Gaussian-Process-Based Emulators for Building Performance Simulation

Parag Rastogi[1,**], Mohammad Emtiyaz Khan[2], Marilyne Andersen[1]
[1]Interdisciplinary Laboratory of Performance-Integrated Design (LIPID),
Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland.
[2]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan.

Abstract

In this paper, we present an emulator of a building-energy performance simulator. Previous work on emulators for this application has largely focused on linear models. Since the simulator itself is a collection of differential equations, we expect non-linear models to be better emulators than linear models. The emulator we present in this paper is based on Gaussian-process (GP) regression models. We show that the proposed non-linear model is 3-4 times more accurate than linear models in predicting the energy outputs of the simulator. For energy outputs in the range 10-800 kWh/m^2, our model achieves an average error of 10-25 kWh/m^2 compared to an average error of 30-100 kWh/m^2 from using linear models. In addition to being very accurate, our emulator also heavily reduces the computational burden for building designers who rely on simulators. By providing performance feedback for building designs very quickly (in just a few milliseconds), we expect our approach to be particularly useful for exercises that involve a large number of simulations, e.g., Uncertainty Analysis (UA), Sensitivity Analysis (SA), robust design, and optimisation.

Introduction

This paper presents Gaussian process regression models to emulate building-energy performance simulators. In building performance simulation, the use of emulators (regression models) has previously been explored to represent experimentally-observed relationships (e.g., exterior convective heat transfer) and complex or computationally expensive simulations (Rastogi, 2016, ch. 2).

Despite the unsuitability of linear models for emulating an inherently non-linear simulator, the literature is dominated by linear models. This is primarily because linear models are easier to train and interpret. Therefore, the use of non-linear methods must be justified by substantially better performance. For energy outputs in the range 10-800 kWh/m^2, our model achieves an average error of 10-25 kWh/m^2 compared to an average error of 30-100 kWh/m^2 from using linear models. Thus, non-linear models are on average 3 times more accurate than linear models. We also introduce and discuss the results of an automatic method that is used to select relevant inputs. This method is called Automatic Relevance Determination (ARD) which, in addition to reducing the complexity of a model, also establishes the relevance of inputs in predicting the output variables.

A review of the literature suggests a consensus that emulators are most useful in applications that rely on distributions of outputs obtained from hundreds of simulations using Monte Carlo (MC), e.g., Uncertainty Analysis (UA), Sensitivity Analysis (SA), and parametric design exploration (de Wit, 2001; Macdonald, 2002; Hopfe, 2009; Eisenhower et al., 2012; Hygh et al., 2012; Amiri et al., 2015; Nault et al., 2015; Nault, 2016). Recently, non-linear models have been proposed for predicting building energy use and other thermal quantities, e.g., Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and Gaussian process regression (Kalogirou, 2006; Zhao and Magoulès, 2012; Rastogi, 2016). Gaussian process regression has been proposed for optimisation (Wood et al., 2015; Wood, 2016; Gilan and Dilkina, 2015), optimal glazing design (Kim et al., 2013), model calibration (Monari, 2016; Heo and Zavala, 2012; Burkhart et al., 2014), and operational control (Yan et al., 2013).

Most of these proposals have, however, remained theoretical. To the best of our knowledge, none of the prominent building simulation programs offers a regression model as a replacement for the *main* simulator (e.g., ESP-r, EnergyPlus, IDA ICE). Experimentally-derived regression relations are used in sub-components of the simulators, e.g., Percentage People Dissatisfied (PPD) models, but models relating 'final' outputs to 'raw' inputs are not offered (e.g., indoor temperature calculated from wall construction, building layout, etc.).

Simulators and Emulators

Given a set of building-design parameters and weather data[1], the simulator outputs a quantity of interest (e.g., energy used for space heating/cooling). In this paper, we are interested in predicting annual

[**] Corresponding author. Current affiliation: Energy Systems Research Unit (ESRU), University of Strathclyde, UK. Email: rastogi.epfl@gmail.com

[1] Human factors and simulator parameters were not varied in our study.

https://doi.org/10.26868/25222708.2017.448

energy-consumption which can be obtained by integrating the output of the simulator over time. We denote this quantity by a scalar y which takes non-negative values. The vector of inputs, denoted by $\mathbf{x}$, includes the building properties, environmental conditions, and human interactions, as well as (free) tuning parameters in the components of a simulator.

The simulator can be represented by a non-linear function f_s that outputs y given an input vector $\mathbf{x}$:

$$y = f_s(\mathbf{x}). \tag{1}$$

We wish to design an emulator that can predict the value of y as accurately as possible:

$$\hat{y} = f_e(\mathbf{x}), \tag{2}$$

where $f_e(\cdot)$ is the emulator. To achieve this, we can choose a function f_e in the set of functions $\mathcal{F}$ that minimises a cost function, e.g., Mean Square Error (MSE),

$$f_e^* = \arg\min_{f_e \in \mathcal{F}} \ \mathbb{E}[y - f_e(\mathbf{x})]^2, \tag{3}$$

where the expectation is taken with respect to the distribution $p(y, \mathbf{x})$ of the input-output pairs that occur in practice.

