# Model Predictive Control of Non-domestic Heating Using Genetic Programming Dynamic Models

Tiantian Dou, Yuri Kaszubowski Lopes, Peter Rockett

*Department of Electronic and Electrical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK.*

Elizabeth A Hathway, Esmail Saber[1]

*Department of Civil and Structural Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK.*

**Abstract**

We present a novel approach to obtaining dynamic nonlinear models using genetic programming (GP) for the model predictive control (MPC) of the indoor temperatures of buildings. Currently, the large-scale adoption of MPC in buildings is economically unviable due to the time and cost involved in the design and tuning of predictive models by expert control engineers. We show that GP is able to automate this process, and have performed open-loop system identification over the data produced by an industry grade building simulator. The simulated building was subject to an amplitude modulated pseudo-random binary sequence (APRBS), which allows the collected data to be sufficiently informative to capture the underlying system dynamics under relevant operating conditions.

In this initial report, we detail how we employed GP to construct the predictive model for MPC for heating a single-zone building in simulation, and report results of using this model for controlling the internal environmental conditions of the simulated single-zone building. We conclude that GP shows great promise for producing models that allow the MPC of building to achieve the desired temperature band in a single zone space.

*Keywords:* Genetic Programming, Dynamic Non-Linear System Identification, Model Predictive Control, Building Energy Management

## 1. Introduction

Model predictive control (MPC)[1] is a powerful control methodology well-suited to systems in which there is an appreciable delay between an input being

[1]Current address: Civil & Building Services Engineering Division, School of the Built Environment and Architecture, London South Bank University, London SE1 0AA, UK

applied and any observable response, and which may also have control constraints; large (non-domestic) buildings are among such systems. Central to MPC is a predictive model of the dynamics of the system being controlled. Given a prediction horizon extending some number of discrete time steps into the future, the controller optimises the sequence of future inputs by minimising some objective function. Typically, this objective comprises a weighted sum over the prediction horizon of the deviations from a desired setpoint together with the control effort, the magnitudes of the control changes. This latter term is designed to penalise rapid switching of the input and hence minimise actuator wear. At every time step, the future input sequence is optimised, the first of this input sequence applied to the system and the whole process repeated at the next time step. This forever advancing prediction interval gives the technique its alternative name of receding horizon control.

Although MPC has been widely employed in the chemical process industries, where it had its origins, applications to buildings are currently only at the research stage – see, for example, Rockett and Hathway [2] for a review. Critical to MPC, whatever its domain of application, is the performance of the predictive model.

The generally superior climate control of MPC in buildings compared to conventional rule-based approaches appears to offer the potential for significant energy savings – maybe up to 25% [2] – and makes buildings MPC worth pursuing in order to reduce $CO_2$ emissions and to improve internal environmental quality. However, at a roundtable discussion at a workshop on MPC in buildings held in Montréal in 2011, Henze [3] noted attendees estimated 70% of total costs for MPC implementation were consumed by the creation and calibration of the predictive model that lies at the heart of MPC. In fact, this figure agrees with the 75% often quoted by the wider process-control community [4]. Traditionally, such models are produced by extensive fine-tuning by highly skilled control engineers. Although the high cost of predictive model creation may be tolerable in the highly-capital intensive environment of petrochemicals, Rockett and Hathway [2] have pointed out that such high costs currently make MPC economically unviable for the control of buildings. It is, therefore, critical for the economic uptake of MPC in buildings to create predictive models of the system dynamics using machine learning-based methods that can learn from data obtained from the building in operation rather than be hand-crafted by experts. Further, the characteristics of buildings change over time, either due to changes in use, internal alterations, or indeed external factors, such as the erection/demolition of adjacent buildings that change the solar gains or façade wind pressures on the building under control. Such changes will change the dynamics of the building and necessitate a recalibration of the predictive model in order to maintain optimised control. Rapid and low-cost recalibration without human intervention is thus also essential to maximise the ongoing benefits of MPC in buildings.

Buildings are widely acknowledged to exhibit non-linear dynamics and therefore require a non-linear predictive model. For example, in the situation described in the present paper (see Section 3.2), the heat transfer from a conventional hydronic radiator to the room space is non-linear [5, 6]; the solar energy entering a building via a window was found to be non-linear function of incident radiation by Sturzenegger et al. [7]. The problem of formulating a non-linear predictive model has been discussed in a seminal paper by Sjöberg et al. [8]. Assuming sampling at discrete, equally-spaced time steps, the one-step-ahead (OSA) prediction $\hat{y}_{k+1}$ of a dynamical system at time $(k+1)$ is given by:

$$\hat{y}_{k+1} = f(\mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n}, y_k, y_{k-1}, \ldots, y_{k-m}) \tag{1}$$

where $\mathbf{u}$ is a vector of inputs, or so-called *exogenous* variables. The problem is to identify i) $f$, the non-linear function, ii) the value of $n$ dictating how many of the previous inputs need to be considered, and iii) the value of $m$, the number of previous (autoregressive) outputs to be included. The sets of delayed variables $\{\mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n}\}$ and $\{y_k, y_{k-1}, \ldots, y_{k-m}\}$ are usually termed *lag sets* and compactly incorporate the 'inertia' of the controlled system. To implement MPC we generally need a model that produces a set of accurate future predictions over the so-called *prediction horizon*, that is, $N$ time steps into the future.

In principal, the search for $f$ in (1) is over the set of all possible functions, but in practice $f$ is often restricted to families, such as Volterra functions or neural networks [9]. Identification of the lag sets (i.e. the best combination of values of $n$ and $m$) is typically performed iteratively in a manner highly dependent on the expertise of a control engineer.

Neural networks (NNs) have been widely used for nonlinear dynamic system identification. In order to enhance the accuracy while minimising the model size, an architectural refinement stage is often required. For instance, NeuroEvolution of Augmenting Topologies (NEAT) [10] uses a genetic algorithm (GA) to evolve both model structure and the associated parameters of neural network models.

A further consideration with Volterra approximators and, especially, neural networks is the large number of parameters that have to be estimated during training, which implies a requirement for a large amount of training data. Moreover, with reference to (1), while training NNs can approximate the function $f$, determining the lag sets specified by $n$ and $m$ usually requires the embedding of the NN training within some global search for the network inputs determined by $n, m$, the so-called *feature selection* problem, i.e. a search problem embedded within another search problem.

To address the challenge specified by (1), an increasing number of researchers have applied genetic programming (GP) to nonlinear dynamic systems identification problems [11, 12] due to the advantage of being able to automatically optimize both model structure and its parameters simultaneously during evolution. Basic GP, however, has often been used to evolve the function $f$ either as a simple regression problem (i.e. without the autoregressive terms $y_k, y_{k-1}, \ldots, y_{k-m}$), or using pre-defined lags sets, that is, pre-specification of $n$ and $m$ in (1).

Rodríguez-Vázquez and Fleming [13] used GP to identify a number of dynamical systems. Grosman and Lewin [11] used GP to generate an empirical dynamic model of a process, a mixing tank, and then applied it in a nonlinear model predictive control (NMPC) strategy. The results show that the GP method provided significantly better regulatory and servo performance than more traditional control approaches. Recently, Feng et al. [12] also investigated the performance of GP on non-linear dynamical systems and NMPC, and claimed that a GP based predictive controller can obtain satisfactory performance.

In the model training stage, however, Rodríguez-Vázquez and Fleming, Grosman and Lewin [11], and Feng et al. [12] employed user-specified, pre-determined lag sets, which are normally very time-consuming to determine manually in practical applications.

Hinchliffe and Willis [14] also used GP to evolve discrete-time models of dynamic processes, however, evolution of the appropriate lag set of input variables was included by adding unary back-shift (i.e. time lag) operators to the GP's function set. The experimental results suggest that the performance of GP shows little difference with filter-based neural networks in terms of model accuracy on an extruder case study. The significant point in Hinchliffe and Willis' [14] work is that their GP formulation is not only able to approximate model structure ($f$), but also construct appropriate lag sets and not require their pre-specification.

Taking advantage of the fact that the Hinchliffe and Willis GP scheme is able to evolve both model structure and lag sets automatically during the evolution process, in this paper, we describe the use of genetic programming for creating the dynamic model necessary for buildings MPC. We believe this to be the first report of the demonstration of buildings MPC using learned GP models. As is common in the control field, we have considered a system simulation in order to rapidly and comprehensively explore the issues involved.

To further underscore the advance made in this paper, it is worth briefly reviewing the process currently used for constructing a grey-box predictive models of buildings. Following the much-cited paper by Hazyuk et al. [6], typically analogous resistor-capacitor (RC) linear networks comprising various numbers of R's and C's, each network representing the different physical elements (walls, floors, rooves, etc), are manually assembled from expert knowledge of the building. This overall composite RC network is combined with injected heat gains from the heating system, solar radiation through windows, etc. (modelled as voltage or current sources) to produce an overall state-space model. It is important to note that the heating and solar inputs are non-linear functions of their controllable variables. For example, energy transfer from a hydronic radiator is a non-linear function of the mean water temperature in the radiator – see [6] for details. Having hand-assembled a model structure, it is necessary to identify the model's parameter values, a task which is generally regarded as "difficult" [6, p.385], and requires input/output response data from the building; this process is typically performed using non-linear least-squares fitting. Unfortunately, parameter identification is sometimes problematic due to *unidentifiability* – the inability to sufficiently accurately determine a parameter's value due to numerical deficiencies of the model [15]. Overcoming unidentifiability requires judicious modifications to the model followed by re-estimation until the conditions for identifiability are met. At that point, the calibrated model can be validated against data from the building independent of that used for parameter calibration; if the predictive ability of the model is insufficient, the whole process is iterated until a satisfactory model structure is found. The amount of highly-skilled human intervention at every stage of this process directly motivates the *automation* of the construction of predictive models to render MPC economically-viable for buildings. A credible route to this automation is the principal contribution of the present paper.

