A Development and Testing Framework for Simulation-Based Supervisory Control With Application to Optimal Zone Temperature Ramping Demand Response Using a Modified Genetic Algorithm

Brian Coffey

A Thesis

in

The Department

of

Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Building Engineering) at
Concordia University
Montreal, Quebec, Canada

June 2008

Your file Votre référence
ISBN: 978-0-494-42488-9
Our file Notre référence
ISBN: 978-0-494-42488-9

ABSTRACT

A Development and Testing Framework for Simulation-Based Supervisory Control With Application to Optimal Zone Temperature Ramping Demand Response Using a Modified Genetic Algorithm

Brian Coffey

There is a growing demand for integrated control strategies for building systems with numerous responsive elements, such as solar shading devices, thermal storage and hybrid ventilation systems. Simulation-based supervisory control is a promising way of approaching this challenge. The essential idea is to use detailed building simulation real-time within a supervisory control system to test possible set-point configurations at each controller time-step before choosing the best configuration to use in the building. This dissertation presents a flexible software framework for simulation-based supervisory control, along with a modified genetic algorithm developed for use within it, and applies it to a case study of demand response by zone temperature ramping in an office space. Rule-based and simulation-based control variants are studied by using a two-model configuration (one acting as the 'real' building, the other being used within the simulation-based control framework). For the case study, simulation-based control was found to perform only slightly better than a logarithmic rule-based approach under ideal conditions, and worse under conditions of model or prediction inaccuracies. The results are of use in guiding further research, and the case study itself has been a good test of the software framework, which can be further developed and should be useful in future simulation-based control research.

ACKNOWLEDGEMENTS

The author acknowledges the funding and support of the Innovations & Solutions Directorate at Public Works and Government Services Canada. The author would also like to personally acknowledge and thank Dr Fariborz Haghighat (Concordia University), Dr Edward Morofsky (Public Works and Government Services Canada) and Ed Kutrowski (Public Works and Government Services Canada) for their advisory efforts.

CONTENTS

FIGURES

TABLES

Chapter 1

Introduction

With increasing concern about climate change and energy security, there is a growing interest in the development of technologies for very low energy commercial buildings. Many important organizations have stated goals pertaining to the mass adoption of zero-energy commercial buildings (generally conceived as very low energy buildings with onsite renewable meeting the remaining load) within the next two decades, including the US Department of Energy (DOE), the American Institute of Architects (AIA) and the American Society of Heating and Air-Conditioning Engineers (ASHRAE), and substantial research and planning is being directed in this area (see CBI, 2008). Passive technologies, such as increased insulation and fixed solar shading, are likely to account for part of the efficiency improvements. But active technologies, such as low-energy HVAC and variable-transmissivity window systems, are also expected to play a significant role in most low energy commercial buildings. System integration and optimal control of these active technologies is an important and growing area of research.

Buildings can also play a significant role in demand response, which is currently of significant concern, and will likely be of greater importance with more intermittent renewable electricity production being added to the grid. Increased flexibility in building systems and their control, for example through the active use of thermal storage, can be used to shift or shed electrical load when necessary. The overlap of demand-response control objectives and efficiency objectives makes the task of controls development for

future buildings more challenging, but luckily many of the strategies for improved efficiency controls can aid in improved demand response, and vice versa.

One such strategy for integrated control is that of simulation-based supervisory control. This strategy is also often referred to as model-based control or model-predictive control. (It is referred to herein without the 'predictive' reference in the name because even though it often must explicitly consider a future horizon, this need not be the case, as shown by some examples in the literature review in the next chapter.) The driving idea behind this strategy is to use a building simulation model within the building's actual control system, automatically testing at each control time-step various possible set-point configurations with simulations before choosing the best one for use in the building. This approach allows for the use of optimization algorithms to determine the best configuration, rather than relying on rule-based approaches that pre-specify set-points based on schedules or in response to particular conditions.

1.1 General Research Objectives

There is a modest amount of previous research in simulation-based supervisory control for buildings applications, but this body of research is growing quickly and is expected to grow further in the upcoming years. Previous studies have been application-specific, and researchers have had to devise the conceptual framework and custom-build software to carry out their case studies. The underlying motive of the research presented in this dissertation is to facilitate future research in this area by developing a standard approach and software for simulation-based supervisory control that can be easily applied to many different cases.

The general research objectives are thus as follows:

- Develop standard software (built around the existing GenOpt software) for simulation-based supervisory control for buildings
- Develop control optimization algorithms for use within this framework
- Test the software with a detailed case study

The case study is the focal point for this dissertation, in part because it helps make the general objectives more tangible, and in part because the case study itself does provide some interesting results.

1.2 Case Study Description

The case study considered is for optimal demand response using zone temperature set-point ramping for an office space in Ottawa. The office space considered is area 8B1 of Place-du-Portage, which is the home of the Innovations & Solutions Directorate of Public Works and Government Services Canada. This office space has been outfitted with a highly-instrumented personal environmental control (PEC) system that gives occupants control over their lighting and over a jet-diffuser that provides conditioned air to their cubicle. The substantial data availability makes it amenable to the creation and calibration of a detailed thermal model, which was done in TRNSYS. For the case study, this model was used both as the model within the simulation-based control strategy, and also to act as the 'actual' building for the tests. This virtual testing arrangement allows for tests of various strategies and configurations under identical conditions, which is essential for the studies described here. (It does not, however, dig into all of the logistics of using

simulation-based control within an existing control system – notes about how this can be configured to operate within the existing system are left to the discussion in Chapter 7.)

In the case study, the personal air-conditioning controls are over-ridden in favour of a demand response action, for one particularly hot afternoon in July. The goal of the demand response action is to decrease cooling demand between 1:00pm and 4:00pm, with the action allowed to begin 30 minutes prior, and the zone temperature set-point constrained to remain between 22°C and 26°C. Figures 1.1 and 1.2 show the base case with no action. (Variants of these graphs will be used throughout the dissertation. In each case, WS93 refers to workstation #93, whose set-point and temperature is used in these graphs as a proxy for the average temperature in the office space, and whose diffuser flow-rate is used as a proxy for the cooling demand, as discussed in Chapter 4.) The vertical bars in Figure 1.2 are of particular interest – they show the average electrical demand for each 15-minute period during the trim. The goal of the demand response action is to decrease the maximum of these bars as much as possible.

Figure 1.1: No trim

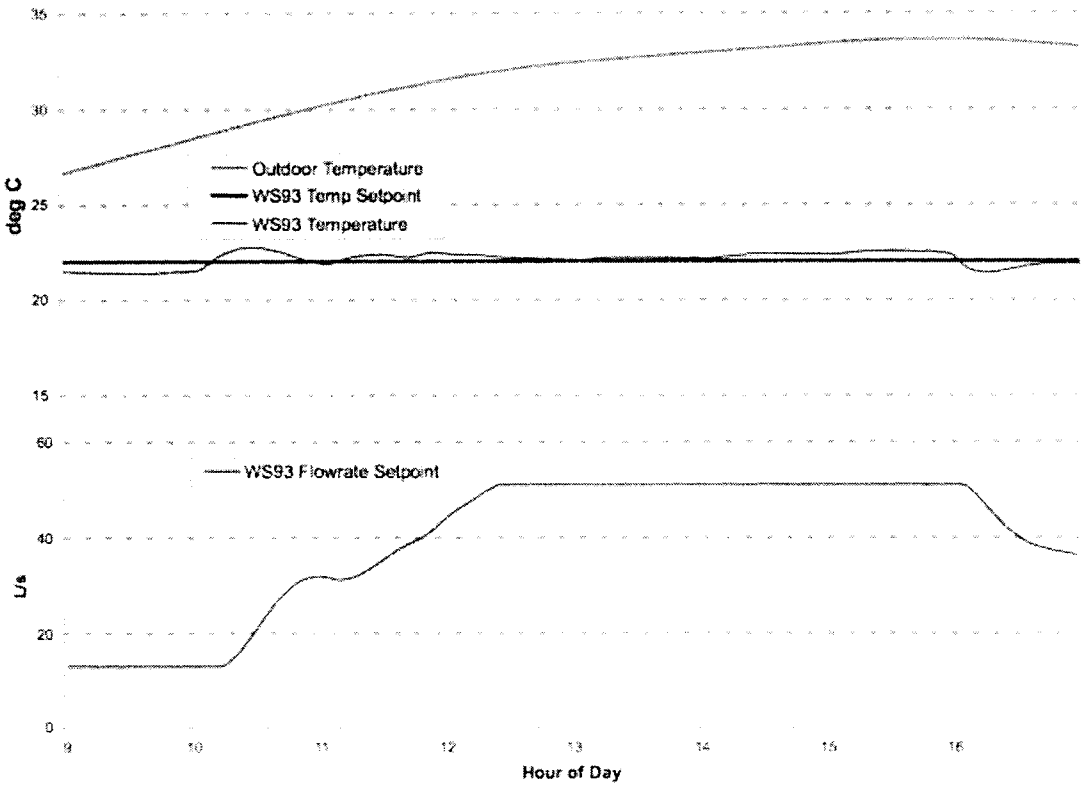
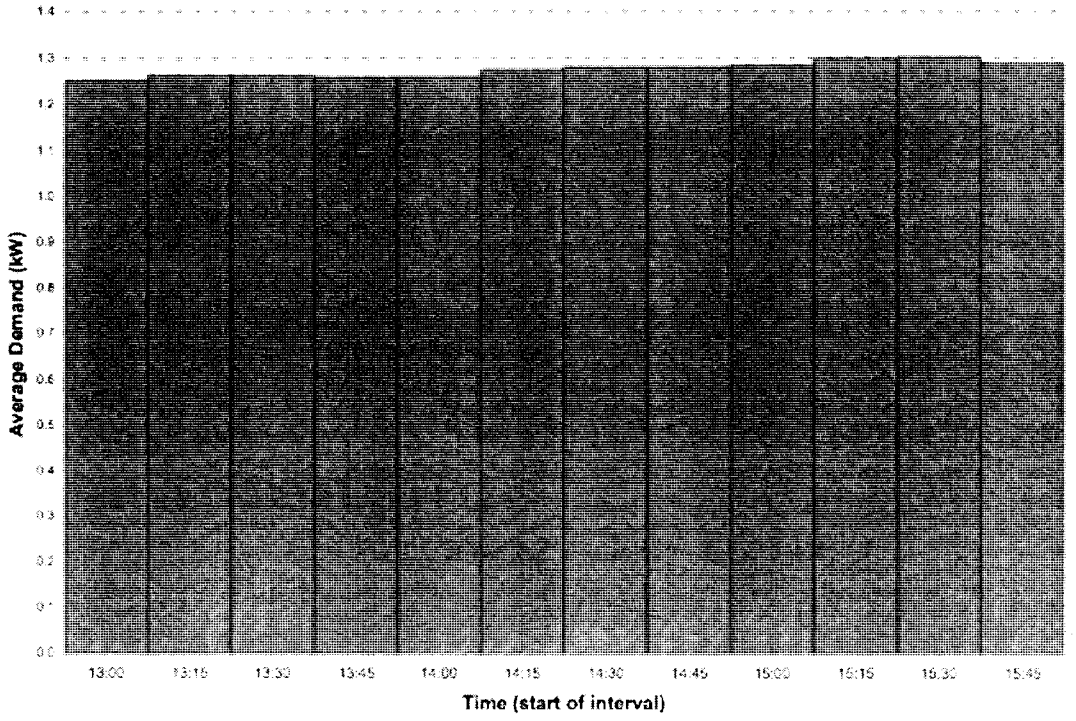


