
6. Repeats steps 4-5 up to 7 more times adding in the new $\boldsymbol{D_{opt}}$ point into the GP model each time. It then finds the minima of the GP model as a proxy for determining the minima of the simulator.

The GPESO technique identifies areas of potential poor performance within the emulator and adds additional training simulations in these areas to improve emulation. For a more in-depth description of this type of sequential optimisation, see Roustant et al. (Roustant, Ginsbourger, and Deville 2012). For more background on GP emulation as a whole, see (Santner, Williams, and Notz 2003; Fang, Li, and Sudjianto 2006; Rasmussen and Williams 2006; O'Hagan 2006).

METHOD: TEST PROCEDURES

We performed 39 GPESOs with varying numbers of simulations (see Table 3). The number of simulations includes the the initial training set and the seven additional simulations each following an evaluation of the EI criteria.

Table 3: Simulation set ups for the GP optimisation

NUMBER OF SIMULATIONS (TRAINING + EI EVALUATIONS)	NUMBER OF GP EMULATORS	REPETITIONS
37	4	3
57	4	3
107	3	3
207	4	3
407	5	3

Each GP sample is repeated three times because the selection of the training set is stochastic and may therefore lead to different results on each iteration.

The number of simulations used by the GA depends on the ratio between the size of the population and the number of iterations:

$$No.of\ Simulations = Population \times Iterations$$

A GA with a set number of simulations can be configured a number of ways. For example, a 100-simulation GA can be set up as a population of 20 with 5 iterations, or a population of 10 with 10 iterations. Table 4 shows the GA simulation routines that we used in our assessment. Due to the stochastic nature of GAs, there will be some variation in the output of each individual routine. We therefore performed each routine at least three times.

Table 4: Simulation routines for the Genetic Algorithms

ROUTINE NO.	POPULATION	ITERATIONS	TOTAL SIMS	REPETITIONS
1	2	15	30	3
2	2	25	50	3
3	3	10	30	3
4	5	10	50	3
5	5	20	100	3
6	5	40	200	3
7	5	80	400	3
8	10	3	30	3
9	10	5	50	3
10	10	10	100	3
11	10	20	200	3
12	10	40	400	3
13	15	2	30	3
14	20	5	100	3
15	20	10	200	3
16	20	20	400	3
17	25	2	50	3
18	40	5	200	3
19	40	10	400	3
20	80	5	400	3

RESULTS

Figure 4 shows best results from each process against the number of simulations required to reach it. For the GAs, we group the results by population size. For the GP emulation method, the results are in red.

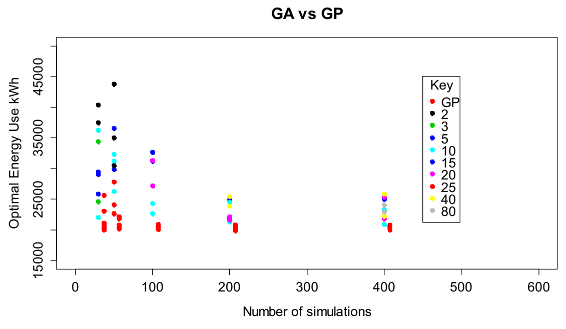

Figure 4: Results of simulations.

Figure 5 to Figure 8 show the results more clearly. We have drawn the areas in the plots by connecting the maximum and minimum values for each simulation size to help show the general trend.

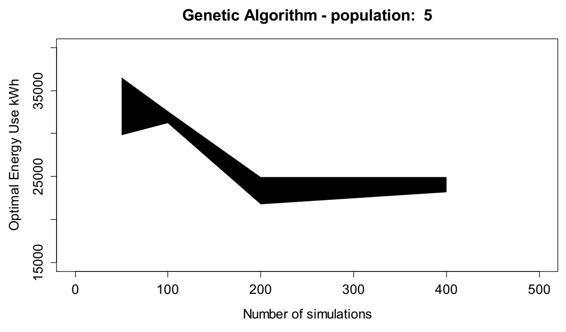

Figure 5: Range of GA outputs for population = 5.