In Section 2, we describe genetic programming for modelling dynamical systems and give an example for a benchmark problem from the chemical engineering literature. We describe the building control methodology we have used in Section 3 together with the procedures necessary for successfully identifying a predictive GP model of the test building. In Section 4 we report typical results of the performance of the predictive GP model as well as the performance of the model predictive control scheme. In this paper, we present only representative, typical results and defer detailed discussion of parameter settings, etc. to a future publication. We do, however, consider these issues together with future work in Section 5. We conclude the paper with Section 6.

## 2. Genetic Programming

Inspired by biological evolution processes, evolutionary algorithms (EA) solve problems by applying the theory of natural selection to a population of individuals with the expectation of evolving fitter models. Genetic algorithms (GA) are one class of EA. Basically, a GA consists of a reproductive strategy for generating offspring with better fitness using the principal genetic operators of crossover and mutation. GP is a subset or an extension of GA. The essential principles of GA and GP are similar although solutions in GP are expressed as programs with hierarchical tree structures, which consist of pre-specified functional and terminal nodes. This flexible tree structure provides a dynamic and variable representation. A typical example of such a GP tree is shown in Figure 1, and its functional expression is given in (2).

$$y(k) = (-y_{k-1}) + (0.2 * (-u_{k-1})) \tag{2}$$

Generally, evolutionary algorithms can be classified into two different types: steady-state and generational. In steady-state evolution, one (or two) offspring are produced at each step and appended to the population; the population size is then reduced down to its original size by removing the weakest one (or two) individuals. (In fact, the term 'steady-state' is a misnomer – quasi-steady state would be more accurate.) In generational algorithms, on the other hand, a whole new child population is produced by repeated selection from a parent population before the child population is swapped to become the parent population and the process repeated. In this paper, steady-state GP is used since this appears

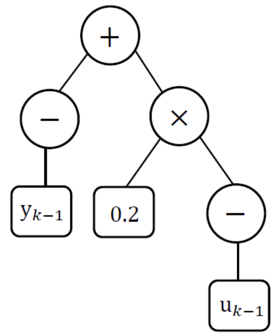


Figure 1: Simple example GP tree.

to yield superior search results [16]. The following algorithm describes the evolution process of a typical steady-state GP.

- Step 1: Population initialization

  In GP, candidates in the initial population are randomly generated. The process of creating random trees can be implemented in different ways [17], but two simple methods (the 'full' and 'grow' strategies [17]) are extensively used. In the 'full' method, the initial trees are created up to a predefined maximum depth; the depth of a GP tree is the minimum number of edges that need to be traversed to reach the deepest leaf starting from the tree's root node. Trees are generated by randomly selecting nodes from the function set until all the leaves reach the maximum tree depth. In the 'grow' method, trees are created with more diverse structures with some probability of terminating tree growth before reaching the depth limit.

  In order to initialize the population of trees with a variety of shapes and sizes, Koza [18] proposed a *ramped half-and-half* method where half the initial population is created using the 'full' method and half using the 'grow' method. In the present work, we have used this *ramped half-and-half* method for population initialization.

- Step 2: Fitness evaluation

  The performance of each tree is evaluated with a fitness function, which is used for estimating how well a solution performs on the given problem. Then the population is sorted according to fitness value. Solutions with higher ranks are more likely to be selected as parent trees to breed child candidates in the evolution process.

- Step 3: Offspring generation

  At each iteration, two GP trees are selected as parents. Two main genetic operations, crossover and mutation, are then applied to produce new offspring solutions. Specifically, the crossover operator randomly selects a crossover point in each parent tree. The child trees are then generated by crossing over and splicing together the two trees at the selected crossover points between two parent trees, as illustrated in Figure 2.

The mutation operation modifies a GP tree by randomly selecting a mutation point in a tree, and then replacing it with a new, randomly-generated subtree, as illustrated in Figure 3. After crossover and mutation, the fitness values of the newly generated offspring are evaluated. The population is then re-ranked after the appending the offspring solutions and the two least-fit individuals deleted to return the population to its original size.

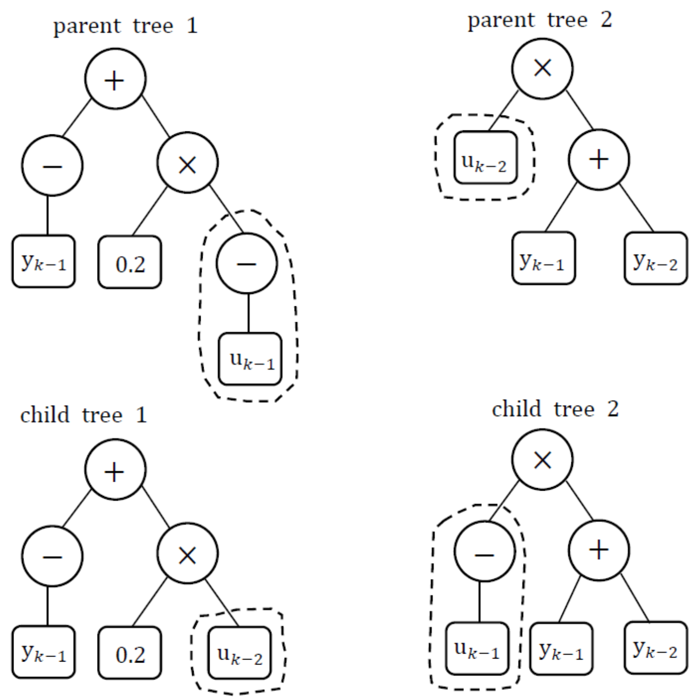


Figure 2: Example of subtree crossover.

- Step 4: Process termination

  The above procedures are iterated from step 2 to step 3 until user-specified termination conditions are met; here we have used a fixed number of iterations.

One of the fundamental problems of GP is bloat – the inexorable growth in tree size with no accompanying improvement in fitness. The use of multiobjective optimization in GP, however, can reduce the effects of bloat by providing a selective pressure that favours smaller models – so-called parsimony pressure. In order to generate compact and accurate models, we have used the twin fitness measures of tree size (number of tree nodes) and mean squared error (MSE)

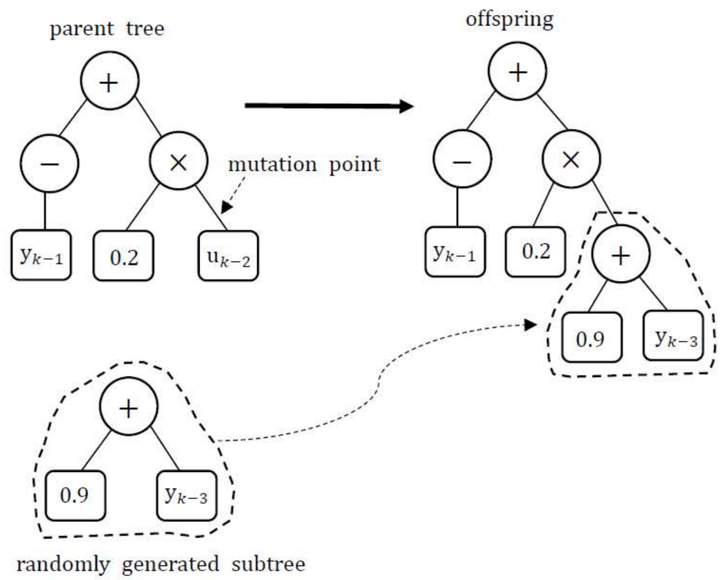


Figure 3: Example of subtree mutation.

over the dataset as two non-commensurable objectives. The Pareto dominance based ranking scheme [19] was used to rank the individuals in the population. A fitness vector $\mathbf{a} = (a_1, ..., a_p)$ is said to dominate $\mathbf{b} = (b_1, ..., b_p)$ if and only if $\mathbf{a}$ is partially less than $\mathbf{b}$, i.e., $\mathbf{a} \prec \mathbf{b}\ \forall i : a_i \leq b_i \wedge \exists i \in 1, ..., p : a_i < b_i$; in our case the fitness vectors are the 2-vectors with node count and MSE as elements. During the evolution process, trees with higher ranks have bigger probabilities of being selected as parent trees to produce child candidates, and at the end of the run the population comprises a set of individuals which trades-off compactness against goodness of fit (small MSE) to the training data. Typically, this final population spans the spectrum of small individuals with large MSE values (underfitted models) through to large models with small MSE values (overfitted models). We select a single, final model as the one which has the smallest MSE over a validation set independent to the training set.

The performance of GP on nonlinear dynamic system identification was first evaluated on a benchmark chemical engineering process, the Eaton-Rawlings reactor model in the following section. Having demonstrated satisfactory performance on this task, GP was applied to finding a dynamical model of the test building for MPC.