Figure 1.2: Demand curve with no trim



1.3 Case Study Objectives

The research questions asked in this case study are of two types. Firstly, we ask whether simulation-based supervisory control is appropriate in this particular case, based on how its performance compares with the performance of simpler rule-based approaches. And secondly, we ask how the performance of simulation-based control in this case varies with changes in the algorithms used, and how it varies with changes in the accuracy of the model within the controller and the predictions fed to it. As such, the demand response performance is compared under the following control strategies:

- rule-based strategies
    - 'jump trim' (stepping to 26°C and remaining there)
    - 'linear trim' (linearly ramping from 22°C to 26°C)
    - 'logarithmic trim' (ramping along a logarithmic trajectory)
- simulation-based control, with a modified genetic algorithm
    - with perfect predictions and model calibration
        - with the modified genetic algorithm as specified
        - with various aspects of the algorithm removed
    - with inaccuracies in the model calibration
    - with inaccuracies in the predictions

1.4 Organization of Dissertation

A literature review for both simulation-based control in general and for demand response with zone temperature ramping is presented in Chapter 2. The overall methodology, including the modified genetic algorithm used in the case study, is discussed in Chapter 3, with software details relegated to Appendix 1. The simulation model used in the case study is presented in Chapter 4, again with some details relegated

to Appendices 2 and 3. Chapters 5 and 6 provide results with rule-based approaches and simulation-based approaches. The results are discussed in Chapter 7, along with some discussion of how these results and the case study process can be used to inform further research. Conclusions and recommendations are presented in Chapter 8.

Chapter 2

Literature Review

Since simulation-based control optimization is a special case of simulation-based optimization in general, this literature review begins with a brief overview of the state of the art of simulation-based optimization in buildings, including a brief discussion of the GenOpt software and of genetic algorithms, both of which are used in the research presented in this dissertation. Previous studies in simulation-based supervisory control are then analyzed. The review ends with previous studies of control for demand response with zone temperature ramping.

2.1 Simulation-Based Optimization for Buildings – State of the Art

In building optimization problems, a wide variety of objectives may be considered, such as first cost minimization, comfort maximization or energy minimization. In most cases, the real objective function can only be approximated, often through the use of building simulation. Various different optimization algorithms may be used with simulation-based objective function calculations, and there are many application examples in the building engineering literature. Thus far, the vast majority of building simulation and optimization studies have focused on design optimization problems, rather than problems related to the operations phase of buildings, but there is a growing interest in such applications. But it is worth noting that both building simulation and the optimization approaches developed for it have grown primarily out of building design

applications, and the algorithms used for design optimization are not necessarily appropriate for control optimization.

2.1.1 GenOpt

In an effort to simplify and standardize the process of optimization with building simulation, M Wetter (while a PhD student working at LBNL) developed GenOpt, which was released in 2001. It is a Java program and is freely downloadable (Wetter, 2004a). GenOpt essentially acts as an interface between any building simulation program and any optimization algorithm. It can use any text-based simulation program (which is the case for DOE2, EnergyPlus and TRNSYS), and it has a standard library of optimization algorithms, which includes generalized pattern search algorithms, particle-swarm algorithms and algorithms for parametric studies, but currently does not include genetic algorithms. This library is structured such that it can be extended by the user. Although intended primarily for use in design optimization, GenOpt has also been used by some researchers for automated model calibration (which is discussed in Appendix 3).

The goal of software development in the current research is to facilitate simulation-based supervisory control research through the creation of a standardized platform for such studies, similar to the way that GenOpt is a standardized platform for design optimization studies. And in the hopes of building on existing tools and existing user-bases, rather than starting from scratch, GenOpt has been incorporated into the simulation-based supervisory control software framework outlined in this dissertation. (Or, thinking of it in the other direction, the simulation-based control software has been

developed as an extension or wrapper for GenOpt.) An overview of this integration is provided in the methodology chapter (Chapter 3) and details are included in Appendix 1.

2.1.2 Genetic Algorithms in Previous Building Optimization Research

Genetic algorithms (GAs) are inspired by biological evolution, using concepts of mating, mutation and natural selection to impel successive populations of individuals (with each individual representing a point in the search space) towards an optimization of the objective function. An initial population of points is chosen, usually at random from throughout the search space, and as the population evolves from one generation to the next new points emerge, with the evolutionary processes favouring the emergence of points that are ever closer to optimal. As such, genetic algorithms are a method of directing attention to more promising areas of a search space. Their path through the search space is stochastic (there is some randomness in the mutation and mating processes through which new points are chosen), so they can sometimes be less efficient as optimizers than other possible algorithms (depending on the problem under consideration and the detailed configuration of the algorithm), but their stochastic search path also makes them more able to avoid being trapped in local minima.

Genetic algorithms have been used fairly extensively in previous research in building science, for design optimization problems. Good examples include Wetter and Wright (2003), Wetter and Wright (2004), Wang et al (2005), Wright and Alajmi (2005), and Charron and Athienitis (2006). But their use in control optimization has been limited – only one such application of a genetic algorithm has been found in the literature, as discussed below.

2.2 Simulation-Based Supervisory Control of Building Systems

Although the idea of simulation-based supervisory control of buildings was noted at least as early as 1988 (Kelly), because of its computational requirements it has not received much research attention until the past decade, with most of the limited research precedents coming from the past five years. Previous work in this field has focused on the optimization of particular systems, with little research on the general methodology, and there has been little crossover in the techniques used for the various systems. The main developments have occurred in the fields of daylighting control, thermal storage system control and for some HVAC control problems.

We first overview the existing research in these three fields, and then analyze the optimization approaches used in the various studies, with some notes about ways that they might be improved. We then briefly consider related research in integrated building control with neural networks, and model-predictive control in other industries.

2.2.1 Active Solar Shading

The daylighting research (for the control of blinds and lights) has been primarily by Mahdahvi and colleagues. Mahdahvi was one of the first to provide a succinct description of the simulation-based control approach, with his 2001 paper entitled "Simulation-based control of building systems operation", where he notes that "In order to realize a simulation-based building systems control strategy, a conventional building automation system must be supplemented with a multi-aspect virtual model of the building that runs parallel to the building's actual operation". (Mahdhavi, 2001) Their experimental work has focused on the control of automated shading devices (Mahdavi, 2001; Mahdavi,

2005; Mahdavi et al., 2005b), using lighting simulation tools such as LUMINA. But they have also considered control for natural ventilation (Mahdavi and Pröglhöf, 2005a).

These studies provide examples of simulation-based control without predictions over a future horizon. The focus of the work has been on occupant comfort rather than energy consumption, so they have had to consider thermal capacitance or energy costs associated with state changes. The purpose of using simulation in these cases is to capture the complexities of a snapshot of the system, rather than to capture the system's dynamics.

2.2.2 Thermal Storage

Buildings-research into simulation-based control for energy storage has been dominated by G Henze and colleagues (Henze et al., 2005, Henze and Krarti, 2005). Since the optimal control of storage systems must consider what the future conditions will be, their studies have included the consideration of predictions over a future horizon. One of their most significant experiments involved the control of a building with active ice storage and passive thermal storage in the building mass, using a 24-hour future horizon, and a 1 hour time-step. TRNSYS was used for the building model and Matlab was used to control the optimization. It was an interesting test of the approach, and they noted some of the expected technical glitches (such as sensor malfunctions and problems with information passing). They found that aside from these technical problems, the approach was successful, except for a few points in time, where the optimization algorithm was stuck in a local minimum, and so actually performed worse than the default strategy would have.