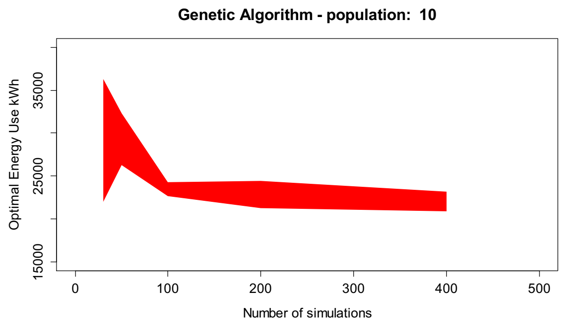

Figure 6: Range of GA outputs for population = 10.

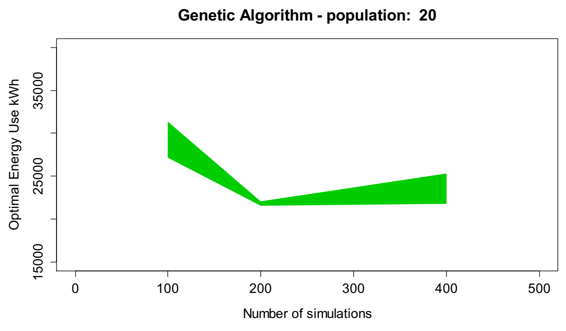

Figure 7: Range of GA outputs for population = 20.

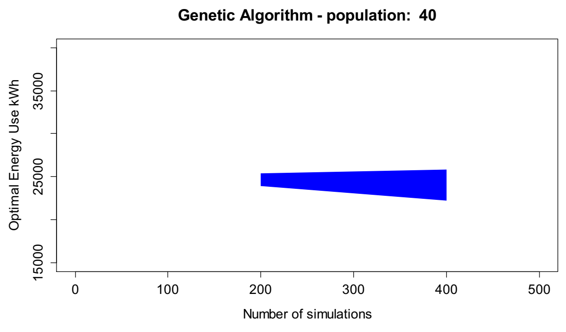

Figure 8: Range of GA outputs for population = 40.

Figure 9 shows the results of the Gaussian Process only and Figure 10 shows the results overlaid with the results of the GA technique.

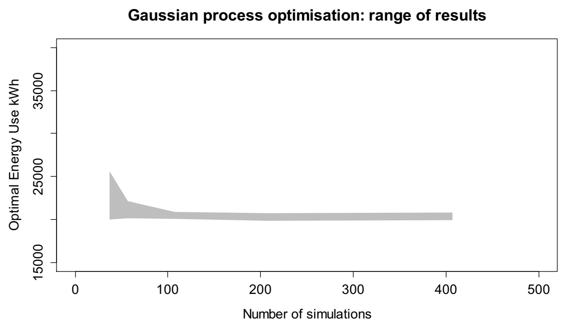

Figure 9: Results of the Gaussian Process analysis only

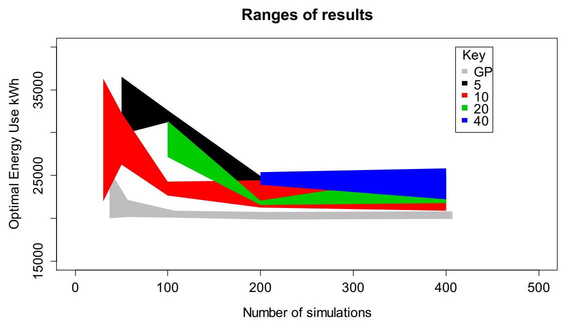

Figure 10: All results overlaid

The GPESO compares favourably to all of the GA population groups and has a much more stable output. Although the GAs perform best with a small population size, they are still outperformed in nearly all cases by the GPESO approach.

Figure 11 and Figure 12 compare the best-performing GA to the GPESO emulations undertaken in parallel. Figure 11 compares the *mean* results from each set (the set is determined by the number of simulations used to produce that result) and Figure 12 compare the *best* results from each set.

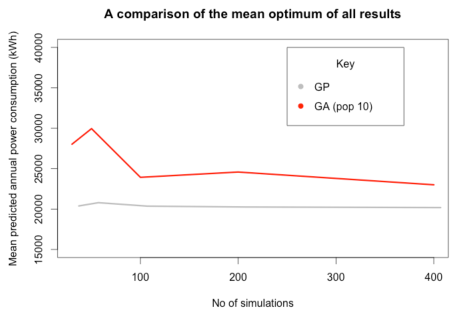

Figure 11: A comparison between the mean of the GA results (population 10) and the GPESO

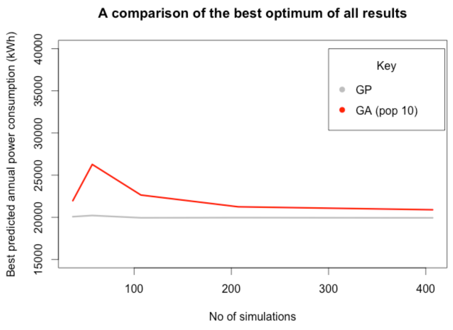

Figure 12: A comparison between the best of the GP results (population 10) and the GPESO

The results show that the GPESO output does not vary much as the number of simulations is increased. The GA does improve with more simulations, but still does not match the performance of the GPESO.