Unfortunately, p is unknown, but we can approximate the above expectation by collecting observations generated from p. For example, a building-design expert can collect N inputs $\mathbf{x}_n$ for $n = \{1, 2, \ldots, N\}$, where $\mathbf{x}_n$ denotes an instance of $\mathbf{x}$. She can do this in practice by collecting many basic building designs and weather data from a variety of locations. The outputs y_n can then be obtained by running a simulation on inputs $\mathbf{x}_n$. Thus, the cost in Equation (3) may be estimated by using a sample approximation over these observations (sample mean of squared errors).

We use the standard training-testing framework, which is a common practice in statistics and machine learning (Hastie et al., 2009). Under this framework, we split the N observations into two mutually-exclusive sets. We use the first set of observations to estimate f_e^*:

$$\hat{f}_e^* = \arg\min_{f_e \in \mathcal{F}} \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} [y_n - f_e(\mathbf{x}_n)]^2, \tag{4}$$

where N_{train} denotes the number of observations in the first set (the *training* set). The second set is used to assess the *goodness-of-fit* of the estimator, e.g., by computing the cost,

$$\mathcal{L}(\hat{f}_e^*) = \frac{1}{N_{test}} \sum_{n=1}^{N_{test}} [y_n - \hat{f}_e^*(\mathbf{x}_n)]^2, \tag{5}$$

where N_{test} denotes the number of observations in the second set (the *test* set). By construction, $N = N_{train} + N_{test}$.

This training-testing framework is the backbone of statistical machine-learning. The cost obtained on an *unseen* test observation-set provides a faithful measure of the emulator's performance in the real world. The degree of faithfulness depends on the size and quality of the training and testing data sets. A large amount of training data (N_{train}) would imply a better estimate of $\hat{f}_e^*$, i.e., close to the optimal f_e^*, while a large amount of testing data (N_{test}) would give us a good estimate of the emulator's performance in the real world.

Good-quality datasets faithfully represent the true data distribution $p(\mathbf{x}, y)$ (assuming it exists), but they are not easy to collect because the true distribution is usually unknown. However, for building-performance simulations, we can rely on design experts who can attempt to collect such datasets, e.g., by running Building Performance Simulation (BPS) on realistic building designs and weather conditions. In this work, we used this method. Note that our method is not fail-proof and the data might still contain a systematic bias. Another important point during such data collection is to include all input variables which are expected to be relevant in the modelling of the output. However, too many inputs might make it difficult to obtain a good fit (due to the *curse of dimensionality*), therefore we need to be careful about our selection. Overall, considerations such as these are important to ensure that the quality of the dataset is adequate for a faithful model-fitting.

Model-Based Emulators

For a linear model, we have $f_e = \beta^T \mathbf{x}$ where β is a real-valued vector of the same size as $\mathbf{x}$. The set of functions $\mathcal{F}$ is the set of all linear functions. The solution obtained by minimising Equation (4) is usually called the Ordinary Least-Squares (OLS) solution. In practice, $\mathbf{x}$ might be collinear, e.g., when two entries in $\mathbf{x}$ represent the same underlying variable. This gives rise to ill-conditioning and might make β explode to infinity. In such situations, it helps to impose a distribution over β. In this work, we employ a Gaussian distribution: $\beta \sim \mathcal{N}(0, \mathbf{\Sigma})$ where $\mathbf{\Sigma}$ is a covariance matrix.

We propose to use a non-linear model obtained by a non-linear transformation of the inputs:

$$f_e(\mathbf{x}) = \beta^T \mathbf{\Phi}(\mathbf{x}), \tag{6}$$

where $\mathbf{\Phi}(\mathbf{x})$ is an M-length vector containing various non-linear transformations. That is, $\mathbf{\Phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x})]^T$ where each ϕ_i is a different non-linear function. This is referred to as the basis-function model in machine learning. As an example, the polynomial basis-function for a scalar x defines $\mathbf{\Phi}_i(x) = x^{i-1}$, therefore $\mathbf{\Phi}(\mathbf{x}) = [1, x, x^2, \ldots, x^{M-1}]^T$. A linear model can be obtained as a special case by setting $\mathbf{\Phi}(\mathbf{x}) = \mathbf{x}$.

Gaussian Process and Kernels

We use the Gaussian process regression framework to estimate the function f_e that minimises Equation (4). Gaussian process regression uses Bayes' rule to compute the posterior distribution over f_e given outputs y_n (Rasmussen and Williams, 2006, ch. 2). This approach works directly in the space of f_e and avoids both a direct estimation of β and a direct specification of $\phi(\mathbf{x})$. Instead, a 'kernel' function specifies the inner products of ϕ as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \, \mathbf{\Sigma} \, \phi(\mathbf{x}_j), \tag{7}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two inputs in our observation set. In practice, a kernel function is easier to specify than ϕ, even though it could be unintuitive.