### *2.1. Identification of the Eaton-Rawlings Reactor*

There have been many reports in recent years exploiting the potential of GP for system identification, particularly in chemical engineering applications [20]. In order to assess the suitability of GP for nonlinear dynamic system identification, one-step-head (OSA) prediction of a well-known benchmark chemical process, the Eaton-Rawlings reactor model, was investigated. The Eaton-Rawlings model [21] describes a second-order reaction occurring in an isothermal continuous stirred-tank reactor (CSTR). The dynamics of this reactor are expressed by the first-order, nonlinear ordinary differential equation

$$\frac{dy}{dt} = -hy^2 - \frac{yu}{V} + \frac{du}{V} \tag{3}$$

where $y$ is the concentration in the CSTR, $h$ is the kinetic rate constant for the reaction, $V$ is the reactor volume, and $d$ is the inlet concentration of the reactant. The manipulated variable $u$ is the inlet flow rate.

If the manipulated input $u$ is assumed to change only at regular sampling instants $t_k$, an exact discretisation is easily derived from the continuous-time equation; see [21] for full details.

$$y(k) = \frac{[1 - \tau(k-1)\mu(k-1)]y(k-1) + 2d\tau(k-1)\mu(k-1)}{1 + \tau(k-1)[y(k-1) + \mu(k-1)]} \tag{4}$$

where

$$\tau(k-1) = \frac{\tanh\left[hT\sqrt{\mu^2(k-1) + 2d\mu(k-1)}\right]}{\sqrt{\mu^2(k-1) + 2d\mu(k-1)}} \tag{5}$$

$$\mu(k-1) = \frac{u(k-1)}{2hV} \tag{6}$$

In this experiment, $h$ was 1.50 litre/mole-hr, $V$ was 10.51 litre, $d$ was 3.5 mole/litre [21]. The inputs $u_k$ were a sequence of steps of uniformly-distributed random amplitudes ranging from 0.5 to 5.0 litres per hour with a switching probability of 1.0. This input sequence was used to perturb the reactor model (4).

To facilitate direct comparison with previous, conventional modelling approaches, we have followed the procedure in Pearson [21] and generated 100 statistically-independent training sequences, each of length $P = 200$. The reactor responses to these input sequences were calculated according to the discretisation formula (4).

The performance of a GP solution was ranked by two objectives: tree size and the MSE. The parameter settings for the GP evolution are described in Table 1; for the identification of the Eaton-Rawlings model, we used GP constants in the range $[0.0 \ldots 1.0]$.

In each GP experiment, a set of solutions was obtained after training from which the candidate with the smallest MSE over a validation dataset was finally selected as the best model for this run. Thus, after 100 independent training processes, the best GP model with the smallest validation MSE among the selected 100 trees was picked as the best overall solution. The best GP model selected had a validation MSE of 0.000121458.

We made quantitative comparison with the model – a nonlinear autoregressive moving average model with exogenous inputs (NARMAX) – that exhibited the best performance compared to other hand-tuned model structures and

Table 1: Evolutionary parameters used for the Eaton-Rawlings reactor system identification

| **Parameter** | **Value** |
|---|---|
| Population size | 100 |
| Evolution strategy | Steady state |
| Initialization method | Ramped half-and-half |
| Maximum tree depth in initialization | 6 |
| Maximum number of tree evaluations | 20000 |
| Function set | $+$, $-$, $\times$, Analytic quotient [22] |
| Terminal set | Input variables; constants in range $\{0.1, 0.2, \ldots, 0.9, 1.0\}$ or $\{0.1, 0.2, \ldots, 1.9, 2.0\}$ – see text. |
| Crossover frequency | 1.0 |
| Mutation frequency | 1.0 |
| Fitness measures | Tree size and MSE |
| Selection method | Pareto ranking |

lags studied by Pearson [21]; the same training datasets were used to train the NARMAX models by minimising the mean squared error metric (8) using the `NLopt` nonlinear optimization library[2]. The best NARMAX model [21] is given by:

$$y(k) = y_0 + \alpha\, y(k-1) + \beta\, u(k-1) + \gamma\, u(k-1)\, y(k-1) \tag{7}$$

where $y_0$, $\alpha$, $\beta$, and $\gamma$ are unknown parameters to be determined by minimizing the objective function, and $Q$ is the length of the training sequence:

$$J(y_0, \alpha, \beta, \gamma) = \sum_{k=1}^{Q-1} [\hat{y}(k+1) - y(k+1)]^2 \tag{8}$$

where $\hat{y}(k+1)$ is the one step ahead predicted value at time $k$, and the $y(k+1)$ is the measured value at time $k+1$.

The best validation MSE of the NARMAX models was 0.000120801, nearly equal to the value of obtained from the best GP tree. The residuals – the differences between the true and predicted values – of the best NARMAX and GP models over the corresponding validation sets are shown in Figure 4 from which it can be seen that the GP tree exhibits comparable model accuracy to the best NARMAX model. The residuals of the NARMAX model show a number of negative spikes lower than -0.02, with the absolute value of the biggest residual around 0.04. The GP model shows two significant positive spikes, but with most of the residuals lying in a small range around zero.

The encouraging approximation ability of the GP model on the benchmark Eaton-Rawlings problem suggests that GP is suitable for more general nonlinear

[2] `https://nlopt.readthedocs.io/en/latest/`

dynamic system identification problems. Particularly, GP does not require the functional form of the model to be pre-specified – rather, this evolves during training. This advantage makes GP a potential technique for identifying a wide range of real world, nonlinear dynamic systems for which the underlying physical principles are not known.

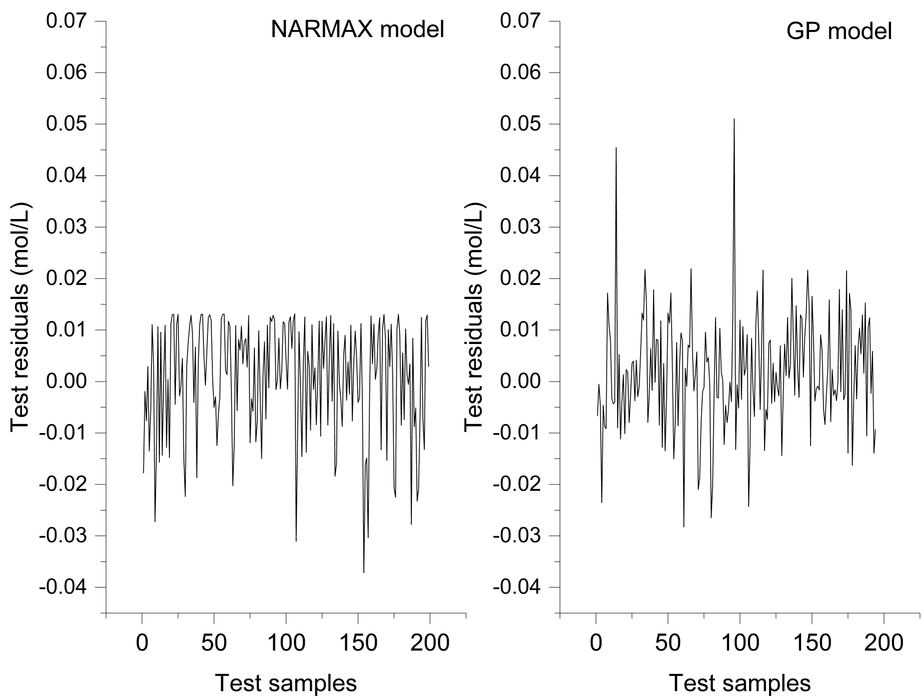


Figure 4: Test residual comparison between the NARMAX model and the best GP model

## 3. Building Control Methodology

In this section, we describe the procedures employed for the MPC of buildings using predictive models obtained through a GP-based system identification. Figure 5 depicts the components of the simulation system used in this work. We used an industry-grade simulator for building physics, described in Section 3.1, to simulate the responses of a test building. This simulator provides a standardised interface – the Functional Mockup Interface (FMI) [23] – that allows the interconnection of external software units – see Section 3.1. We used this facility for two separate tasks: first, for the open-loop collection of system identification (SID) data detailed in Section 3.3, and second for the simulation of the building under model-predictive control, as explained in Section 3.6. In Section 3.5, we describe how we employed GP using the collected SID data to obtain the required predictive models for MPC.

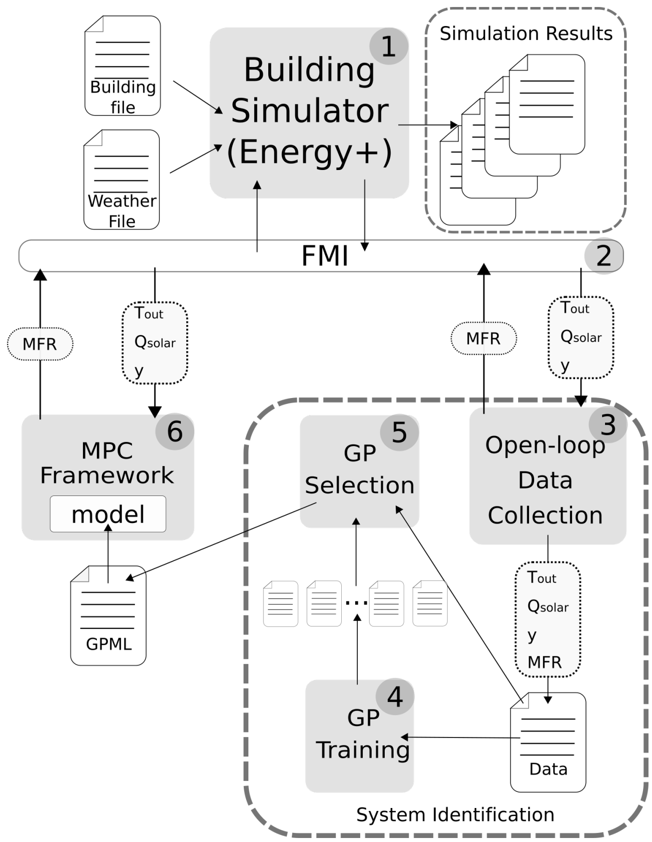


Figure 5: Overview of the process employed in this work for MPC. The mass flow rate (MFR) of hot water through the radiator, external temperature $T_{out}$, the sum of direct and diffuse solar radiation $Q_{solar}$ and zone temperature $y$ are communicated to/from the EnergyPlus simulator via functional mock-up interfaces (FMIs).