They have also analyzed various aspects of simulation-based control that are peripheral to the central concept, but which are very important to its application. In particular, they have studied the importance of model accuracy (Henze et al. 2004), worked on the development of a method for automated model calibration to ensure continued accuracy over time (Liu and Henze, 2005), and worked on the development of a hybrid control system that attempts to incorporate the aspect of continual updating (found in the learning algorithm approach to control) with the simulation-based control approach (Liu and Henze, 2006). Their approaches to continual updating and automated calibration have helped to inspire some of the work discussed here in Appendix 3.

2.2.3 HVAC

The research in simulation-based HVAC control has been somewhat more dispersed, with a number of researchers making contributions to the field. The most pertinent studies of simulation-based HVAC system control have been of two types: supervisory control of an entire system, determining optimal set-point configurations for each time-step; and in determining the optimal start time for heating or cooling, to minimize energy consumption while ensuring occupant comfort when they enter the building in the morning. Most often, the models used are highly simplified, so the computation time associated with the optimization is not usually a concern.

Good examples of simulation-based supervisory control of entire HVAC systems are by Wang and Jin (2000) and Nassif et al. (2005). Both of these studies look at simulation-based control of VAV systems, and they take a very similar approach. At each time-step, the supervisory controller must determine the supply air temperature set-point, the supply

duct static pressure, the chilled water supply temperature, the outdoor air ventilation rate and the temperature set-points for each zone. The supervisory controller uses a simplified model of the HVAC system, and determines the set-points at each time-step by using a genetic algorithm. (The model was simplified to the point that, in the study by Nassif et al., the controller was able to run the model 50,000 times in less than 3 minutes.)

The problem of determining the optimal start time for heating has been considered by a number of researchers. Clarke et al. (2001) studied this particular problem, but in the process provided a very good overview of simulation-based control and its potential in general, brought together researchers from Honeywell Controls Systems Ltd and the University of Strathclyde, and worked out some of the interfacing concerns between BEMS systems and simulation tools (ESP-r in particular). A modification of the optimal start problem was considered by Kummert et al. (2005a and 2005b), who looked at the optimal control of passive solar buildings with night setback, in an attempt to minimize the energy consumption while also minimizing occupant discomfort due to morning undercooling and afternoon overheating. A forecasting horizon was used, along with a simplified model (a linear state-space representation) and the optimization was done by quadratic programming. Three control strategies were compared in a test facility: a Conventional Controller; a Reference Controller (PID); and an Optimal Controller (model-based predictive controller). Weather predictions were obtained by email and used by the optimal controller.

Another noteworthy research precedent in optimal HVAC control using simulation is the PhD thesis of Flake (1998). Flake considered the optimal supervisory control of a chilled water plant, using a detailed TRNSYS model of the plant under consideration.

This work was not, however, conceived as a real-time control approach, but rather as a simulated approach that could be used to inform rule-based strategies for actual implementation, so he did not have to contend with the computation time challenge associated with global HVAC control using a detailed simulation model.

2.2.4 Optimization Approaches in Previous Research

A number of different optimization approaches were used in the previous research discussed above. One key element of many of the approaches (particularly for HVAC control) was to use a simplified model of the system (Kummert et al., 2005; Nassif et al., 2005; Wang and Jin, 2000). This allows for easier optimization: in some cases, the model might be simplified to the point that analytical optimization techniques can be used; in other cases, using a simplified model just decreases the computation time required for a simulation run in a generate-and-test algorithm.

However, a more detailed model is often desired. The HVAC control strategies considered often avoided detail by considering very simplified load estimations rather than a detailed transient simulation of the building physics. This works fairly well for the HVAC problems considered, but it is not suitable for studies of passive storage in the building mass (Henze et al., 2005, Henze and Krarti, 2005), or for other more complex systems. The research in this dissertation is directed towards the use of detailed building physics models.

When using a detailed model, if the number of possible system states is small and the simulations can be run quickly, it may be possible to test all of the possible configurations. This, however, is rarely the case. In some cases, it may be possible to

provide a rule that chooses a reasonable subset of the possibilities to test, and then to test all of the possibilities in this subset (as done by Mahdavi et al., 2005b). But in general scheme, we cannot expect to constrain the search space enough to use enumeration, so an optimization algorithm must be used. Aside from Henze et al. (2005) and Henze and Krarti (2005), which provides an example of a detailed model and a quasi-Newton gradient-based optimization, these previous studies have not considered the use of standard optimization algorithms with detailed building simulation models. (Flake, 1998, used simulated annealing optimization and a detailed simulation model, but as noted earlier, he was not dealing with the challenge of time-constrained optimization inherent in the real-time application of simulation-based control.)

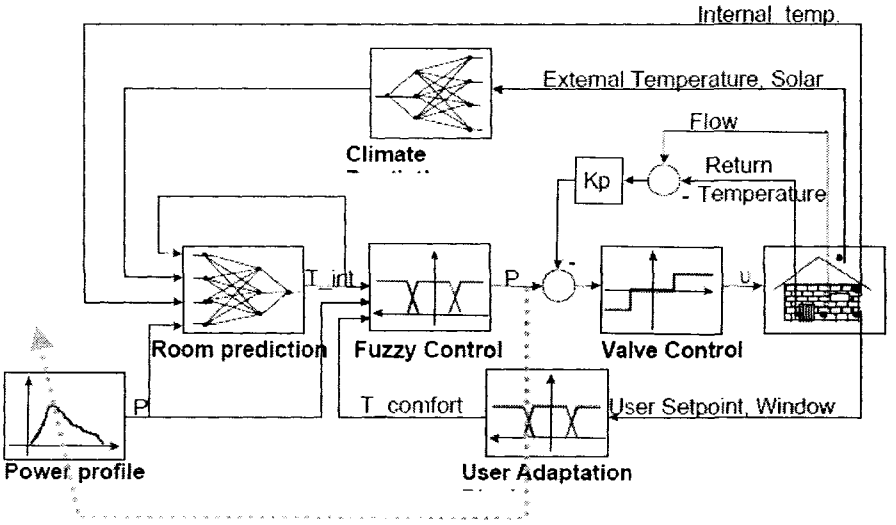
As discussed briefly in the next chapter and used extensively throughout the research in this dissertation, an important aspect of simulation-based control problems (particularly those that include prediction horizons) is that the optimization problem faced at any given time-step is very similar to the optimization problem faced at the previous time-step. Using previous time-step solutions to inform the algorithm at the current time-step has proven very useful in this dissertation research, but it was not taken advantage of in the previous studies, with the exception of the research described in Henze et al (2005) and Henze and Krarti (2005), where it was briefly noted that this was used to reduce the execution time for the optimization.

2.2.5 Related Research: Integrated Control with Neural Networks

Some advanced non-simulation-based approaches to supervisory building control use complex schemes that allow the control system to improve over time. One such approach

is quite similar to the simulation-based approach, and thus deserves note here. It uses artificial neural networks (ANNs) to estimate loads under different conditions, rather than the more complex physics-based approach used in simulation-based control. An illustrative example of this ANN approach is the EDIFICIO project carried out by a number of European research agencies and headed by a group from EPFL, as shown in Figure 2.1.

Figure 2.1: EDIFICIO (Priolo et al., 2001)



Clarke et al. (2001) outlined the differences between these learning-algorithm approaches and the simulation-based approach, and discusses the advantages and disadvantages of each. In general, learning-algorithm based controllers are easier to install and operate, and do not have the same concerns about model accuracy or computation time that the simulation-based control approach has. But unlike model-based controllers, they require an in-situ learning period before use, they do not deal well with physical or occupant-use changes in the building, and they tend not to perform well under weather conditions they have not seen before. And since they lack any building physics

in their internal calculations, they cannot be used to explain why particular conditions or configurations produce particular effects, so they are of less use in building diagnostics or retrofit than are the models used in the simulation-based approach.

But the ability to improve performance over time, as embodied by these learning-algorithm approaches, is a very useful feature that should be incorporated into software and methodologies for the simulation-based approach – this will be touched on briefly in Chapters 3 and 7, and Appendices 1 and 3. And the use of ANNs to estimate loads includes an inherent loss in accuracy relative to a more detailed building-physics approach, but the vast difference in computation time still makes such use of ANNs appealing – Chapter 7 discusses ways of incorporating them into various parts of the simulation-based control approach.

2.2.6 Simulation-Based Control in Other Industries

Although still relatively uncommon in building engineering, model-predictive control has seen more extensive research and application in other fields, particularly in chemical, electrical and mechanical engineering (Morari and Lee, 1999, and Mayne et al, 2000, for seminal surveys, and Borrelli et al, 2006). A detailed study of model-predictive control (as it is commonly referred to in other fields, where the models are generally used to capture system dynamics) in these other industries was considered beyond the scope of this current research – a detailed study would require both more time and a stronger background in these other fields than was available for this current research. But future research should consider lessons from other industries that may be applied to the development of model-predictive control for buildings.

2.3 Summary: Simulation-Based Supervisory Control

There have been some previous studies of simulation-based supervisory control for applications in solar shading optimization, thermal storage optimization and HVAC set-point optimization. However, these studies have seen numerous application-specific methodological differences, and there does not seem to have been any work in trying to develop generally applicable methods or tools for simulation-based supervisory control.

There are lessons to be learned both from what has and what has not been considered in previous approaches: the prevalent use of very simplified models and Henze's difficulties with local minima are both indicative of the computational challenge associated with simulation-based supervisory control; Mahdavi's approach of pre-selecting a sub-set of points to consider suggests that imposing extra constraints on the search space may be an important strategy to consider in general; and the use of previous time-step results may be a way of significantly improving algorithm performance. And the ability to learn over time, as embodied in the neural network approaches, is a useful feature that simulation-based control strategies should try to include. As discussed in the next chapter, the software developed has attempted to incorporate some of these lessons from previous research, while taking advantage of the state of the art in building simulation-based optimization in general.