A variety of models can be obtained this way, e.g., a linear model is obtained by using the linear kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{\Sigma} \mathbf{x}_j. \tag{8}$$

In this paper, we will compare the linear model to the following nonlinear model which is obtained by using a Squared Exponential (SqE) kernel function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \, \exp \left[-\frac{1}{2} (\mathbf{x}_i - \mathbf{x}_j)^T \, \mathbf{\Sigma} \, (\mathbf{x}_i - \mathbf{x}_j) \right], \tag{9}$$

where $\sigma_f^2 > 0$ is the signal variance. This kernel is also referred to as the Radial Basis Function (RBF) kernel in the context of Artificial Neural Network (ANN).

Input-Variable Selection

Nonlinear models are powerful and flexible, which is good when we have a large data set of good quality, but otherwise nonlinear models might 'over-fit', i.e., they might fit to the noise instead of the signal. To avoid this, one solution is to reduce the complexity of the model (number of input or explanatory variables).

In this paper, we use an automatic variable selection method called ARD (Rasmussen and Williams, 2006, sec. 5.1). In this method, we set $\mathbf{\Sigma}$ to be a diagonal matrix with each diagonal entry $\Sigma_{ii} = 1/l_i$ where $l_i > 0$, and then estimate $\mathbf{l} = [l_1, l_2, \ldots, l_M]$. As $l_i \to \infty$ (or $1/l_i \to 0$), the importance of the corresponding input dimension x_i goes to 0. We use the log-marginal likelihood of the GP regression model to estimate $\mathbf{l}$ and other parameters of the kernel (Rasmussen and Williams, 2006, Eq. 2.30). We call this type of models the ARD models.

We compare these models to those in which there is no variable selection, i.e., we set all l_i to one value l. We call these models isotropic (ISO) models since this choice makes the Gaussian distribution on β an isotropic Gaussian distribution.

List of Models

Table 1 lists all the models we compare in our experiments. The first model is the 'Mean' model which uses the sample mean of the outputs in the training set as the prediction. This model ignores all the inputs and is expected to perform significantly worse than the models that use them. We compare the baseline to the two versions of linear models and nonlinear models each, as shown in Table 1. We expect ARD to be more useful for non-linear models compared to linear models. This is because there is less danger of over-fitting with linear models.

Table 1: List of models compared in this study.

Model	Description
Mean	Mean of the outputs y_n
Lin-ISO	Linear model with ISO
Lin-ARD	Linear model with ARD
NonLin-ISO	SqE kernel with ISO
NonLin-ARD	SqE Kernel with ARD

The Automatic Relevance Determination (ARD) procedure may or may not establish the practical relevance of the input variables, because it only determines their relevance in predicting the output variables. This procedure is affected by the estimation procedure and the amount of data, e.g., the minimiser might get 'stuck' in a local minimum due to poor initialisation. This is more likely when we do not have enough data to estimate the marginal likelihood.

Dataset Description

The case studies are based on abstract representations of typical buildings, taken from the United States Department of Energy (USDOE) Commercial Buildings Reference Database (Deru et al., 2011) (referred to as the 'USDOE' series after this). Details about how the data were produced can be found in Rastogi (2016, sec. 4.2 and B.4). The buildings were modelled without Heating, Ventilation, and Air Conditioning (HVAC) systems, in EnergyPlus v8.5 (NREL and USDOE, 2015). The weather data used are described in Rastogi (2016, sec. A.5 and B.4). Seventeen different building types, distinguished by usage, were simulated (e.g., hospitals, apartments). In addition to a different usage profile or programme, each type had a different layout and arrangement of rooms. Further variety was introduced by using three wall construction profiles for each building type.

Using the USDOE database and weather data from several climates (regions) worldwide, we obtain a total of $N = 88242$ input-output pairs. Each input $\mathbf{x}_n$ contains 28 dimensions, i.e., 28 input variables, which are described in Table 2. We used two outputs, $y_n^{heating}$ and $y_n^{cooling}$, the ideal heating and cooling energy consumption respectively. Both these outputs were obtained by summing the respective hourly loads over one year to give the ideal heating or cooling energy consumption. The distribution of these

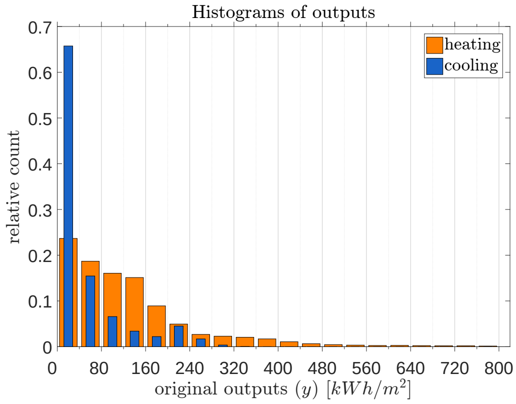


Figure 1: Distribution of the outputs $y_n^{heating}$ for heating loads and $y_n^{cooling}$ for cooling loads. The weight of the distributions is skewed towards lower values because we selected, unintentionally, more moderate climates (climates that need less cooling/heating or none) than extreme ones (need more heating, cooling, or both). Each building produces a slightly different distribution by itself – all simulations are plotted together here.

outputs is shown in Figure 1. We fit two separate regression models for $y_n^{heating}$ and $y_n^{cooling}$, although it is also possible to use one model to predict both the outputs.