### *3.1. Building Simulator – EnergyPlus*

EnergyPlus is a building energy simulator used to model energy consumption based on dynamic heat transfer calculations [24]. The description of the building is provided to EnergyPlus as a text file – the input data file (IDF) – that follows a prescribed format. The file controls all aspects of the simulation from the building geometry and fabric to the building services and other simulation parameters, such as occupancy.

The (key) influence of the external weather is incorporated into the building simulation using a separate file containing weather data, a so-called *weather file*. This allows repeating the computations under different climatic conditions. In this work, we have used design weather files generated from UK meteorological data collected at a station located in Manchester, UK. Two different weather files, both containing one year's data, were used in this work: the training and validation datasets were extracted from the first weather file, while the second file was used to test the model for a whole unbroken year.

EnergyPlus supports Functional Mock-up Interfaces (FMIs) [23], a standardised interface for coupling software units for co-simulation. These software units can add a variety of functionalities to the simulation and are referred to as 'slaves'. The main simulator – EnergyPlus in our case – is referred to as the 'master'. In practice, the FMI defines a set of C language function prototypes that need to be implemented by the slave unit. The master will then call these functions at appropriate times during the simulation to perform various operations, such as send data, perform calculations, read data, etc. Here we make two distinct uses of the FMI functionality (see Figure 5): first, we inject an excitation sequence to perform open-loop system identification (described in Section 3.3). Referring to Figure 5, the mass flow rate (MFR) to the heating radiator is varied, and the external temperature ($T_{out}$), sum of the direct and diffuse solar radiation ($Q_{solar}$) and zone temperature ($y$) are logged at every time sampling interval. Second, having trained the GP models on the system identification data collected in the previous step, we used FMIs to control the building's heating during model predictive control experiments (described in Section 3.6). Again referring to Figure 5, here under MPC control, the current environmental conditions ($T_{out}$, $Q_{solar}$ and $y$) are acquired from the EnergyPlus simulation via FMIs, the optimal MFR control value calculated externally taking into account the heating schedule, and this optimal MFR applied at the next time step update.

### *3.2. Test Building Description*

For this initial report of implementing MPC using a learned dynamic model, we developed a single zone space with a conventional hydronic radiator supplied by a boiler producing water at a fixed output temperature of 67 °C. The heating system was sized using EnergyPlus which determined the maximum flow of hot water through the radiator to be 0.11 kg/s. The simulated test building is illustrated in Figure 6. The zone is a square room with dimensions of 10 m $\times$ 10 m and a height of 3 m. All four walls contain one double glazed window unit measuring 2 m $\times$ 2 m with a sill height of 0.5 m, and placed at the centre of the external walls. This design has a window-to-wall ratio of 13% with equal exposure to North, East, South and West directions. The single zone space has been set to be located in Manchester, UK, which has an oceanic climate (Köppen classification = Cfb) and classified as ASHRAE (American Society of Heating, Refrigeration and Air Conditioning Engineers) climate zone 5c. The construction sets and internal gains for this climate recommended by ASHRAE Standard 189.1 [25] were considered for this space to make sure a realistic set of inputs was defined in the building model for estimating the internally-generated heat as well as the heat loss from the façades.

We have used a setpoint temperature of 20 °C during occupied (working) hours of 9am to 5pm. Outside those hours, the schedule specified that the zone temperature was $\geq$ 6 °C to ensure frost protection. For the present set up, we have used this schedule for seven days a week to gather as much data about the the MPC operation as possible.

The airflow through the space consists of infiltration (0.00023 $m^3/s$ per $m^2$ of exterior surface) and a ventilation rate of 10 l/s per person in accordance with the Chartered Institute of Building Services Engineers (CIBSE) Guide A [26]. The occupancy density was 0.0565 persons/$m^2$, the default value for office buildings in EnergyPlus based on the ASHRAE 189.1 2009 standard. Heat gains from people, lighting and electrical equipment were also incorporated in a schedule.

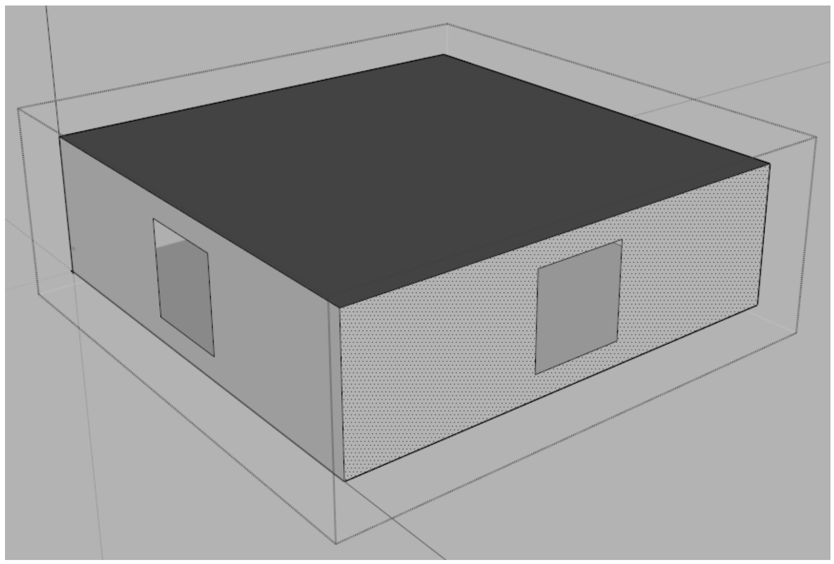

Figure 6: SketchUp representation of the simulated building

### *3.3. Open-loop System Identification*

The design of appropriate excitation signals for collecting identification data is the most crucial step in system identification as the gathered data are required to be informative enough to capture the underlying system dynamics under all relevant operating conditions while for practical reasons, the duration of the SID experiments should be as short as possible to minimise disruption. Mathematically, the amplitude of the excitation signal should cover the full range so as to maximise the power of the excitation signal and thus the signal-to-noise ratio; the spectrum of the input signal should excite all frequencies of interest.

For linear systems, *pseudo-random binary sequences* (PRBSs) [27] are commonly used for system identification. In a PRBS, the signal switches between two fixed amplitudes in such a way that that the autocorrelation function of the sequence approximates the properties of white noise, and hence excites all modes of the system. For nonlinear systems – such as that under consideration here – switching between two fixed amplitudes cannot capture the nonlinear behaviour [9], and so we have employed *amplitude modulated pseudo random binary sequences* (APRBSs) [9] in which the amplitude of a conventional PRBS is randomly varied, thereby probing the nonlinear characteristics of the system.

A PRBS sequence was generated using linear feedback shift registers where the length of the excitation sequence is controlled by a characteristic polynomial of some degree $n$, and where each polynomial coefficient was either 0 or 1. The maximum repetition period is given by $(n^2 - 1)$ (i.e. the maximum length of the sequence before it starts to repeat itself). The consecutive occurrence of the same bit is referred to as a plateau. We employed the polynomial $x^7 + x^6 + 1$, resulting in a sequence length of 127 bits with 64 plateaux, depicted in Figure 7(a). The interval between the minimum and maximum radiator flows (0.00 to 0.11 kg/s) was divided by the number of plateaux in the PRBS resulting in a set of different amplitude levels, which were randomly assigned to the PRBS's plateaux, thereby generating the APRBS [9]; an example sequence is shown in Figure 7(b). The process of randomly assigning amplitude levels to the PRBS plateaux was repeated to obtain a set of different excitation sequences. Note that each repetition of the process is likely to generate a completely different APRBS cycle, as seen in Figures 7(b-c).