The two best solutions offered by the GPESO and GA (with a population of 10) are within 5% of each other. The solutions have similarities, but there are also significant differences (differences greater than 20% are highlighted in **bold** in Table 5).

Table 5: Best solutions from the GA (population size 10) and the GPESO. (Note that the GA and GPESO results have the same sample size)

	Gaussian process	Genetic algorithm	Unit
Number of simulations	407	400	Integer
Best predicted annual energy use	19934	20894	kWh
Building area	200	206	m^2
Aspect ratio	1.00	1.16	Ratio
Glazed ratio (S)	**0.59**	**0.37**	Ratio
Glazed ratio (E)	**0.02**	**0.42**	Ratio
Glazed ratio (W)	0.15	0.16	Ratio
Projection factor (S)	**0.65**	**0.47**	Ratio
Projection factor (E)	**1.00**	**0.35**	Ratio
Projection factor (W)	**0.05**	**0.31**	Ratio
Orientation	-58	1	Degrees

The major differences between the two solutions are in the glazing, the brise soleil overhang factor and the orientation. However, given that the two buildings have roughly the same floor area and aspect ratio, the *form* of both buildings are very similar (see Figure 13 and Figure 14).

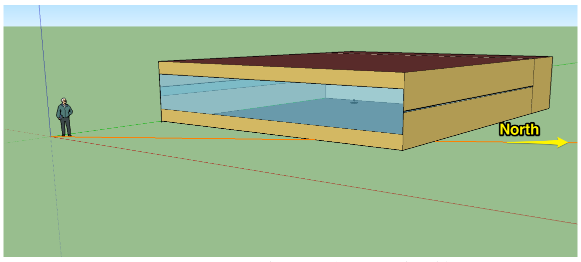

Figure 13: Best GPESO solution (19934 kWh)

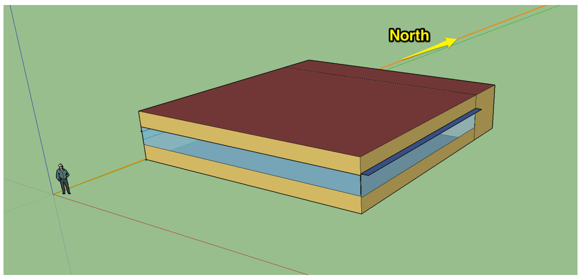

Figure 14: Best GA solution (20894 kWh)

Both buildings have different orientations but have little or no glazing to the Northerly façade.

CONCLUSION

Quantifying the relative performance of each method is beyond the scope of this paper, however, the results suggest GPESO is more efficient than the GA, regardless of the ratio between population size and iterations.

The GPESO output appears to be much more stable, even when using only 57 simulations. In contrast, the GA required a greater number of simulations to produce a more stable output. Even with 400 simulations, the stability of the GA was much less than that obtained by the GPESO technique.

The GAs with smaller populations tended to produce more consistent results. The best solutions from each method have significant differences and similarities. The solutions are, despite differences in their physical characteristics, within 5% of each other. Therefore there are likely to be several local optima within this particular objective function. Although the results suggest that the GPESO technique is better at finding the global optimum, further work is required to confirm and quantify this.

Future work should also consider different building layouts and climates, as well as building types and usages. It is likely that the efficacy of each technique is dependent on the buildings and parameters being studied.

More efficient global optimisation methods may require '*hybrid approaches'*. Ramallo & Coley (Ramallo-González and Coley 2014) have shown that hybrid techniques can reduce the optimisation time considerably and there is therefore scope for GPESO to be hybridised with other techniques to allow more efficient search routines. Further work should also examine other factors such as robust optimisation and uncertainty analysis.