Even though our dataset contains multiple building and weather types, we only train one emulator. It is possible to train different models for each type, but then the amount of data for each type will be substantially smaller. There are other issues, e.g., deciding which model to use during testing, or how to capture correlations between different subtypes. Using one powerful non-linear model, that can capture complex dependencies in the data, avoids all these issues. Such an approach performs much better than simple linear models, as we show in this paper.

All inputs and outputs were standardised, i.e., we computed their means and standard deviations, subtracted the mean and then divided by the standard deviation. This way all the inputs and outputs lie within the same range, which improves the robustness of the numerical procedures used during estimation. The original data and results shown in this paper may be downloaded from `https://doi.org/10.5281/zenodo.291858`. The code may be downloaded from `https://github.com/paragrastogi/GPregressionInBS`.

Results

We now compare the models shown in Table 1 using the training-testing framework discussed earlier. In our first experiment, we investigate the effect of the amount of training data on prediction quality. We expect all models to perform badly when the amount of training data is limited. As the amount of data is increased, we expect nonlinear models to perform better than linear models.

To estimate the real-world performance of a model, we set aside about 60% of the 88,242 observations (simulations) for testing ($N_{test} = 52,945$) and use parts of the remaining data set to train the model. Note that the test set is always 'left out', i.e., a model never uses the data in the test set for training and is completely unaware of it. The test data is fairly large and therefore we expect it to produce a faithful estimate of the real-world prediction error of the model. We present the Root Mean Square Errors (RMSEs) on the test set, which are obtained by taking the square root of Equation (4).

The training dataset size N_{train} varies from 50 to 4000. For a given training size N_{train}, we draw that many training observations from the large overall training set and use it to train a model. We then predict the left-out test data using the trained model and compute RMSE. We repeat this process 100 times to obtain an empirical distribution of the RMSE estimate. This gives us a confidence estimate which can be used in a significance test.

Figure 2 shows the results for heating loads in the left column and cooling loads in the right column. Figures 2a and 2b show the evolution of RMSE as a function of the size of training data for heating and cooling loads respectively. Each curve in the plots shows the performance of a model from Table 1. The thick line shows the median of the RMSE distribution while the shaded area around the thick line shows the 25th and 75th percentiles. As N_{train} increases, all models perform much better than the 'Mean' model, which only uses the mean of the output from the training data and completely ignores the inputs. The linear models, for example, are about 25% better, while the non-linear models are 50-75% better (depending on the type of nonlinear model). This comparison is similar to how the R^2 is calculated for a fit to data. If a model performs worse than a mean of the training data, i.e., a horizontal line fit, then it should be discarded.

Figures 2c and 2d show the empirical distributions of (the absolute value of) prediction errors for two different training set sizes, $N_{train} = 100$ and $4,000$ respectively. These errors correspond to one of the 100 runs shown in Figures 2a and 2b. The errors obtained by using the larger training set are much smaller than those obtained by using a smaller training dataset, as expected. With more training data, we are consistently able to reduce larger values of error. Figures 2e and 2f show a similar comparison between linear and nonlinear models. The nonlinear model too decreases the number of larger mistakes.

Overall, the performance of the linear models is uniformly worse than that of the non-linear models. As expected, when the amount of data is small, the performance of the linear and non-linear models is comparable. However, the error curves for the linear model plateau very quickly and the performance does not improve with the amount of data increased. This shows that the linear models are not adequate for modelling the complex nonlinear data, and are unable to use the information present in the data. On the other hand, the non-linear models continue to improve as the amount of data is increased. Between the two nonlinear models, the model with ARD performs significantly better. This shows that reducing the number of features improves the performance. This is also expected since a nonlinear model with too many features might overfit, and reducing the feature dimensionality reduces this problem.

We now discuss the relevance of features found by using the ARD method on the Gaussian process (GP) regression model with a nonlinear kernel (SqE). Recall that as $1/l_i \to 0$, the relevance of the feature x_i reduces (compared to other features and assuming that the range of features are roughly in the same order of magnitude). As discussed earlier, the selection of features is affected by the estimation procedure and the amount of data. We present results for $N_{train} = 4,000$ since the model performance seems to plateau around this training data size (Figures 2a and 2b). To see the artifacts (inconsistencies) introduced by the estimation procedure, we plot the empirical distribution of $1/l_i$, obtained from the 100 runs for $N_{train} = 4,000$. For irrelevant variables, we expect $1/l_i$ to be nearly zero *most* of the time.