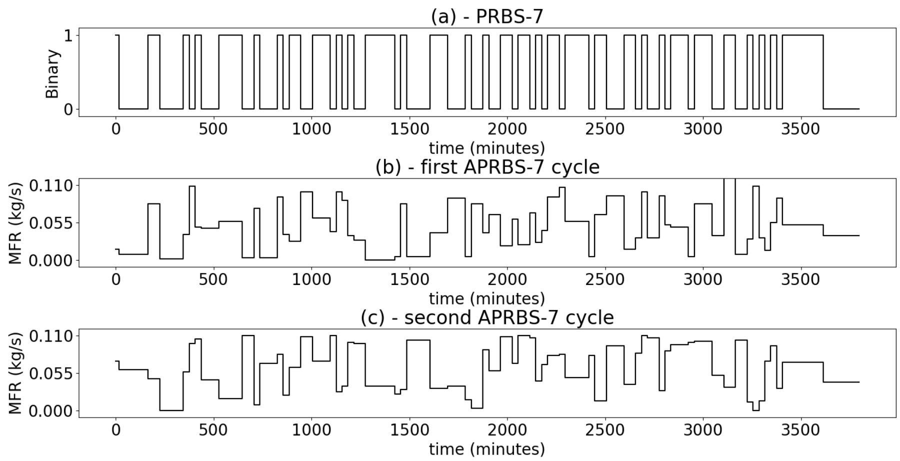


Figure 7: A PRBS-7 sequence (a) and two different examples of APRBS-7 cycles (b-c) generated over a minimum-to–maximum flow amplitude range of 0.0 to 0.11 kg/s.

In addition to the characteristic polynomial and the interval of the input sequence, an APRBS is specified by a minimum hold-time $T_h$, that is the duration of each bit; Nelles [9] suggests that the minimum hold-time should be the same as the dominant time constant of the process. In our case, the only input that can be excited is the mass flow rate through the radiator so we estimated the dominant time constant as approximately 30 minutes by applying a step excitation to the simulated zone. As a consequence, a single APRBS-7 cycle takes around 3,810 minutes (about 2.6 days), as depicted in Figure 7.

*3.4. Input selection*

In system identification, the selection of model inputs is key to model accuracy. Too many redundant or irrelevant input variables hampers the search while increasing the computational burden. Conversely, if input variables of significant influence are omitted, the model will have systematic errors and be more likely have poor prediction accuracy. The input and output variables used in this paper are listed in Table 2. Full details of the input selection criteria and experiments will be published elsewhere, but, in brief, we performed a sensitivity analysis over the set of weather variables with respect to predictive accuracy leaving us with $T_{out}$ and $Q_{solar}$ as the most influential; the mass flow rate (MFR) is, of course, the manipulated variable and $y$ the predicted zone temperature. All the variables were scaled so as to have the training values falling in the range 0 to 1. The scaling factors used are listed in Table 2.

Table 2: Variables used in the system identification model

| **Variable** | **Variable name** | **Type** | **Scaling factor** |
|---|---|---|---|
| $T_{out}$ | Outdoor Air Drybulb Temperature (°C) | Input | 21.0 |
| $Q_{solar}$ | Sum of Direct and Diffuse Solar Radiation (W/m$^2$) | Input | 839.8 |
| $MFR$ | System Node Mass Flow Rate (kg/s) | Input | 0.11 |
| $y$ | Zone Air Temperature (°C) | Output | 21.0 |

*3.5. Genetic Programming for Building Identification*

- Training & validation

  A dynamical predictive GP model was developed based on an EnergyPlus simulation model and the open-loop excitation data (see Section 3.3). Two different weather files were employed: One weather file, denoted `TRY`, comprising 365 days and 35,040 samples, was used to generate data for model training and validation (model selection). The other dataset, denoted `DSY` and of the same size, was used for estimating the model generalisation and prediction accuracy. A particular challenge with this MPC application is that weather conditions play a very important role in determining the internal temperatures of the building but they cannot be experimentally perturbed in the same way as the radiator MFR variable. We have thus used two weather files to allow an evaluation of performance over a complete year independent of the training/validation data.

  Since the (approximate) time constant of the simulated building is $\sim$30 minutes, the sampling interval for the MPC was set at half this figure, namely 15 minutes. The selected input variables (see Section 3.4) were sampled every 15 minutes for training, validation and testing of the GP models and the MPC process predicts temperatures on this interval.

Given a GP model $f$, at time $k$ the prediction of the zone temperature $\hat{y}_{(k+i)}$ at time $(k+i)$ is approximated from:

$$\hat{y}_{(k+i)} = f(\mathbf{u}_{k+i-1}, \mathbf{u}_{k+i-2}, ..., \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \\ \hat{y}_{(k+i-1)}, \hat{y}_{(k+i-2)}, \ldots, y_k, y_{k-1}, \ldots) \tag{9}$$

where $i$ ranges from 1 to $N$, the length of the prediction horizon, and $\mathbf{u}$ are the indexed sequence of exogenous input vectors. In this experiment, a $\mathbf{u}$ vector consisted of three variables: $T_{out}$, $Q_{solar}$, and $MFR$ – see Table 2. The zone temperature $y$ is the predicted variable.

In the training phase, all the input and output information is known since $\mathbf{u}$ consists of measured values determined by the APRBS excitation sequence (Section 3.3) and the known weather data; the zone temperatures up to and including the current time $k$ are also known. For multi-step ahead prediction, the required autoregressive values *later than* time $k$ use previously predicted values from a series of one-step ahead predictions. For example, at time $k$, $\hat{y}_{(k+1)} = f(\ldots, y_k, y_{k-1}, \ldots, y_{k-m})$. To predict two steps ahead, $\hat{y}_{(k+2)} = f(\ldots, \hat{y}_{(k+1)}, y_k, y_{k-1}, \ldots)$. Note the use of a predicted zone temperature $\hat{y}_{(k+1)}$ at time $(k+2)$ since when the model is used in its ultimate control application, the actual value $y_{(k+1)}$ will be unknown as it lies in the future – it therefore has to be estimated. Similarly, the prediction three steps ahead $\hat{y}_{(k+2)}$ uses both $\hat{y}_{(k+1)}$ and $\hat{y}_{(k+2)}$, and so on. Previously predicted values are used $\forall i \in [1 \ldots N]$, as necessary.

Two objectives are used to measure the performance of candidate models during evolution: tree size and MSE. The tree size indicates the complexity of a GP model, and provides parsimony selection pressure that favours simpler models during the evolutionary process. We also seek to minimise the MSE over the training dataset by:

$$MSE = \frac{\sum_{k=k'}^{P} \sum_{i=1}^{N} [\hat{y}_{(k+i)} - y_{(k+i)}]^2}{[P - \max(n, m)] \times N} \tag{10}$$

where $N$ is the length of the prediction horizon, and $P$ is the largest index on the training dataset used. Since we require the GP model to provide accurate predictions over the *whole* prediction horizon, minimising (10) provides a selective evolutionary pressure to achieve this. The values of upper limit on the outer summation and the normalising term in (10) requires some clarification: Suppose we have $Q$ records of available training data. To train a model that predicts $N$ steps ahead, the final $N$ records of the dataset can only be used for evaluating (10) – the index $k$ cannot exceed $(Q-N)$. Similarly, for an autoregressive model with $n$ lagged $\mathbf{u}$ values and $m$ lagged $y$ values, the first $\max(n, m)$ records of the training set are needed to calculate the very first prediction. Consequently, $k'$, the lower limit on the outer summation in (10), cannot be less that $[\max(n, m) + 1]$.

In summary, the MSE in (10) is calculated over $(Q-N)-[\max(n,m)-1]$ consecutive records. (Conventionally, the lower limit of this outer summation is taken as $k=1$ and any lagged inputs with (strictly) negative $k$ indices are taken as zero. We have not used this approach here as, in our experience, this sometimes produces anomalous transient predictions.)

The detailed GP parameter settings for building identification were identical to those shown in Table 1, except we have used constants in the range $[0.0 \ldots 2.0]$.

- Exporting the Selected GP Tree

  After model validation, the best GP model was selected for use in the EnergyPlus MPC framework. Rather than the cumbersome inconvenience of embedding the GP training within the MPC framework, we have exported the trained GP model using the Genetic Programming Markup Language (GPML) [28]. GPML is an XML-based standard for the interchange of genetic programming trees. The implementations of reading and writing GPML are simple and straightforward since a number of mature, open source XML libraries are available. A trained-and-validated GP tree can thus be directly embedded as a 'plug-in' component using GPML in larger systems, which provides both convenience and modularity.

### *3.6. MPC Test Framework*

Based on the model $f$ in (1) approximated with a GP as described in Sections 3.3 - 3.5, a control law could, in principle, be obtained as the inverse of $f$ – that is a mapping from desired states to the necessary inputs. However, as $f$ is dynamic and nonlinear, calculating its inverse is not a trivial task. The alternative is to perform an explicit optimisation at the current time $k$ of the set of values $\mathcal{U}_k = \{\mathbf{u}_k, \mathbf{u}_{(k+1)}, \ldots, \mathbf{u}_{(k+N-1)}\}$ using $f$, where $N$ is the prediction horizon, and adjusting $\mathcal{U}_k$ to yield the desired sequence of setpoint temperatures. Thus we obtain $\mathcal{U}_k$ from:

$$\mathcal{U}_k = \operatorname{argmin} \sum_{i=1}^{N} J(k+i) \tag{11}$$

where $N$ is the length of the prediction horizon. Since the (approximate) time constant of the building's response was determined to be 30 minutes, and the MPC sampling interval was 15 minutes, we adopted a value of $N=12$ giving a prediction horizon of 3 hours.