2.4 Demand response with temperature ramping

A number of previous studies (see, for example Xu et al., 2004, Xu 2006) have shown that demand response with zone temperature reset can be a very effective demand trimming strategy, producing substantial savings in cooling demand over the trim period

without sacrificing occupant comfort. Most of these previous studies have considered very simple control strategies (usually just resetting from the lower part of the comfort range to the upper part – what we refer to in this case study as a 'jump trim'), focusing on the demand response potential with strategies that could easily be mass-deployed.

However, research by K Lee and J Braun (2004, 2006a, 2006b and 2007) has looked more closely at the optimal control of this set-point modification to minimize the peak demand over the trimming period. Their work has been along two related tracks: (1) the development of an inverse model of a small commercial building, and using the model to determine optimal set-point trajectories (Lee and Braun, 2004, 2007); and (2) the development and evaluation of simple control rules that can approximate optimal control (Lee and Braun, 2006a, 2006b).

In the first track, they use the fact that the optimal set-point trajectory produces a flat demand over the trimming period. They use the inverse model to calculate, at any given point in time, what the set-point must be to produce a particular demand level. They then iterate over the demand level to find the minimum level possible within the problem constraints, and the collection of set-points associated with that level is thus the optimal set-point trajectory. This is a clever approach to the problem. The implementation is only slightly different than the model-based control approach considered in this dissertation, in that it does not march along making control decisions each time-step in real-time, but rather determines the full trajectory before the trim. But it could easily be made to march along and re-do the optimization at each time-step as new information became available (as noted in Lee and Braun, 2007). The use of the demand-threshold fact by backing out the set-point values for a specific demand level was possible because they were using an

inverse model – it would not be possible with most detailed simulation tools (Chapter 7 includes a brief discussion of possible improvements in building simulation that may help in aspects of simulation-based control such as this). There are various trade-offs between the approach used in this dissertation (detailed model with a less-efficient and less-effective optimization strategy) and the approach of Lee and Braun (simpler model with a better optimization strategy). Fully evaluating the relative merits of the two approaches could constitute a good follow-up study to the research in this dissertation.

The second track of Lee and Braun's research in this area, that of developing and evaluating near-optimal rule-based strategies for demand response with zone temperature ramping, is what is used most in this dissertation. In particular, they have determined that a very effective strategy is to use a logarithmic set-point trajectory – the equation and graphs are provided and discussed in detail in Chapter 5, where their effectiveness with the case study is considered.

In general, the problem of optimizing demand response with temperature ramping has been well covered by Lee and Braun, and their simplified rule-based approaches are very effective. Having such well-charted territory is useful for an early case study of a methodology and software that is still being further developed.

Chapter 3

Methodology

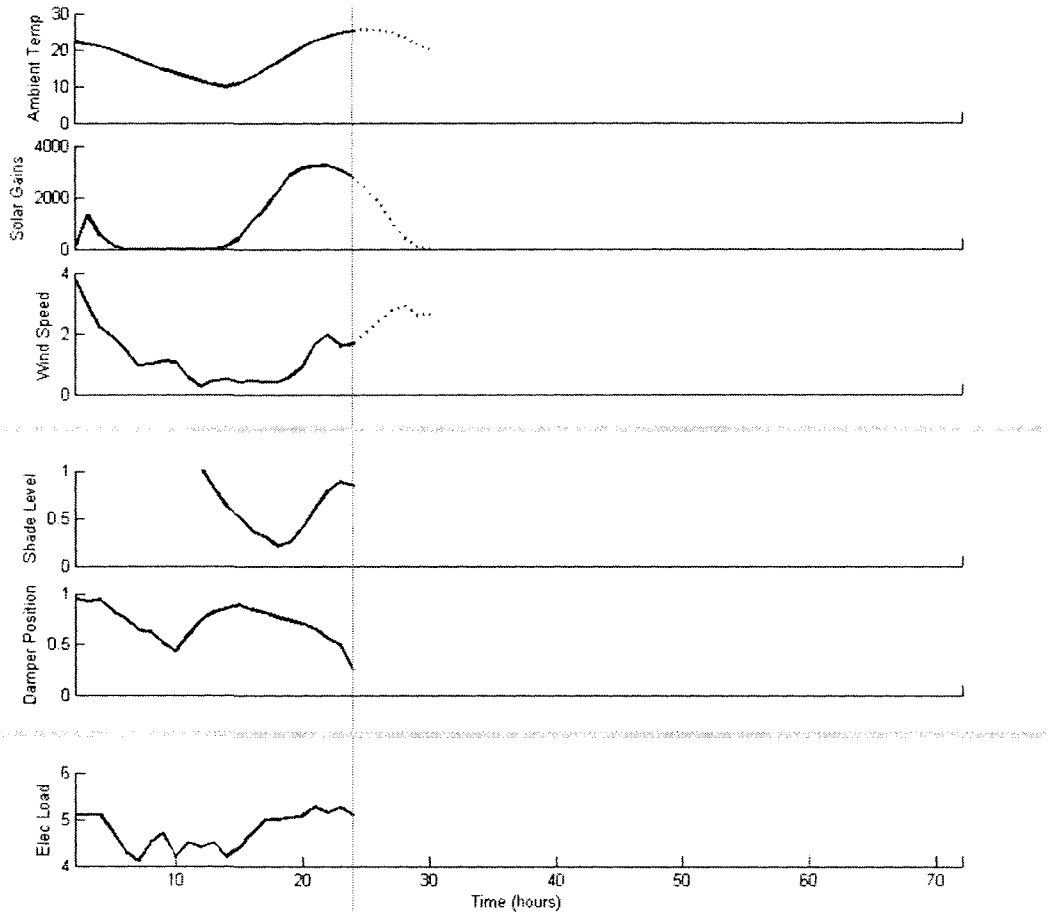
Although the basic idea behind simulation-based supervisory control is fairly simple, it can get quite complicated as one begins to look closer at the details. This chapter first attempts to clarify the essential concepts involved in the simulation-based approach. It then describes a software framework intended to clarify and simplify the methodology and facilitate further research. The requirements for such a framework are first noted, drawing on the essential concepts and the lessons from the previous chapter. An overview of the software framework is then provided, with interested readers referred to Appendix 1 for more details of its use. We then discuss the virtual testing environment, where two copies of the building simulation model are used: one within the simulation-based supervisory controller; and the other slowed down to act as the 'actual building'. The modified genetic algorithm developed for use within this framework is then described.

3.1 Mathematical Problem Definition for Simulation-Based Optimal Control

An example of a supervisory control scenario is shown in Figure 3.1. The top three graphs represent the external conditions data (e.g. ambient temperature, solar gain, windspeed) that can be used by the supervisory controller. The dotted lines represent predicted future conditions. The supervisory controller must choose values for the control variables (e.g. damper position), in the hopes of minimizing the objective function (e.g. electrical load). In Figure 3.1, the control variables graphs are in the center, and the objective function graph is at the bottom. Over time, the vertical line moves to the right,

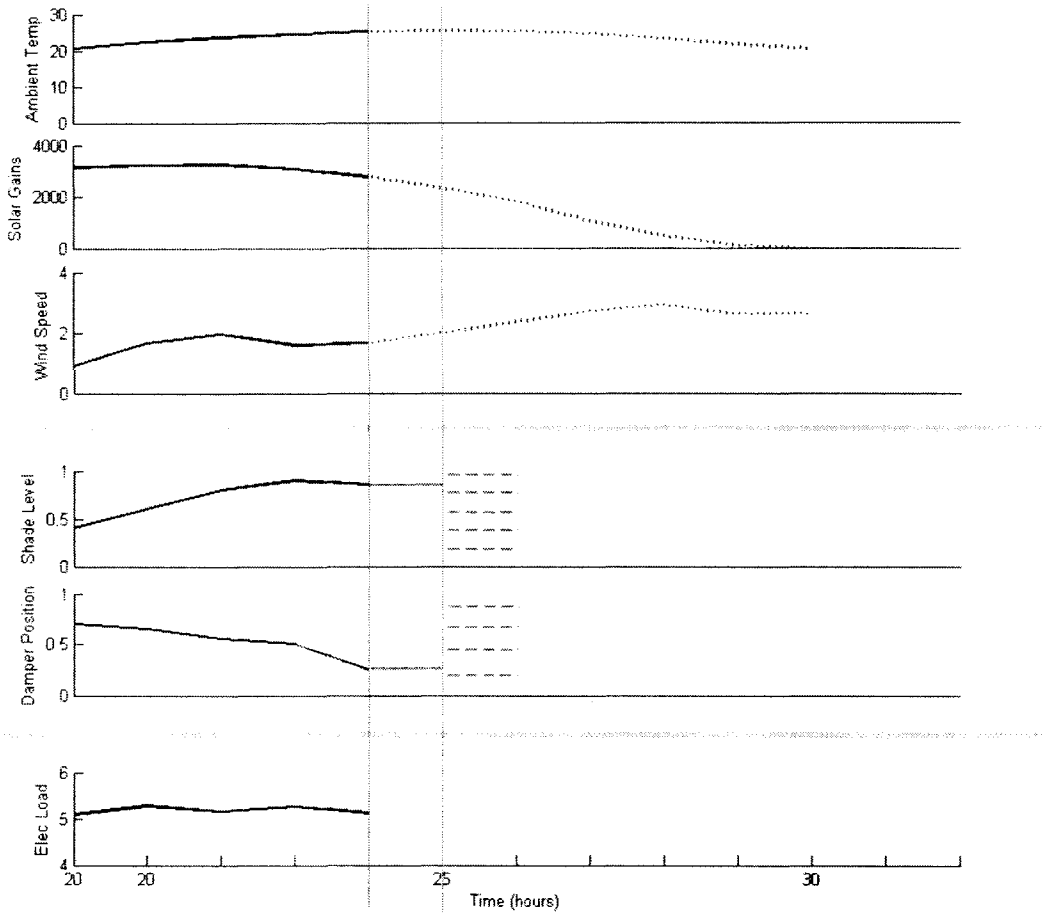
with the supervisory controller continually faced with new sets of current conditions for which it must determine the values of the control variables.