Figure 3 contains empirical estimates of the distribution of $1/l_i$ for a feature. The name of the feature is shown in the title and its description is in Table 2. The distribution is estimated over a 100 values and normalised (scaled to 0-1). Both x and y axis are limited to 0-1 for easy visualisation. We show only those inputs which take a non-zero value at least 20% of the time. We now discuss the consequences of these results.

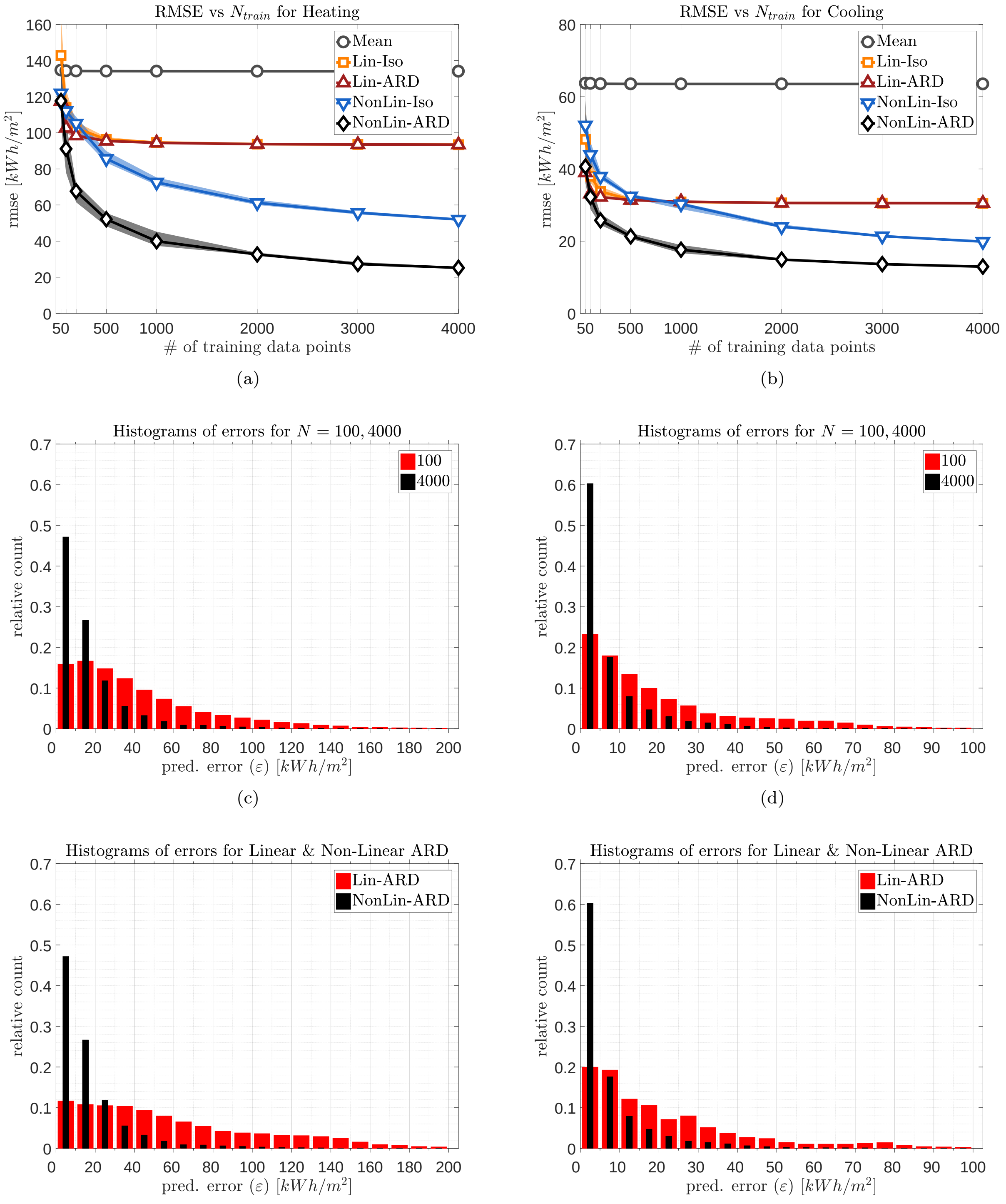
Only two inputs, Window-to-Wall Ratio and Window-to-Floor Ratio, are consistently significant, for both models. The Internal Heat Gain variable (*sumihg*) is significant in the heating model but missing in the cooling model. Also in the heating model, the influence of mean sunlit percentage (*msunp*) is small, though consistent, while that of the volume-to-wall-area ratio (form-factor or *ff*) is occasionally moderate. The median and Inter-Quartile Range (IQR) of Dry Bulb Temperature (TDB), *medtdb* and *iqrtdb*, are weakly significant for cooling models, though not for heating. While the significance of Window-to-Wall Ratio (WWR) and Window-to-Floor Ratio (WFR) is not surprising, the presence of *both* in a single fit is unexpected, given their high correlation (Rastogi, 2016, fig. 4.3). The dominance of these variables could be because the buildings we chose are more driven by envelope (fenestration) loads. The IQR of TDB in a year tends to be moderately correlated with its median (Rastogi, 2016, fig. 4.3). We guess that the solar parameters are missing because their effect is well accounted for by the envelope ratios: WWR and WFR. Overall, the dominance of building parameters (WWR, WFR, and Internal Heat Gain) is not surprising in what is largely a sample from commercial buildings with medium to high fenestration ratios. The almost complete rejection of all climate parameters *is* surprising, however, and we expect may be different in a data set dominated by residential buildings.