The function $J$ is defined as:

$$J(k+i) = \underbrace{(\Delta T_{k+i})^2}_{\text{temperature}} + \underbrace{\lambda_0 \left|\Delta U_{k+i}\right|}_{\text{control effort}} + \underbrace{\lambda_1 \mathbf{u}_{k+i}}_{\text{energy}} \tag{12}$$

where $\Delta T_{k+i} = \hat{y}_{(k+i)} - r_{(k+i)}$ is the difference between the predicted $\hat{y}_{(k+i)}$ and setpoint (i.e. desired) $r_{(k+i)}$ temperatures at time $k+i$, and $\Delta U_{k+i} = \mathbf{u}_{k+i} - \mathbf{u}_{k+i-1}$ is the control effort. The first term in (12) obviously penalises deviation from the desired setpoint temperature, while the second term seeks to minimise the extent of changes in the control variable; such a term is frequently included in an MPC setup to minimise wear on the system's actuators.

The third term in (12) seeks to minimise the sum of the input quantity, which in the present case is a proxy for input energy. We found it necessary to include this term in (12) to ensure that heating was turned off outside working hours when the constraint on the setpoint temperature was relaxed to simply being >6 °C to ensure frost protection. Unless this explicit energy minimisation term was included, (11) could be minimised by maintaining the zone temperature at 20 °C – obviously, >6 °C – but setting the sum of the magnitudes of the control efforts ($|\Delta U|$) to zero. That is, not turning off heating at the end of the working day thereby undesirably maintaining heating during the night.

Note that the second and third terms in (12) are weighted by the regularisation constants $\lambda_0$ and $\lambda_1$, which place differing relative penalties on each of the three factors in (12). The values of $\lambda_0$ and $\lambda_1$ were determined by grid search with the aim of maximising their values subject to acceptable temperature regulation. Predictions of zone temperature $\hat{y}_{(k+i)}$ are calculated using the GP model described in Section 3.5.

One notable point is that in the training process, the weather variables $T_{out}$ and $Q_{solar}$ are always assumed known, and for which we have used measured values. During the validation and test phases, however, the practical use of the model means that future values of $T_{out}$ and $Q_{solar}$ are unknown. Consequently, we have used *persistent predictions* for future (as yet unknown) weather variables. Namely, the value of the weather variable at time $k$ is assumed to persist unchanged for the whole of the current prediction horizon. Persistent weather prediction is known to be reasonably accurate over the short-term [29, 30] while having the advantage of being simple to implement.

The minimisation in (11) was performed using an implementation of the nonlinear, derivative-free optimiser COBYLA [31] together with the Multi-Level Single-Linkage (MLSL) procedure [32, 33] from the `NLopt` nonlinear optimization library[3]. Although MPC has the advantage of being able to straightforwardly incorporate constraints on the solution, in the present work we have used no such constraints.

The detailed MPC optimisation parameter settings for building testing are shown in the Table 3.

## 4. Results

In Section 4.1 we describe the results of training the GP predictive model using the open-loop system identification data produced using the procedure setout in Section 3.3. Section 4.2 presents the control results from using the trained GP predictive models in an MPC controller to regulate the zone temperature of the building model described in Section 3.2.

[3] `https://nlopt.readthedocs.io/en/latest/`

Table 3: MPC optimisation parameters used in this work.

| **Category** | **Parameter** | **Value** |
|---|---|---|
| MPC | Prediction horizon $N$ | 12 steps |
| MLSL (global) | Population size | 4 |
| | Maximum number of evaluations | 131,072 |
| | Stop when objective value less than | 0.001 |
| COBYLA (local) | Maximum number of evaluations | 8,192 |
| | Stop when objective value less than | 0.001 |
| Fitness function | $\lambda_0$ | 10.0 |
| | $\lambda_1$ | 10.0 |

### *4.1. GP Model Training*

We conducted thirty GP training runs, each with an independent initial population, using the open-loop system identification data generated with the procedures described in Section 3.3 to obtain 30 individual models with the best validation set MSE per run. Among these, the model with the smallest validation MSE overall was finally selected as the best model. January's data (2,880 records) were used as the training dataset and February's data as validation dataset. Typically, the CPU runtime of each independent experiment takes ∼400 s (on a given computer with 3.30 GHz CPU).

The residuals (i.e. errors) for each of the $i$-step ahead predictions ($i \in [1 \dots N]$) measured over the 12-month independent test set for the best GP model are shown in Figure 8. Here we are assessing the important performance measure of whether the model produces sufficiently accurate predictions over the whole (test) year, and does not, for example, erroneously demand heating of the building in the middle of summer.

From Figure 8, residuals of the one step ahead prediction fall into range from -1.25 to 1.75 °C; even in the summer months (June to August) , most residuals are below 1.5 °C. Unsurprisingly, as the predictions extend further into the future, the envelope of residuals expands although only slightly. In addition, a distinct seasonal 'bow' becomes more noticeable with increasing $i$ value.

Our choice of the metric in (10) was deliberately designed to give equal weight to the prediction errors over the whole prediction horizon. In 12-step-ahead prediction, residuals in January, February, March, November and December range from -2 to 2.5 °C. Residuals in the months such as May, June, July and August reach their highest values of 4 °C and with an average lower bound of 0 °C error. After increasing during summer, the sizes of the residuals decrease later in the year. In summary, the GP model provides promising predicted temperatures: the magnitudes of the residuals expand with increasing prediction step size $i$ with the biggest residual less than 4 °C. The best model presented in GPML form can be found at:

`https://figshare.com/articles/gpTreeConstantWFInVall_xml/7398797`

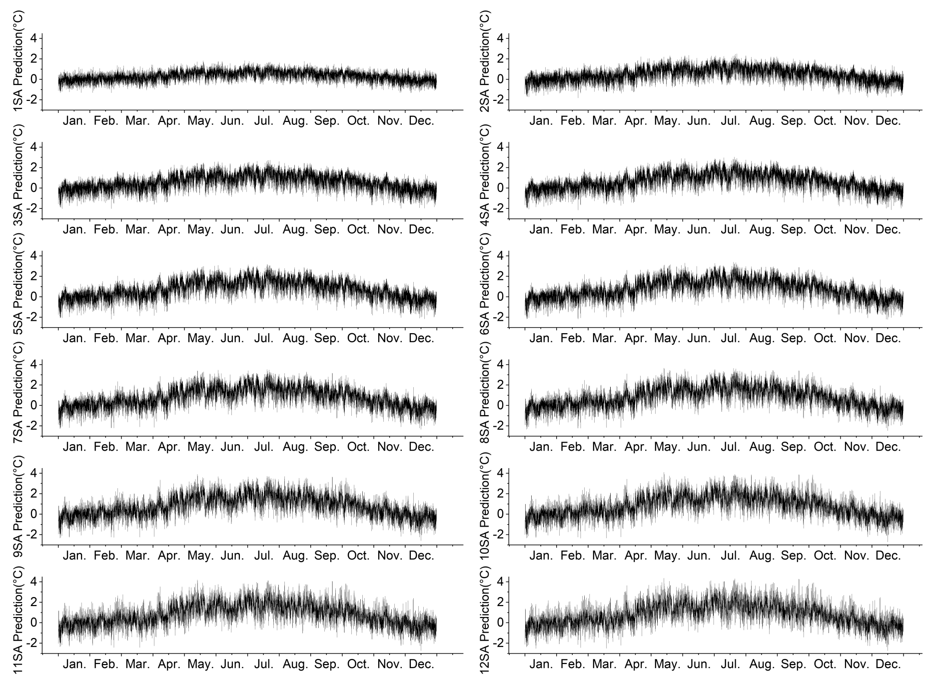


Figure 8: Residuals (predicted - actual zone temperatures in °C) of the selected GP model over the test dataset for different future predictions. Each plot shows the residuals for a given number of steps ahead – for example, "OSA" = 1-step head, "2SA"= 2-steps ahead, etc. The units of the ordinate axes are Celsius.

*4.2. MPC Performance*

Figure 9 shows a typical result of controlling the test building during the month of February (from the 32$^{\text{nd}}$ to 59$^{\text{th}}$ day of the test year) using the MPC framework from Section 3. The test weather data are independent of the data used to train/validate the predictive model. The reference value $r_{(k+i)}$ was set to match 20 °C during working hours (9:00 to 17:00). To achieve frost protection out-of-hours, the reference value was set at $\geq$ 6 °C. That is, we only apply a temperature penalty out-of-hours if the temperature falls *below* the frost-protection reference value. Formally, $\Delta T_{(k+i)}$ out-of-hours is defined as:

$$\Delta T_{(k+i)} = \begin{cases} \hat{y}_{(k+i)} - r_{(k+i)} & \text{if } \hat{y}_{(k+i)} < r_{(k+i)}. \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

As described in Section 3.6, we have predicted future (unknown) weather values using persistence, that is, assuming the variable has the same value over the whole of the prediction horizon as it does at the start.

The upper plot in Figure 9 shows the zone temperatures, and also a ±1 °C range during working hours (dotted lines). The lower plot shows the MFR control variable (hot water flow through the radiator).

From Figure 9, it is clear that zone temperatures are mostly being maintained within the ±1 °C band during working hours. A noteworthy feature of the MFR control variable is that it varies fairly smoothly over the day, and is being reduced to small values towards the ends of the working days; we infer that MPC is exploiting the energy stored in the building's fabric to maintain the setpoint temperature up to the end of the working day, thereby avoiding direct heating, if possible, which may lead to energy savings.

Considering the thermal performance of MPC, it successfully maintained the zone temperature within the ±1 °C range for 83.3% and ±2 °C for 95.2% of the (working) time. The 1 °C band is generally deemed comfortable, with a small proportion (less than a quarter) of occupants feeling mildly uncomfortable at the extremes of the 2 °C band [26]. Although there will be some discomfort for the short periods outside these bands that fall at either end of the working day, these are rare.