Figure 3.1: Idealized supervisory control problem at time *t*



If infinite computation power were available, then a simulation-based supervisory controller would be able to set these control variable values instantly, so the above figure would be an adequate diagram of the problem. However, the simulation-based optimization process requires some time, so the problem faced by the simulation-based supervisory controller at each time-step is as shown in Figure 3.2. Because of the time lag, the calculations performed by the supervisory controller must use predicted conditions for the point in time for which it is making a decision.

Figure 3.2: Supervisory control problem considered by simulation-based controller

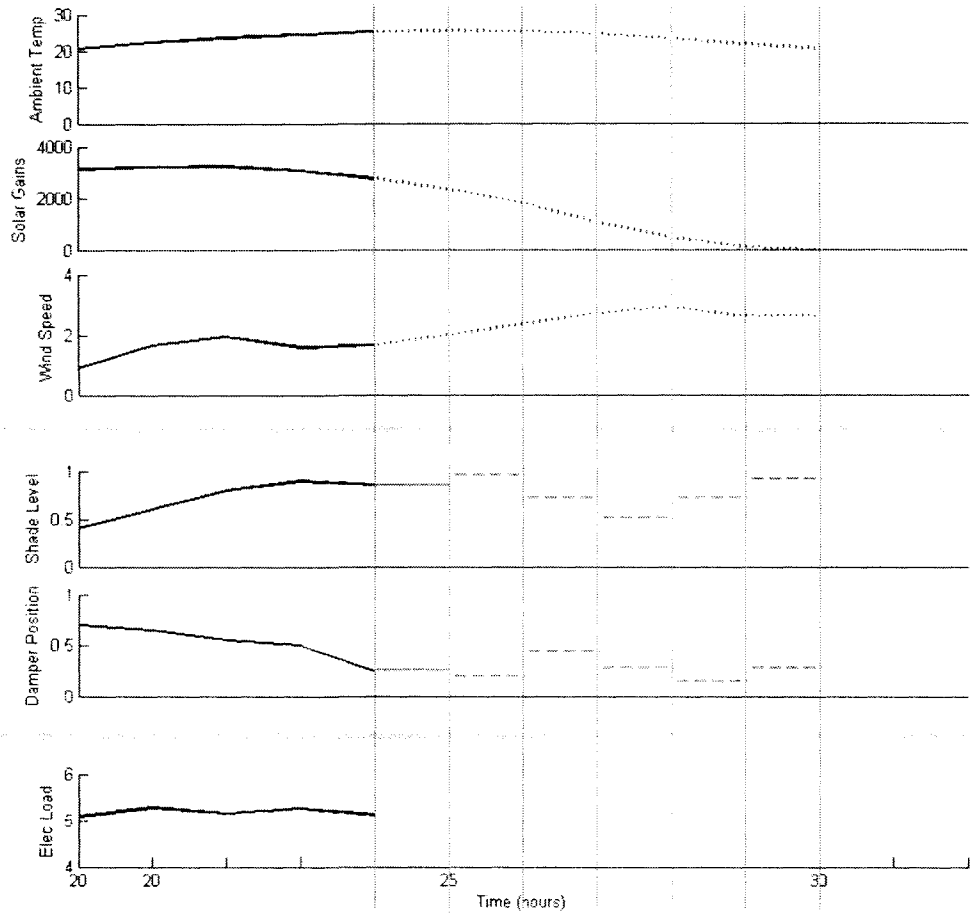


The vertical line on the right represents the future point for which the supervisory controller is currently making a decision. The dashed lines represent some of the many different control values that might be considered by the simulation-based controller in its search for the optimal values. If it wishes to consider all of the permutations of the individual points shown here for each of the two variables under consideration, it would have to consider 5*4 sets of configurations. If the problem involved *i* control variables, and for each variable, *m* different possible values were considered, the supervisory controller would have to consider m^i different sets of control values. Since each set of values is evaluated by running a simulation, then there is an important tradeoff that must

be made between the number of points that can be considered and the length of time that the system must wait to receive a decision.

This concern is exacerbated when more points in the future are considered. As noted in the previous chapter, use of predicted values over a future horizon is necessary for any system involving energy storage, or for any system in which there is a penalty incurred for changing control configurations. (For example, if the system requires x kWh to change the position of a solar shading device, and such an action would save less than x kWh over any one particular controller time-step, this action would not be selected without the consideration of further points in the future, even though one change in position might save substantial energy over the course of a day.) Figure 3.3 shows the problem faced by a simulation-based controller when it considers a future horizon of five time-steps. One of many possible configurations over this future horizon is shown in the graphs of the control variables.

Figure 3.3: Considering a future horizon with predicted conditions



If the controller was to consider five possible values for the first control variable and four possible values for the other one, and was optimizing over the entire future horizon, then it would have to consider $(5*4)^5$ possible sets of values. If greater accuracy is desired than just four or five possible values per variable, or if more variables are placed under consideration, then the search space for the optimization problem grows very quickly. This provides a glimpse of the size of the optimization problem under consideration at each time-step.

Each time that the simulation-based supervisory controller is asked to determine new values for the control variables, it is faced with the optimization problem defined by the following set of equations, where t represents the time for which a decision is required, $c_{k(T)}$ represents the value of condition k at time T, $s_{l(T)}$ represents the value of control variable l at time T, h is the number of future time-steps considered, and Z is the objective function.

$$\text{minimize } Z = f(C,S) \qquad \text{(Eq 3.1)}$$

where $C = \{c_{1(t)}, c_{2(t)}, c_{3(t)}, \ldots c_{j(t)}, c_{1(t+1)}, c_{2(t+1)}, c_{3(t+1)}, \ldots c_{j(t+1)}, \ldots, c_{i(t+h)}\}$,

$S = \{s_{1(t)}, s_{2(t)}, s_{3(t)}, \ldots s_{i(t)}, s_{1(t+1)}, s_{2(t+1)}, s_{3(t+1)}, \ldots s_{i(t+1)}, \ldots, s_{j(t+h)}\}$,

and $f(C,S)$ approximated by simulation.

Most often, the building control system will be configured to call the simulation-based supervisory controller once per time-step (e.g. every ½ hour). However, a variation on this approach is to call the simulation-based supervisory controller only when an event triggers it (such as when a sensor notes that a desired condition is not being met). This approach is used in one of the studies considered in the literature review (Mahdavi and Pröglhöf, 2005). The essential concepts remain the same, except that the controller is only called upon after triggers, rather than every time-step.

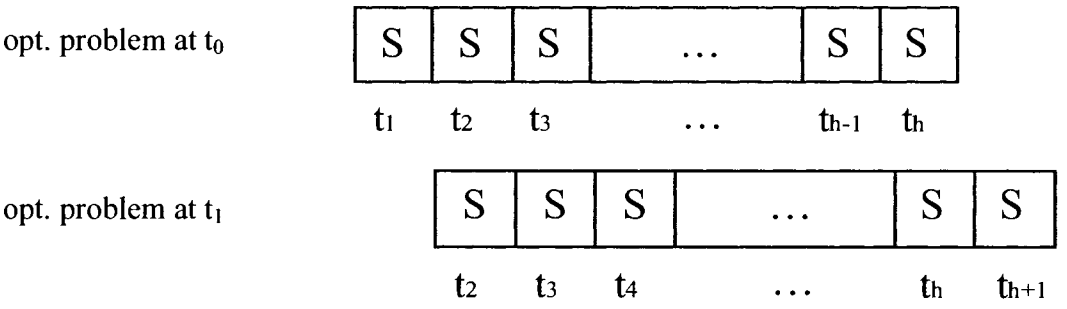
Truly optimal control decisions would be possible if the model being used was perfectly accurate, the predicted values of the future conditions also perfectly accurate, the optimization algorithm was perfectly effective and efficient, and the time-step was infinitely small. None of these conditions can be absolutely met, but sub-optimal decisions are also welcome if they cause better building performance than the default

decisions. And by using this approach while improving models, predictions and optimization algorithms, we can eventually get closer to truly optimal control.

3.1.1 Search Space Overlaps Between Time-Steps

As noted in the literature review, an important aspect of the simulation-based control optimization problem is that each optimization problem faced is quite similar to the problem faced at the previous time-step. In particular, when using a future horizon, the problem is the same as the previous one, except that each time-step is moved back one step, as shown in Figure 3.4

Figure 3.4: Time-step overlap



This feature of the simulation-based control optimization problem turns out to be quite useful, as shown in the results in Chapter 7.

3.2 Software Framework

One of the main drivers for the research discussed here, building on the literature review in the previous chapter, is the idea that research into simulation-based supervisory

control could benefit from a standard framework for the development and testing of simulation-based control algorithms and applications, similar to the way that GenOpt facilitates the development of simulation-based optimization algorithms and their application to design optimization. We look first at the functional requirements of such a framework. A brief description is then provided of the prototype software that has been developed to meet these requirements, followed by a description of the virtual testing environment used for the case study.