Discussion

Emulators are fast-response surrogates for simulators, delivering comparable results in a fraction of the time. A typical BPS run (i.e., a building design with one set of weather data and operating conditions) may take from a few minutes for very simple models, to hours for reasonably detailed ones. In addition, the accuracy of the 'predicted' energy use of a design depends on the accuracy of the inputs, and whether the parameters of the simulator have been set appropriately. Even if all fixed inputs (like building properties) are known with perfect accuracy, and the tuning parameters of a simulator are set to some ideal values, the presence of random inputs like weather and human factors means that the simulator can not deliver exact (accurate) predictions. A standard approach to accounting for the effect of random inputs is to use MC analyses, but this usually requires a large number of BPS runs, significantly increasing the computation time. For such computationally-intensive tasks, emulators can be much faster while being reasonably accurate.

Building simulation is essentially a solution of several interconnected partial and ordinary differential equations, feeding human preference models and being fed by empirical models for individual components, so it cannot reasonably be expected to conform to a linear regime. We do not support the idea of large databases of pre-simulated cases being emulated with linear regressors, where a new design problem may be 'located' and its energy use estimated. We argue, instead, that the benefits and performance of a nonlinear emulator outweigh the costs of training it on anything from a few dozen to a few hundred simulations (depending on the complexity of the simulation model and design exercise, usually indicated by the number of building and environmental parameters used as input variables). The response from a query would be instantaneous during a design exploration or MC exercise.

Emulators also heavily reduce the computational burden for building designers who rely on simulators. For example, the emulator can first be trained with observations from the simulator using a wide variety of building designs and weather data. This pre-trained model can then be used by building designers for



(a)

(b)

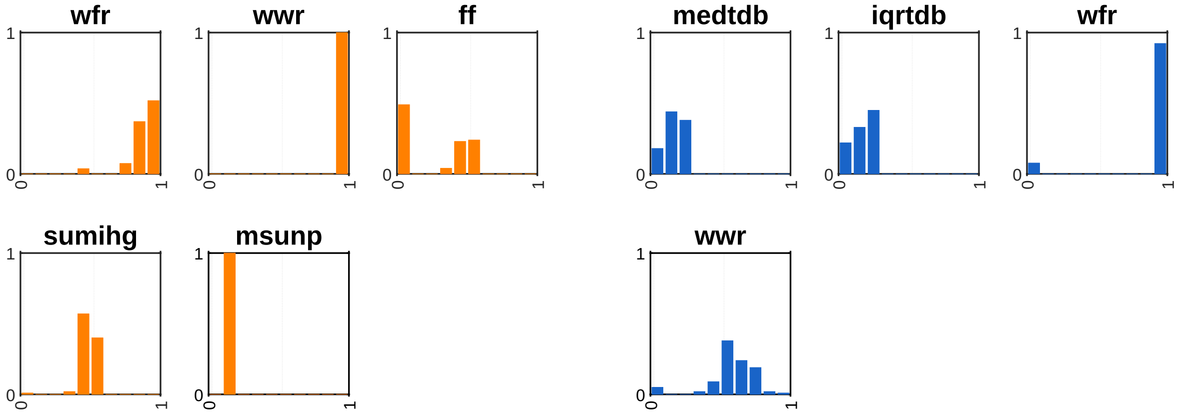
(c)

(d)

(e)

(f)

Figure 2: Plots on the left show results for the heating loads and on the right show results for the cooling loads. Figures a and b show the Root Mean Square Error (RMSE), plotted against the size of the training data set. The lines indicate median errors, and the filled areas are bounded by the 25th and 75th percentiles. The non-linear models outperform linear models, and the predictions improve with size of training data set. Figures c and d compare the empirical estimates of distributions of absolute errors, $|\varepsilon| = |y_j - f_e(\mathbf{x}_j)|$, for $N_{train} = 100$ and $4,000$. We see that a large training data size reduces larger values of error. Figures e and f compare the same for the linear and non-linear models, where we see that the nonlinear model makes fewer errors.