## 5. Discussion and Future Work

As stated above, scope of this paper is to present what we believe to be the first report of buildings MPC using a predictive model learned data acquired from the building. Our aim here has been to present and document the methodology we have used although a great deal of work remains to be done both in terms of 'fine tuning' this, and in extending it. One area that does needs to be explicitly discussed is comparator methods. Since the GP-controller regulates the zone temperature generally within a highly satisfactory ± 1 °C, and exhibits no instances of overheating and therefore wasted energy, the only reasonable basis for comparison is the ease/cost of generation of the predictive model – see [2]. As pointed out in Section 1, we are aware of no comparable technique

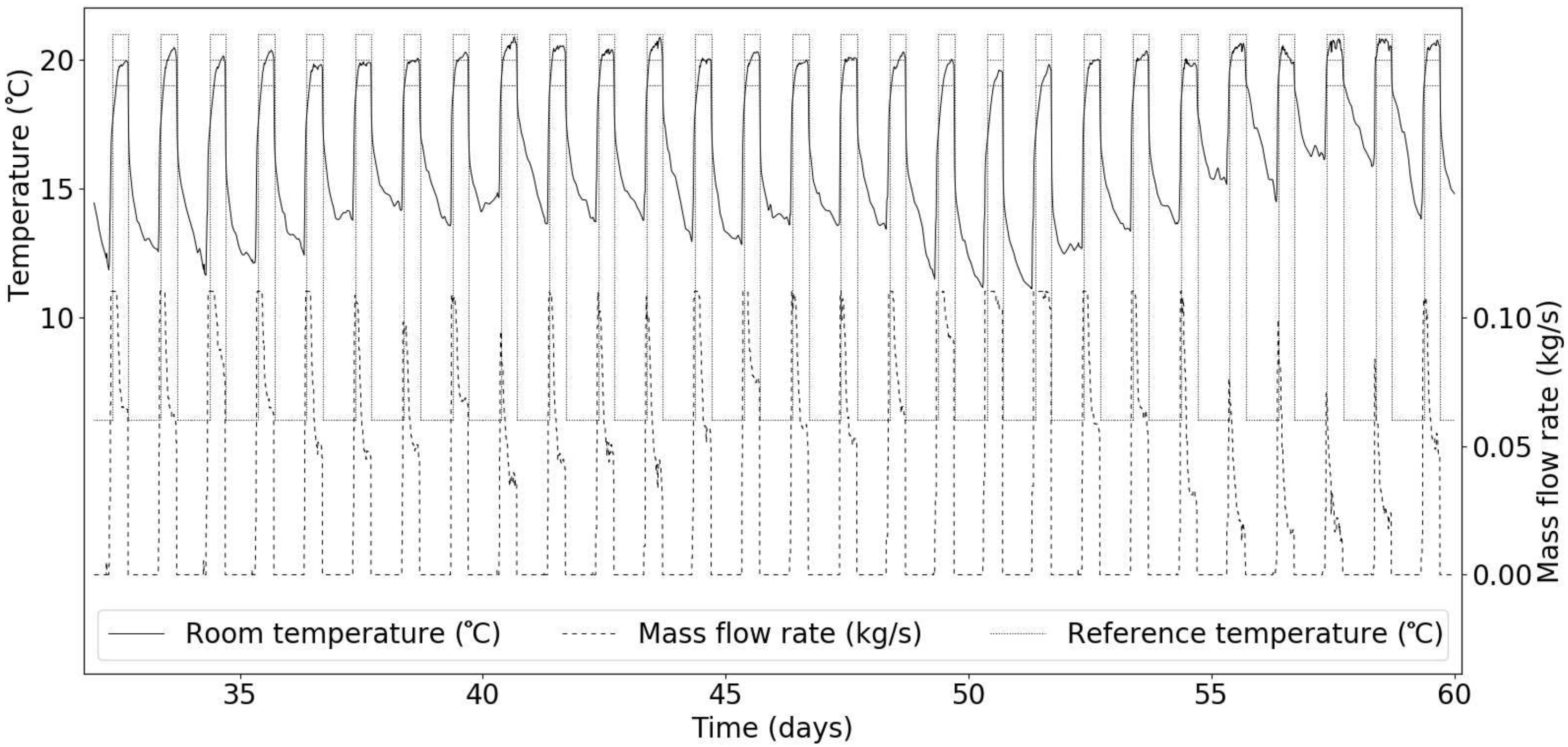


Figure 9: Typical results of MPC controlling a single-zone room for the (test) month of February. The upper plot is the zone temperature, and the lower plot the mass flow rate (MFR) controlled variable. The rectangular upper plot represents the temperature schedule. The dotted lines represent the temperature schedule together with a ± 1 °C tolerance band shown during occupied hours (9am to 5pm).

with which to make direct comparison, and in which both the structure of the predictive model *and* the composition of the lag sets are simultaneously identified. In all other applicable techniques of which we are aware, the search over possible model forms would need to be embedded within a search over all possible lag sets (feature selection). The GP approach used here thus represents a robust and simple automated workflow for practical application of MPC with few 'tuning' parameters; further GP tends to be fairly robust to its tuning parameters.

We can identify a number of interwoven topics that will be the subject of future work, and will be published elsewhere.

Although we have reported only one instance of a predictive model trained on 30 days open-loop excitation data, optimisation of the training process clearly needs to be explored systematically. Although the test residuals shown in Figure 8 are clearly adequate to produce acceptable control, as evidenced by Figure 9, the model residuals exhibit a noticeable seasonal effect – an upward 'bowing' in the middle of the plots. In the summer months, the actual temperatures are systematically somewhat higher than those predicted by the model, but in the present application with only heating of the building, this turns out not to make a great difference since heating is not necessary in the summer months. For a more complicated building, however, that includes cooling as well as heating, these seasonal effects may produce unacceptable conditions although for a combined heating/cooling system, the SID procedure would obviously also need to include excitation of both heating and cooling. Consequently, improving the model quality is therefore clearly an area for future work, and a number of factors need to be examined.

The duration of the open-loop excitation experiment: Although we report only data for 30 days of system identification, it seems possible to train adequate models with shorter data sequences than this. The trade-off between the length of the SID experiment and model quality needs to be explored. Naively, one would expect model quality to improve with longer SID sequences (= more training data), but extending the system identification experiment has implications for both the amount of energy used during SID as well as the practicality of conducting the experiment.

In terms of practical application, we have established that acceptable predictive models can be trained on data accumulated over 30 days. Ideally, this SID timescale needs to be shortened to be more compatible with current building project schedules, and to be integrated with existing commissioning/testing regimes for the heating system. One potential area of future work is to use a much shorter SID duration and form the predictive model using consensus prediction over multiple models [34] as has been widely used in meteorology, econometrics, etc.

For the sake of simplicity, we have assumed persistent weather predictions. That is, we assume the weather inputs have the same values over the entirety of the prediction horizon as they do at the start of the prediction horizon. Persistence is known to be acceptable in the short term, but it is possible that more elaborate methods [35] may give improved results, especially for larger buildings that mandate longer prediction horizons. In particular, short-term weather predictions are widely available online and an avenue of future research is to examine the effect of replacing the assumption of persistence with more sophisticated predictions.

The length of the prediction horizon has been chosen to be around six times the characteristic time constant of the building (although the step response of the building is clearly not a first-order characteristic). The trade-offs inherent in selecting a prediction horizon are well-known in the MPC literature [1]: a longer horizon allows a more relaxed planning timeframe and tends to avoid overly aggressive control moves, while producing more uncertain predictions due to the length of time into the future over which they are being made. Shorter prediction horizons face the converse issues. Systematic examination of setting the prediction horizon in the context of buildings is therefore warranted.

The MPC framework we have reported uses the rather conventional objective of penalising deviation from a setpoint, in this case zone temperature, together with an appropriately weighted term to minimise control effort (a proxy for actuator wear). In addition, we have included a term designed to minimise energy consumption over the prediction horizon, this latter term proving necessary for proper operation out of working hours. Clearly the regularisation constants (here denoted $\lambda_0$ and $\lambda_1$) will have an influence on the control although quite how significant these will be also needs to be investigated.

Such a regularisation framework has been commonly used in previously published reports on buildings MPC [2], and appears to have been adopted straightforwardly from its widespread use in chemical engineering and related industrial applications of MPC. In process engineering, of course, it is frequently important to maintain some optimal process temperature to maximise product yield, etc. Maintaining zone temperatures within a small band, however, is generally unnecessary in buildings. Indeed relaxing the temperature constraints on a zone can have beneficial energy-saving advantages. More generally, tuning regularisation constants is known to be problematic and time-consuming, and to a large extent, a conventional regularisation framework militates against our overall objective of automating implementation of MPC in buildings in order to make it economically viable. Consequently, alternative minimisation objective functions may well be more appropriate in a building setting. For example, minimising energy usage (over the prediction horizon) subject to the explicit optimisation constraints of maintaining zone temperatures within, say, a $\pm 2$ °C band.