3.2.1 Requirements

Based on the essential concepts of simulation-based supervisory control discussed above, the lessons learned from previous studies in this field, and the current state of development in building simulation and optimization software, the following aspects should be included in a software framework:

- Simulation-tool independent (can use a variety of simulation programs)
- Ease of user-modification and further development: since this technology is still in the research phase, people using the software will want to have a great deal of control over its details; and the software must be made such that it can be easily improved upon in the future
- Addressing the considerations of thermal storage and system modification energy penalties
- Addressing the need for model accuracy and keeping the models up to date
- Addressing the challenges of optimization with time constraints and avoiding local minima

Two possible ways of dealing with this last challenge are particularly promising, and the software framework should be set up to facilitate their application:

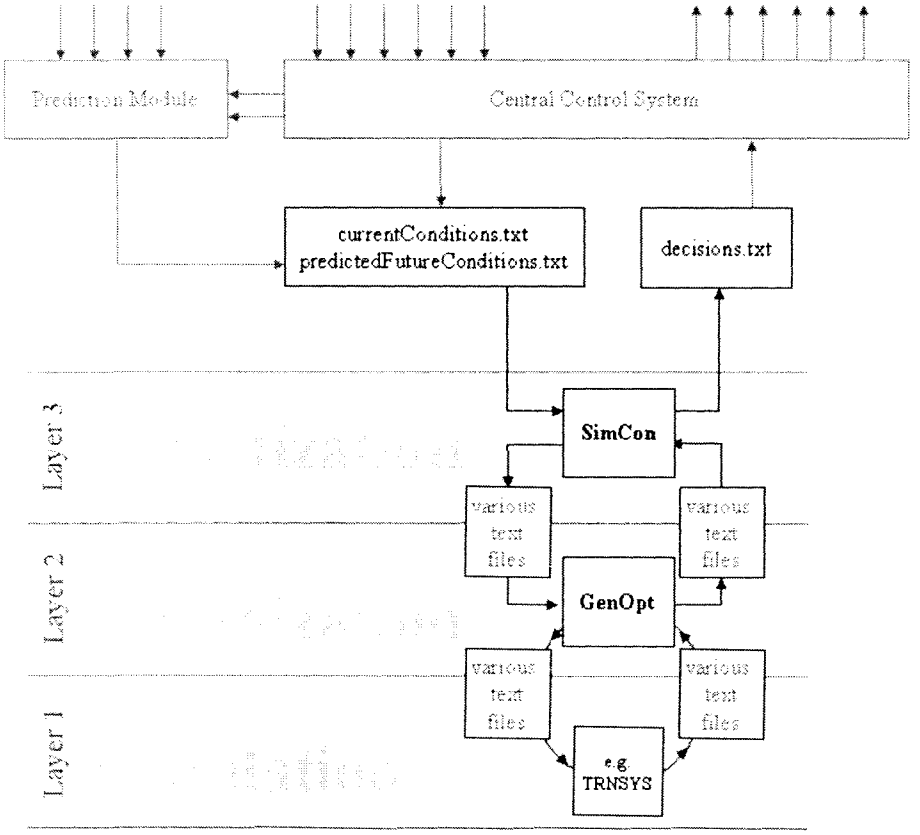
- Using knowledge-based rules to identify promising areas of the search space
- Learning from optimizations at previous time-steps

It was also decided that since GenOpt is a relatively well-established optimization tool for building simulation, software for control optimization could well build upon this existing software, rather than starting from scratch. As such, GenOpt plays a central role in the software framework described below.

3.2.2 Overview

Within the developed software framework, the problem of supervisory control optimization is broken into three layers. The first layer consists of the simulation software that is being used to evaluate possible control configurations. The second layer is the optimization layer, which runs one of many possible optimization algorithms in search of the best control configuration for the given time-step. The third layer is an organization layer, which interfaces with the building's central control system, sets up the optimization problem at each time-step, and performs a number of other organizational and learning tasks. Text files are used to pass information between layers. The basic structure is shown in Figure 3.5, along with how it interfaces with the building's central control system and with a prediction module.

Figure 3.5: Software framework structure



It is felt that the prediction of future conditions is a separate problem from the optimization problem itself, so this was kept as a separate module, outside of the software framework, and is not dealt with in detail here. Depending on the conditions data necessary for the system under consideration, the prediction module might use a variety of different methods to make its predictions. It might use historical trends in the conditions data it gathers from sensors or from the central control system (or from the database created by SimCon, which is discussed later). It might use automated downloads from weather information providers on the internet. Or it may use some more complex method, or some combination of these two approaches. Regardless, weather prediction (or electricity price prediction, etc) is a complex consideration that must be dealt with

separately – in the studies presented here, we look primarily at cases where the predictions are fully accurate, but we also present some initial studies of the effects of prediction inaccuracies on performance.

The simulation layer of the software framework can use any energy simulation program, as long as it reads and writes to text files and can be called from the command line. The optimization layer uses GenOpt as the interface between the optimization algorithms and the simulation program. The organization layer uses a java program named SimCon, which was written to meet the requirements of this layer.

For a given time-step, the computation process begins with the text files for the 'current conditions' and the 'predicted future conditions' being read by SimCon, and ends with a 'decisions' text file being written by SimCon. So from the point of view of a central control system, its use is quite simple, requiring just that it write some of its sensor information to text files and send them to SimCon, wait a pre-determined amount of time, and then read a decision text file, which tells it what set-points to use.

However, the internal workings of the framework are less simple. Some of the details of how SimCon works are discussed in Appendix 1.

3.2.3 The Organizational Layer

Aside from its role as traffic controller, the organizational layer does two important things: (1) it stores the results from previous time-steps; and (2) it writes an 'optimization instructions file' which gets read by the optimization algorithm within the optimization layer. Similar to how GenOpt has an extendible library of optimization algorithms, SimCon has been developed with an extendible library of 'optimization instruction'

algorithms, from which the user can select (or develop) the algorithm that will write the 'optimization instructions' file. Since this file must be read by the optimization algorithm in GenOpt, these two algorithms must be coordinated to some extent. In the case study considered here, the optimization instructions file is used to specify starting points and constraints for the optimization algorithm. (Other uses are possible, as discussed briefly in Chapter 7, and should be considered in future research.) In particular, the 'optimization instruction' algorithms used here specify starting points and constraints based on the results from the previous time-step, and based on knowledge-based rules about what points will likely be good. These two features – learning from previous time-steps and knowledge-based starting point rules – figure prominently in the results presented in Chapter 7. They have a significant and positive effect on the performance of simulation-based control.

3.2.4 Virtual Testing Environment

A single run of SimCon is just for one time-step – it takes conditions and predictions information, performs the optimization, and then returns decisions information. In order to test its performance, SimCon needs to be used within a real-time environment, with a central controller sending it conditions and predictions information at each time-step, and then acting on its decisions. This could be an actual building control system. Or, more practically, and in some ways more usefully for research, it can be connected to the control system of a virtual building. This link between SimCon and another building model could be done a number of different ways. The Building Controls Virtual Test Bed, currently being developed at LBNL (Wetter et al, 2008) could provide a good

avenue for this in the near future. In the case study considered here, the virtual building is a copy of the model used within the simulation-based controller, but slowed down to real-time (through the addition of a TRNSYS object that causes the model to sleep for a specified time period at each time-step), and with the addition of a supervisory control object that calls SimCon. (This is done through an Excel link, as discused in Appendix 1.)

This two-model virtual testing arrangement allows not only for easier testing of SimCon than would be possible through an actual supervisory controller implementation in a real building, but also allows one to test various configurations under identical conditions, and allows one to fairly easily consider the impacts of various model inaccuracies (by changing parameter values so that they are different in the two models), improper initialization of the model in the controller, or prediction inaccuracies.

3.3 Modified Genetic Algorithm for Simulation-Based Control

An initial 'optimization instructions' algorithm and optimization algorithm pair was developed for this framework, based on a genetic algorithm. Although genetic algorithms have a number of promising qualities, such as their ability to avoid local minima, and some qualities that may make them non-ideal for control optimization applications, particularly their slow convergence rates, the main reason for their use here is because of the interesting possibilities for coupling with SimCon. In particular, a genetic algorithm can be initialized with numerous starting points (instead of just one, as is the case with many other optimization algorithms), so they can use some rule-based or learning-based suggestions for starting points, and still have other randomly-generated

starting points that may allow it to discover unexpected points that may be better than those around the given points.

3.3.1 Basic genetic algorithm description

The current release of GenOpt does not include a genetic algorithm in its library, so one was added to the library for this research (based on a genetic algorithm I wrote in java for a course project). It uses one-point crossover and uniform mutation, and allows the user to specify the population size (in the case study: 15), the percentage elite to maintain (20%), and the rates for crossover (80%) and mutation (20%). To avoid problems that may result in later generations due to a depletion of diversity, the algorithm is configured to test for stagnation and if necessary it increases the mutation rate to 100% for a generation. It also allows the user to stipulate the maximum number of simulations to run (a necessary addition for use within control optimization), and the user can choose to keep the values considered for each variable to a desired level of precision by having the GA search over points on a user-defined grid (0.1°C). This last consideration was added to avoid wasting simulations on points that are very close but not identical.

3.3.2 Using the optimization instructions file

The genetic algorithm in GenOpt is also configured to read and use starting point and constraints information from the optimization instructions file. A companion 'optimization instructions' algorithm was written for the SimCon library which writes starting points and constraints to the instructions file, based on the following three considerations.

3.3.2.1 Using results from previous time-steps for starting points

Since a future horizon is being used in the optimization, there is an overlap between the previous solution and the current case – what was the second time-step in the previous solution is now the first time-step in the current case. So the values obtained in the previous solution (which are stored by SimCon, currently in another text file) can be transferred over to an initial point in the current optimization by moving them all back one time-step, and finding an arbitrary way of filling in the last time-step values for the current point – in this case, the last time-step value is set equal to the second-last one.