(a)

(b)

Figure 3: Empirical estimates of the relevance parameters $1/l_i$ *for significant input variables – heating (Figure 3a) and cooling (Figure 3b) loads. A value of* $1/l_i$ *close to zero, for a particular input, indicates that that input was not involved in determining the prediction of the model. A significant input is one for which at least one non-zero value of* $1/l_i$ *has a relative share greater than 20% of the total samples. The title of each plot is the variable code (see Table 2 for an explanation of the variable codes). The histogram is obtained from the 100 runs over training-data subsets of size* $N_{train} = 4,000$.

exploration of new designs by predicting the performance of new buildings very quickly. The emulators respond almost instantaneously to a query, regardless of the complexity of the building design being probed, which is not the case for a simulator. The EnergyPlus simulations that constitute the training/testing data for this paper each took between 15 seconds and 9 minutes on a laptop. The training of a regression model with, e.g., 4000 inputs takes far longer (between 1 and 3 hours, depending on the type of regression model used). Clearly, the effort of training a regression model is not justified in the case of a handful of simulations (e.g., when the user is checking a design for compliance with local energy laws). However, the user of a hypothetical future regression-model-based design exploration system need not concern themselves with the effort of training it. Since the regression model can sit on a remote server to be queried, the user is only interfacing with the emulation as if they were working in a building simulation program.

In the development of emulators so far, BPS has been used as the 'ground truth', which means that the emulators are trained to faithfully reproduce the responses of the simulator (Rastogi, 2016, ch. 2). We do not, as yet, have well-developed case studies with measured data to compare the performance of emulators against simulators. While it is clear that neither simulators nor emulators can model reality *exactly*, emulators can be retrained when better data becomes available while simulators cannot. In this sense, emulators are both fast and flexible. Calibrating a simulator using real data is a manual process whose conduct depends on the judgement of the user, and is not comparable to retraining a regression model.

Conclusion

In this paper, we presented non-linear and linear Gaussian process regression models for emulating ()bps. We showed that nonlinear models achieve RMSE values that are 3-4 times lower than those of linear models. Finally, we discussed the results from using fits in which the input variables are selected using the ()ard method.

Gaussian process regression offers a promising and flexible approach to creating emulators for building performance simulation. We have built upon previous work (both ours and from the literature), to present alternatives to classical linear regression in the form of Gaussian process regression. The proposed regression models, though more expensive to train than linear models, give more accurate predictions. We expect to further refine these and other regression methods in future studies for use in emulating BPS. The development of easy-to-use regression modules for BPS programs, which help a user to probe and customise a regression model for their design problem, is ongoing. We also plan to test Gaussian process regression and other machine learning approaches, like ANN and random forests, on larger data sets with many more features.

The use of emulators greatly eases the process of quantifying uncertainty and sensitivity in building simulation, an argument we have previously summarised in Rastogi (2016). The effort needed to simulate enough data for training may be lessened by using a combination of a large variety of automatically-sampled random inputs, e.g., weather, and a manageable number of manually-created 'design variants', e.g., 5-6 combinations of building properties. This is how the data used in this study was generated. In future work, we expect to increase the features used in the model to account for more aspect of the building. The emulators may also be extended to deliver time series of temperatures, energy, etc., which may be more informative in making decisions. These emulators are envisioned as part of a larger project to use emulators for machine-assisted design exploration.

Acknowledgement

The authors would like to thank Dr. Carlos Becker, for his patient and valuable support, and Professor Anthony Davison, for his incisive and constructive criticism. Parag was funded by the EuroTech Universities Alliance and the SECURE (Synergistic Energy and Comfort through Urban Resource Effectiveness) project funded by the CCEM (Competence Centre for Energy and Mobility) during his work at EPFL. This work was partly carried out at the University of Tokyo (Sugiyama-Sato group) and the RIKEN Center for Advanced Intelligence Project.

References

Amiri, S. S., M. Mottahedi, and S. Asadi (2015, December). Using multiple regression analysis to develop energy consumption indicators for commercial buildings in the U.S. *Energy and Buildings 109*, 209–216.

Burkhart, M. C., Y. Heo, and V. M. Zavala (2014, June). Measurement and verification of building systems under uncertain data: A Gaussian process modeling approach. *Energy and Buildings 75*, 189–198.

de Wit, S. (2001, June). *Uncertainty in predictions of thermal comfort in buildings.* PhD, Delft University of Technology, Delft, The Netherlands.

Deru, M., K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M. Rosenberg, M. Yazdanian, Y. J. Huang, and D. B. Crawley (2011, February). U.S. Department of Energy commercial reference building models of the national building stock. Technical report, National Renewable Energy Laboratory (NREL).

Eisenhower, B., Z. O'Neill, S. Narayanan, V. A. Fonoberov, and I. Mezić (2012, April). A methodology for meta-model based optimization in building energy models. *Energy and Buildings 47*, 292–301.

Gilan, S. S. and B. Dilkina (2015). Sustainable Building Design: A Challenge at the Intersection of Machine Learning and Design Optimization. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence.*

Hastie, T., R. Tibshirani, and J. Friedman (2009, August). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer.

Heo, Y. and V. M. Zavala (2012, October). Gaussian process modeling for measurement and verification of building energy savings. *Energy and Buildings 53*, 7–18.