The system identification experiments reported here involve open-loop excitation of the building. It is well-known that open-loop excitation can drive the system's states to extremes since there is no feedback control to prevent this. Apart from potentially consuming significant amounts of energy, performing an open-loop system identification experiment on an occupied building would probably be unacceptable. Indeed, it is highly likely that the occupants would take atypical actions, such as opening windows and doors, to make the internal conditions more acceptable to themselves, thereby undermining the validity of the system identification data. Rockett and Hathway [2] have already suggested closed-loop system identification as a way of addressing the shortcomings of open-loop identification for periodic recalibration of the controller: closed-loop identification [36] maintains the system under control using an initial predictive model while typically applying small perturbations to the desired setpoint from which an improved, control-capable model can be derived. This process of closed-loop re-estimation can, of course be repeated periodically as-and-when the building's characteristics change. Closed-loop recalibration of MPC's predictive model is clearly an area for future work where issues such as: the necessary excitation sequence and the sensitivity of the re-estimation procedure need to be explored.

The work presented in the paper has demonstrated successful MPC implementation over a single-zone building. Clearly extension to multiple-zone buildings is an obvious area of future work, where the necessary SID procedures will need to accommodate multiple, interacting thermal zones.

Finally, although the work presented here has been done in simulation – as is very common in the initial steps of a control project – the ultimate proof of the methodology is to demonstrate its use on a real building. This too is currently work in progress.

## 6. Conclusions

In this paper, we have reported the first use of genetic programming to obtain predictive models for the model predictive control (MPC) of internal building temperatures. Currently, the large-scale adoption of MPC in buildings is rendered uneconomic by the time and cost involved in the design and tuning of predictive models by expert control engineers. We have shown that GP is able to automate this process using an open-loop excitation experiment. The resulting MPC simulation is able to maintain the internal temperature of a single-zone test building to within ±1 °C of the desired setpoint most of the time; we further infer that MPC is able to effectively exploit the heat stored in the building's fabric towards the end of a working day rather than applying direct heating. The results in this paper have significant implications for enabling the wide-scale deployment of MPC in non-domestic buildings, and for the potential reduction in $CO_2$ emissions by improving the efficiency of building operation.

## 7. Acknowledgements

We gratefully acknowledge the financial support of the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N022351/1.

[1] E. F. Camacho, C. Bordons, Model Predictive Control, 2nd Edition, Springer, London, 2004.

[2] P. Rockett, E. A. Hathway, Model-predictive control for non-domestic buildings: Critical review and prospects, Building Research & Information 45 (5) (2017) 556–571. doi:10.1080/09613218.2016.1139885.

[3] G. P. Henze, Editorial – Model predictive control for buildings: A quantum leap?, Journal of Building Performance Simulation 6 (3) (2013) 157–158, special Issue on Model Predictive Control for Buildings. doi:10.1080/19401493.2013.778519.

[4] M. A. Hussain, Review of the applications of neural networks in chemical process control - simulation and online implementation, Artificial Intelligence in Engineering 13 (1) (1999) 55–68. doi:10.1016/S0954-1810(98)00011-9.

[5] M. Wetter, Modelica library for building heating, ventilation and air-conditioning systems, in: F. Casella (Ed.), 7th International Modelica Conference, Como, Italy, 2009.

[6] I. Hazyuk, C. Ghiaus, D. Penhouet, Optimal temperature control of intermittently heated buildings using Model Predictive Control: Part I – Building modeling, Building and Environment 51 (2012) 379–387. doi:10.1016/j.buildenv.2011.11.009.

[7] D. Sturzenegger, D. Gyalistras, M. Morari, R. S. Smith, Semi-automated modular modeling of buildings for model predictive control, in: 4th ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys '12), Toronto, Canada, 2012, pp. 99–106. doi:10.1145/2422531.2422550.

[8] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, A. Juditsky, Nonlinear black-box modeling in system identification: A unified overview, Automatica 31 (12) (1995) 1691–1724. doi:10.1016/0005-1098(95)00120-8.

[9] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer, Berlin, 2001.

[10] K. O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation 10 (2) (2002) 99–127. doi:10.1162/106365602320169811.

[11] B. Grosman, D. R. Lewin, Automated nonlinear model predictive control using genetic programming, Computers and Chemical Engineering 26 (4-5) (2002) 631–640. doi:10.1016/S0098-1354(01)00780-3.

[12] Q. Feng, H. Lian, J. Zhu, Model predictive control of nonlinear dynamical systems based on genetic programming, in: 36th Chinese Control Conference (CCC), Dalin, China, 2017, pp. 4540–4545. doi:10.23919/ChiCC.2017.8028072.

[13] K. Rodríguez-Vázquez, C. M. Fonseca, P. J. Fleming, Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming, IEEE Transactions on Systems, Man & Cybernetics - Part A: Systems & Humans 34 (4) (2004) 531–545. doi:10.1109/tsmca.2004.826299.

[14] M. P. Hinchliffe, M. J. Willis, Dynamic systems modelling using genetic programming, Computers and Chemical Engineering 27 (12) (2003) 1841–1854. doi:10.1016/j.compchemeng.2003.06.001.

[15] G. Casella, R. L. Berger, Statistical Inference, Duxbury, Pacific Grove, CA, 2002.

[16] T. Dou, P. Rockett, Comparison of semantic-based local search methods for multiobjective genetic programming, Genetic Programming and Evolvable Machines 19 (4) (2018) 535–563. doi:10.1007/s10710-018-9325-4.

[17] R. Poli, W. B. Langdon, N. F. McPhee, A Field Guide to Genetic Programming, Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.
URL `http://dces.essex.ac.uk/staff/rpoli/gp-field-guide/A\_Field\_Guide\_to\_Genetic\_Programming.pdf`

[18] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.

[19] C. Fonseca, P. J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: A unified formulation, IEEE Transactions on Systems, Man & Cybernetics - Part A: Systems & Humans 28 (1) (1998) 26–37. doi:10.1109/3468.650319.

[20] R. Vyas, P. Goel, S. S. Tambe, Genetic programming applications in chemical sciences and engineering, in: A. H. Gandomi, A. H. Alavi, C. Ryan (Eds.), Handbook of Genetic Programming Applications, Springer International Publishing, Cham, 2015, pp. 99–140.

[21] R. Pearson, Discrete-time Dynamic Models, Oxford University Press, 1999.

[22] J. Ni, R. H. Drieberg, P. I. Rockett, The use of an analytic quotient operator in genetic programming, IEEE T. Evolut. Comput. 17 (1) (2013) 146–152. doi:10.1109/TEVC.2012.2195319.

[23] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, H. Elmqvist, A. Junghanns, J. Mauß, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. P. an S. Wolf, C. Clauß, The Functional Mockup Interface for tool independent exchange of simulation models, in: $8^{\text{th}}$ International Modelica Conference, Dresden, Germany, 2011, pp. 105–114. doi:10.3384/ecp11063105.

[24] D. B. Crawley, C. O. Pedersen, L. K. Lawrie, F. C. Winkelmann, EnergyPlus: Energy simulation program, ASHRAE Journal 42 (4) (2000) 49–56.

[25] ASHRAE, Standard for the design of high-performance green buildings except low-rise residential buildings,, Tech. Rep. ASHRAE 189.1, American Society of Heating, Refrigerating and Air-conditioning Engineers, Atlanta GA (2009).

[26] CIBSE, CIBSE Guide A: Environmental Design, Tech. rep., The Chartered Institution of Building Services Engineers, London, UK (2015).

[27] T. Söderström, P. Stoica, System Identification, Prentice Hall, New York, 1989.

[28] T. Dou, Y. Kaszubowski Lopes, P. Rockett, E. A. Hathway, E. Saber, GPML: An XML-based standard for the interchange of genetic programming trees, "Genetic Programming and Evolvable Machinesdoi:10.1007/s10710-019-09370-4.

[29] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. M. de Pison, F. Antonanzas-Torres, Review of photovoltaic power forecasting, Solar Energy 136 (2016) 78–111. doi:10.1016/j.solener.2016.06.069.

[30] D. Corne, A. Reynolds, S. Galloway, E. Owens, A. Peacock, Short term wind speed forecasting with evolved neural networks, in: 15$^{th}$ Annual Conference Companion on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 2013, pp. 1521–1528. doi:10.1145/2464576.2482731.

[31] M. J. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in: S. Gomez, J.-P. Hennart (Eds.), Advances in Optimization and Numerical Analysis, Vol. 275, Springer Netherlands, 1994, pp. 51–67. doi:10.1007/978-94-015-8330-5_4.

[32] A. H. G. Rinnooy Kan, G. T. Timmer, Stochastic global optimization methods part I: Clustering methods, Mathematical Programming 39 (1) (1987) 27–56. doi:10.1007/BF02592070.

[33] A. H. G. Rinnooy Kan, G. T. Timmer, Stochastic global optimization methods part II: Multi level methods, Mathematical Programming 39 (1) (1987) 57–78. doi:10.1007/BF02592071.

[34] R. T. Clemen, Combining forecasts: A review and annotated bibliography, International Journal of Forecasting 5 (4) (1989) 559– 583. doi:10.1016/0169-2070(89)90012-5.

[35] A. R. Florita, G. P. Henze, Comparison of short-term weather forecasting models for model predictive control, HVAC & R Research 15 (5) (2009) 835–853. doi:10.1080/10789669.2009.10390868.

[36] M. Gevers, Identification for control: From the early achievements to the revival of experiment design, European Journal of Control 11 (4-5) (2005) 335–352. doi:10.3166/ejc.11.335-352.