3.3.2.1 Using rules for suggesting starting points

When Henze et al (2005) performed their case study on thermal storage control using simulation-based control, they found a number of points where the simulation-based controller actually performed worse than the standard rule-based controller they were comparing against. In these cases, the simulation-based controller was stuck in local minima. This situation of performing worse than a rule-based controller could easily be avoided (at least in the case of a well-calibrated model and good initialization and prediction) if the knowledge embedded in the rule-based controller was also embedded in the simulation-based controller. (Note that, with an imperfect model or poor predictions, this knowledge is not necessarily enough to keep the simulation-based controller from performing worse than the rule-based controller, as discussed in the Chapters 6 and 7.)

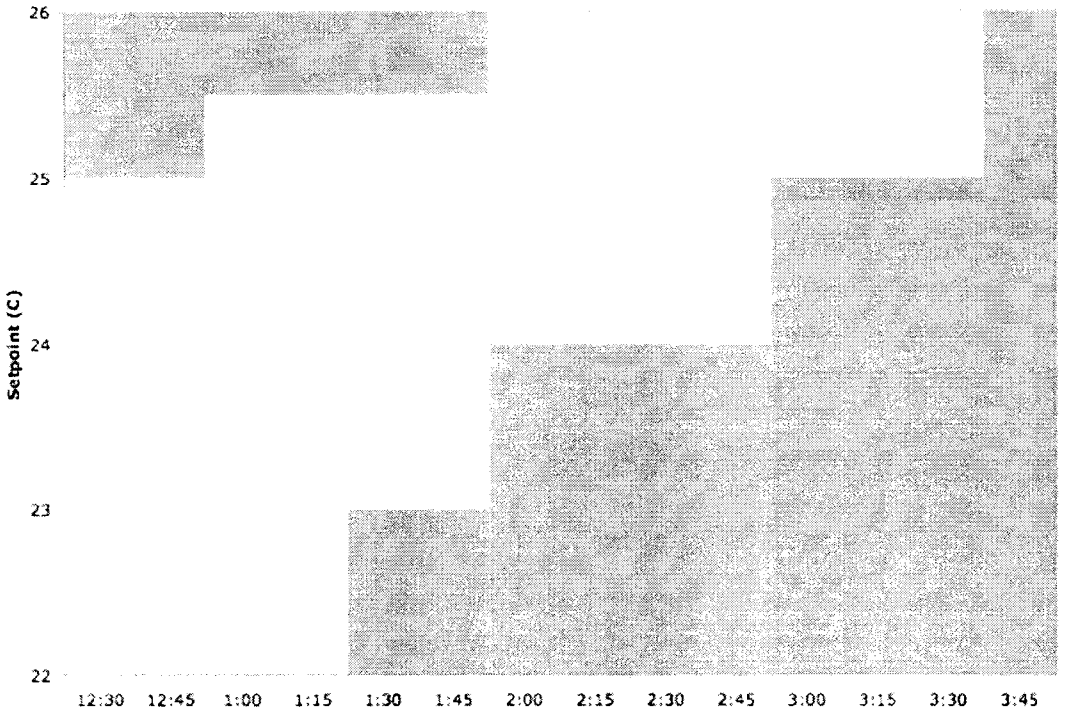
Of course, one of the reasons for using model-based control may be because the system or the control objectives are too complex to devise good control rules for the

system. So in some cases, this approach of using rule-based suggestions may not be appropriate. But in this case, the logarithmic trim has already been shown to be a very good, though sub-optimal, control rule. So it is used here to suggest starting points, which we hope to improve through successive iterations of the genetic algorithm.

3.3.2.3 Using rules to set search-space constraints

In order to make the time-constrained optimization problem more tractable, it is helpful to eliminate parts of the search space that are unlikely to produce good control points. From knowledge of the given case, it may be assumed, for example, that the optimal point does not include set-points near the end of the trim that are less than 25°C. So the constraints for these variables can be modified to $25°C < s_i < 26°C$. Using this type of reasoning, the optimization instructions file sets the constraints shown in Figure 3.6, where the grey areas have been eliminated from the search space.

Figure 3.6: Constraints set by the optimization instructions file



3.4 Notable Limitations to Framework and Algorithm

Although the framework and algorithm described above are good starting points for the study of simulation-based control optimization, some limitations should be noted before we dig into the details of the case study.

Framework limitations:

- Discrete time-steps are required. The software framework, as currently implemented, does not have any way of specifying continuous control.
- Constant time-step lengths. There is currently no way of using a variable time-step length for a particular variable, nor between variables.
- Forward modeling only. There is currently no way of backing out set-point values from the simulation given a desired objective value.

Algorithm limitations:

- As a genetic algorithm, it is inherently stochastic, so different runs of the same algorithm produce different results.
- There is no convergence test with the genetic algorithm, so there is no way of determining if the values found by the algorithm are the exact optimum.

3.5 Summary

This chapter has outlined the essential concepts, software framework and algorithms that are used in the case study. It is hoped that the software framework can be useful for further studies of simulation-based control, and that it can be further developed by building on the lessons learned from the case study. Similarly, the modified genetic algorithm used herein can hopefully be improved through further research, and more simulation-based control algorithms can be developed and tested within this framework.

We now begin to look at the case study in detail, beginning with a description of the office space and the TRNSYS model in the next chapter.

Chapter 4

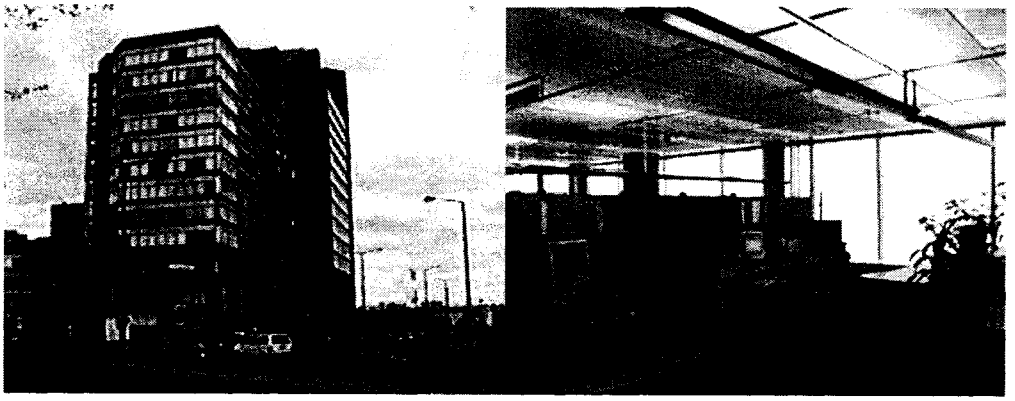
Case Description and Model

The office space and ventilation system studied is described in the first section of this chapter. The TRNSYS model of it is described in some detail in the second section, with some details in Appendix 2. The verification and calibration of the model is discussed in the third section, with some of the details of the calibration approach relegated to Appendix 3. The chapter ends with a description of the weather conditions for the day used in the demand response control studies.

4.1 System Description

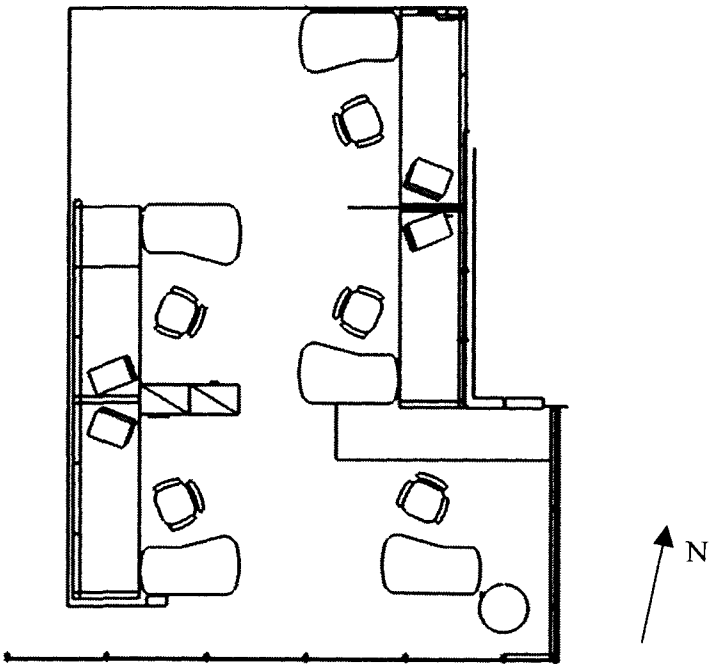
The ventilation system under consideration is a jet-diffuser system with personal environmental controls (referred to herein as the 'PEC system'). It is an experimental system installed in the office space of the Innovations & Solutions Directorate (I&S) of Public Works and Government Services Canada (PWGSC), in the Place du Portage office complex in Gatineau, Quebec. Figure 4.1 shows the building and the office space.

Figure 4.1: Place du Portage, Phase III; and office space of I&S



The PEC system is installed for 19 workstations (divided into five subsections: three subsections with 4 workstations each, one subsection with 5 workstations, and one subsection with 2 workstations). The model described here is only for a subsection of 5 workstations – to try to bound the model complexity and computation time, it was decided that only one subsection would be modeled, and at the beginning of model development, the subsection of 5 workstations had the most reliable data, so it was chosen. This office subsection is shown in Figure 4.2. The south wall is an exterior wall, and is approximately 65% glazing. The east wall is an interior wall. The north and south boundaries of the area are demarcated by partitions, but are open to the rest of the office space.