Hopfe, C. J. (2009). *Uncertainty and sensitivity analysis in building performance simulation for decision support and design optimization.* PhD, Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

Hygh, J. S., J. F. DeCarolis, D. B. Hill, and S. R. Ranjithan (2012, November). Multivariate regression as an energy assessment tool in early building design. *Building and Environment 57*, 165–175.

Kalogirou, S. A. (2006). Artificial neural networks in energy applications in buildings. *International Journal of Low-Carbon Technologies 1*(3), 201–216.

Kim, Y.-J., K.-U. Ahn, C. Park, and I.-H. Kim (2013). Gaussian emulator for stochastic optimal design of a double glazing system. In *Proceedings of the 13th IBPSA Conference, August*, pp. 25–28.

Macdonald, I. (2002). *Quantifying the Effects of Uncertainty in Building Simulation.* Doctoral, University of Strathclyde.

Monari, F. (2016). *Sensitivity Analysis and Bayesian Calibration of Building Energy Models.* PhD, University of Strathclyde, Glasgow, UK.

Nault, E. (2016). *Solar Potential in Early Neighborhood Design. A Decision-Support Workflow Based on Predictive Models.* PhD Thesis, Ecole polytechnique fédérale de Lausanne, Lausanne, Switzerland.

Nault, E., P. Rastogi, E. Rey, and M. Andersen (2015, September). The sensitivity of predicted energy use to urban geometrical factors in various climates. In *Proceedings of PLEA 2015*, Bologna, Italy.

NREL and USDOE (2015). EnergyPlus.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian processes for machine learning.* Adaptive computation and machine learning. Cambridge, Mass: MIT Press.

Rastogi, P. (2016, August). *On the sensitivity of buildings to climate: the interaction of weather and building envelopes in determining future building energy consumption.* PhD, Ecole polytechnique fédérale de Lausanne, Lausanne, Switzerland. doi:10.5075/epfl-thesis-6881.

Wood, M., M. Eames, and P. Challenor (2015, December). A comparison between Gaussian Process emulation and Genetic Algorithms for optimising energy use of buildings. In *Proceedings of BS 2015*, Hyderabad, India. IBPSA.

Wood, M. J. (2016, February). *An exploration of building design and optimisation methods using Kriging meta-modelling.* PhD, University of Exeter, Exeter, UK.

Yan, J., Y.-J. Kim, K.-U. Ahn, and C.-S. Park (2013). Gaussian process emulator for optimal operation of a high rise office building. In *Proceedings of BS 2013*, Chambéry, France.

Zhao, H.-x. and F. Magoulès (2012, August). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews 16*(6), 3586–3592.

Table 2: Initial inputs, codes, and units.

Group	Quantity	Statistic	Name	Code	Units
Building	U-value	Average	Average U-value of envelope	*uval*	W/m^{2}K
	Thermal Mass	Sum	Sum of thermal storage capacity	*tmass*	MWh/K
	Envelope Ratios	Ratio	Ratio of window area to wall area	*WWR*	—
			Ratio of window area to floor area	*WFR*	
	Massing	Ratio	Form Factor (Volume / Wall Area)	*ff*	—
			Roof Ratio (Roof / Wall Area)	*rr*	
Mixed	Shading	Average	Average sunlit percentage of envelope	*msunp*	%
	Infiltration	Sum	Annual sum of energy gained from infiltration	*sinfg*	GWh
			Annual sum of energy lost to infiltration	*sinfl*	
	Internal Heat Gain		Annual sum of Internal Heat Gain	*sumIHG*	GWh
Climate	Degree Days	Sum	Annual sum of cooling degree days	*cdd*	°C-day
			Annual sum of heating degree days	*hdd*	
	Dry Bulb Temperature (Hourly)	Average	Annual average of dry bulb temperature	*avgtdb*	°C
		Median	Median dry bulb temperature	*medtdb*	
		IQR	Inter-quartile range of dry bulb temperature	*iqrtdb*	
	Dew Point Temperature (Hourly)	Average	Annual average of dew point temperature	*avgtdb*	°C
		Median	Median dew point temperature	*medtdb*	
		IQR	Inter-quartile range of dew point temperature	*iqrtdb*	
	Global Horizontal Irradiation (Hourly)	Average	Annual average of global horizontal irradiation	*avgghi*	MWh/m^2
		Sum	Annual sum of global horizontal irradiation	*sumghi*	
		IQR	Inter-quartile range of global horizontal irradiation	*iqrghi*	
	Direct Normal Irradiation (Hourly)	Average	Annual average of direct normal irradiation	*avgdni*	MWh/m^2
		Sum	Annual sum of direct normal irradiation	*sumdni*	
		IQR	Inter-quartile range of direct normal irradiation	*iqrdni*	
	Humidity (Hourly)	Average	Annual average of relative humidity	*avgrh*	%
		Median	Median relative humidity	*medrh*	