NOMENCLATURE

x,	simulator / emulator input;
$f(\cdot)$,	function representing the simulator;
$\hat{f}(\cdot)$,	function representing the emulator;
$\boldsymbol{D}$,	training data;
$c(x,x)$,	correlation function;
$v(x,x)$,	covariance function;

ACKNOWLEDGEMENTS

This work was supported by the Engineering and Physical Sciences Research Council [EPSRC grant number EP/J002380/1].

REFERENCES

ASHRAE, and US Green Building Council. 2014. *ASHRAE 189.1-2014: Standard 189.1-2014 -- Standard for the Design of High-Performance Green Buildings*. ASHRAE. https://www.ashrae.org/resources--publications/bookstore/standard-189-1.

Bellman, Richard Ernest. 1957. *Dynamic Programming*. Princeton University Press.

Brochu, Eric, Vlad M. Cora, and Nando de Freitas. 2010. "A Tutorial on Bayesian Optimization of Expensive Cost Functions." http://arxiv.org/pdf/1012.2599.pdf.

Bull, Adam D. 2011. "Convergence Rates of Efficient Global Optimization Algorithms." *Journal of Machine Learning Research* 12: 2879–2904.

Design Builder Software Ltd. 2015. "Design Builder." http://www.designbuilder.co.uk/.

Drignei, Dorin. 2009. "A Kriging Approach to the Analysis of Climate Model Experiments." *Journal of Agricultural, Biological, and Environmental Statistics* 14 (1): 99–114.

EDSL. 2015. "TAS." Milton Keynes. http://www.edsl.net/main/Software.aspx.

Fang, K, R Li, and A Sudjianto. 2006. *Design and Modeling for Computer Experiments*. Chapman and Hall / CRC.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.*

Holdaway, Margaret R. 1996. "Spatial Modeling and Interpolation of Monthly Temperature Using Kriging." *Climate Research* 6 (3): 215–25. doi:10.3354/cr006215.

Holland, JH. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.

IES Ltd. 2015. "IES Virtual Environment (www.iesve.com)." http://www.iesve.com.

Jones D.R. 2001. "A Taxonomy of Global Optimization Methods Based on Response Surfaces." *Journal of Global Optimization* 21 (4): 39. doi:10.1023/A:1012771025575.

Jones, DR, M Schonlau, and WJ Welch. 1998. "Efficient Global Optimization of Expensive BlackBox Functions." *Journal of Global Optimization*, no. 13: 345–83.

Krige, DG. 1951. "A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand." *J.Chem.Met.Min.Soc.S.Afr.*, 119–39.

O'Hagan, A. 2006. "Bayesian Analysis of Computer Code Outputs: A Tutorial." *Reliability Engineering & System Safety* 91 (10-11): 1290 – –1300. doi:10.1016/j.ress.2005.11.025.

Ramallo-González, A.P., and D.A. Coley. 2014. "Using Self-Adaptive Optimisation Methods to Perform Sequential Optimisation for Low-Energy Building Design." *Energy and Buildings* 81 (October): 18–29. doi:10.1016/j.enbuild.2014.05.037.

Rasmussen, C. E., and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning. International Journal of Neural Systems*. Vol. 14.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 2012. "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal Of Statistical Software* 51 (1).

Santner, TJ, BJ Williams, and W Notz. 2003. *The Design and Analysis of Computer Experiments. Santner TJ, Williams BJ, Notz W (2003).*

Scrucca, Luca. 2013. "GA: A Package for Genetic Algorithms in R." *Journal of Statistical Software* 53 (4): 1–37.

Sivanandam, SN, and SN Deepa. 2007. *Introduction to Genetic Algorithms.* Springer-Verlag, Berlin.

Tokmakian, Robin, and Peter Challenor. 2013. "Uncertainty in Modeled Upper Ocean Heat Content Change." *Climate Dynamics* 42 (3-4): 823–42.

Uihlein, Andreas, and Peter Eder. 2010. "Policy Options towards an Energy Efficient Residential Building Stock in the EU-27." *Energy and Buildings* 42 (6). Elsevier B.V.: 791–98. doi:10.1016/j.enbuild.2009.11.016.

US Department of Energy. 2013. "EnergyPlus (apps1.eere.energy.gov/buildings/energyplus)." http://apps1.eere.energy.gov/buildings/energyplus/energyplus_about.cfm.

Wood, Mike, Matthew Eames, and Peter Challenor. 2014. "Proof of Concept for the Bayesian Analysis of Computer Code Output in Building Energy Modelling." In *Building Simulation and Optimisation*.