Figure 4.2: Office area with five workstations

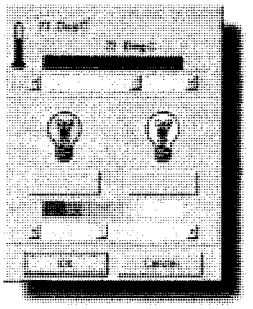


Each workstation is provided with a dimmable light and a ceiling-mounted jet-diffuser nozzle that shoots supply air towards the floor near the occupant. Compared with a standard air distribution system, the jet diffuser system provides supply air more directly to the occupants, instead of attempting to condition the whole space. This results in vertical temperature gradients throughout the space, as the air in the upper part of the room is not conditioned as much as the air closer to the occupants.

Since the supply air is provided more directly to the occupant, the ventilation system can increase or decrease supply air to a particular workstation by increasing or decreasing the flow rate through the diffuser, which in turn affects the temperature of that workstation relative to the others in the office. This system thus allows for more individualized climate control at each workstation. Temperature sensors are mounted at seated head level on a partition at each desk, providing the feedback necessary for temperature control at each workstation.

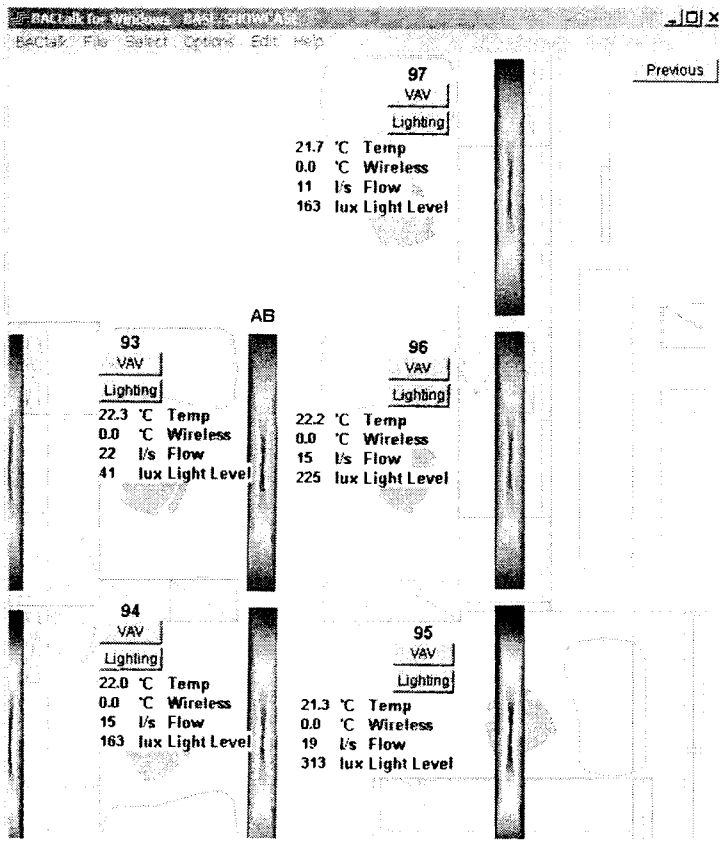
This individualized climate control is combined with individualized lighting control with dimmable lights. Occupants are allowed to set their own temperature and light level set-points either through electronic displays at their desk or through icons on their computer (as shown in Figure 4.3).

Figure 4.3: Occupant set-point control through their computer



All of the occupant set-points, along with the sensor readings of temperatures and flow-rates, are fed to a central control system, which performs the necessary control actions, and which also logs data and makes it available to researchers through an html interface. A large amount of data is available, and it is organized visually into various different screens, one of which is shown in Figure 4.4.

Figure 4.4: An example of data presentation for one section of the I&S office



The occupant-defined set-points can be over-ridden by system administrators through the online system, but this function is currently not used (except for control-system testing or for system researchers to play jokes on colleagues in the office). The case study presented in this chapter will consider over-riding occupant set-point values

during demand response actions by using this central system for supervisory control to determine individual set-point values.

4.2 System Model

The first step in setting up a simulation-based supervisory control system is to produce a computational model of the system under consideration. The model must be able to take inputs describing the current system state, the context (e.g. outdoor temperature), and particular values of the control variables (in this case, the temperature and lighting set-point values), and output reasonable estimations of the occupant comfort and system energy use associated with these inputs. For this case study, the model was created using TRNSYS.

4.2.1 Modeling Assumptions

For the creation of this model, a number of simplifying assumptions had to be made about the office space and air diffuser system under consideration. The most important assumptions are discussed in this section, before starting into a detailed description of the model in the next section.

The office space was modeled as a rectangular volume, ignoring the slight variation along the east side of the space. All of the boundaries, including the floor and ceiling, were assumed to be adiabatic, except for the external wall along the south side of the space. Aside from the small exterior portion along the south end of the east wall, the east wall is an interior wall, and is thus assumed adiabatic. The consideration of the north and west sides of the space is more difficult – they are both open to the rest of the office

spaces on that floor, with just cubicle partitions providing a boundary line. It is very difficult to determine what heat transfer might be occurring with the rest of the spaces, and this aspect of the model causes some uncertainty about its outputs. The office area to the west of the space is equipped with the same type of ceiling-diffuser system, and is expected to have similar temperatures (within 1°C) and airflow patterns as the space under consideration, so heat exchange between these two spaces is likely fairly small, and is assumed to be zero in the model. To the north of the space under consideration is an open walkway that connects the office space to the rest of the floor. It is unclear how much air mixing occurs with the rest of the floor through this walkway, and it is unclear if it would result in a net heat gain or a net heat loss for the space under consideration, or if this heat exchange would be very significant. Again, it is assumed in this model that the net heat exchange is negligible, based on the notion that the sources of heating and cooling are dispersed well enough throughout the office floor that large heat transfer rates do not occur between particular office spaces. But this is a source of uncertainty.

The model is used here to study demand response through temperature setback using the air diffuser system in this space. It is assumed that as the temperature set-points are rising within this space, they are rising at a similar pace throughout the rest of the office spaces in the building. As such, the assumption that these boundaries are adiabatic is still used. But under the more complex conditions of rising set-points, the modeled space's interaction with its surrounding spaces is likely more complex, and the adiabatic assumptions are more likely to be inaccurate. But no readily-available solution can be found to make the boundary conditions more accurate without also making the model more computationally demanding, and thus inappropriate for the quick simulation runs

needed in this model-based control study. So the adiabatic assumptions are used, but the resultant uncertainties in the model accuracy are noted.

The heat gain or loss through the external south wall is considered in four parts: conduction through the wall; infiltration or exfiltration; solar gains; and long-wave radiation. Most of the details for these considerations are discussed below, but three particular assumptions should be noted here.

(1) A constant value was used for the convective heat transfer coefficient between the inside surface of the wall and the room air. This is a common assumption to use in building modeling, but is never ideal, and is particularly less than ideal when modeling a space that has ceiling jet diffusers with variable flow rates.

(2) When modeling office buildings, one can often ignore the effects of air transfer through the envelope by assuming that the building is pressurized and thus no infiltration occurs, and any exfiltration has no net effect on the building's energy use because the escaping air is replaced by outdoor supply air that had to be brought in and conditioned regardless. But the model discussed here does not include the central HVAC system of the building (it only includes the zone's diffuser system and induction air units, taking the supply air from the central system as an input to the model), so the air transfer through the envelope must be considered whether it is infiltration or exfiltration, since either one represents a load that must be met by the supply air through the diffusers, and this supply air flow rate calculation is needed within the model. (And it is also not clear that the air transfer through this external wall on the 8^{th} floor is entirely exfiltration, so it may have to be considered even if the entire HVAC system was considered as well.) To consider the heat loss or gain because of this infiltration or exfiltration, the infiltration equation

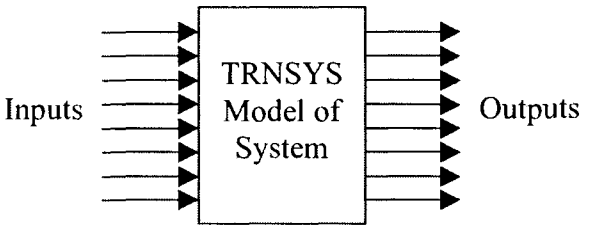
available within the TRNSYS Type19 module was used in this model. This is likely fairly accurate as long as the pressure difference between the office space and the outdoor air is not so large that it outweighs the effects of temperature difference or wind speed, and as long as this pressure difference is relatively constant. But this may be a source of inaccuracy in the model.

(3) The window on the south wall has white, vertical Venetian blinds on the inside of the window. These can be adjusted by the occupants to provide some solar shading for the space. As such, the solar transmittance ratio used as an input value to the Type19 zone module in this model should vary throughout the simulation, according to the positioning of the blinds. However, no data is kept concerning the positioning of these blinds. In the model, it is assumed that the blinds are not adjusted, and the solar transmittance ratio used was based on a combination of judgment from observation of the space and on calibrating this value slightly to make it better fit measured data.

4.2.2 TRNSYS Model Overview

At the broadest level of analysis, the TRNSYS model of the system can be viewed as a simple box with a set of inputs and outputs, as shown in Figure 4.5. This is how the model is viewed by the optimization algorithm within the simulation-based control system, so it is worth highlighting these inputs and outputs before looking at the details of the calculations it uses to determine the output values given the inputs.

Figure 4.5: Model overview



Inputs (time-series input data, read from a text file):

- Weather data
  - Ambient temperature (T_{amb})
  - Diffuse solar radiation passing through window
  - Direct solar radiation passing through window
- HVAC system data (air coming into the duct that feeds the PEC system)
  - Supply air temperature (T_{supply})
  - Supply air pressure (P_{supply})
- Occupant choices or overrides (x5)
  - Occupant sensor – occupied / not occupied
  - Occupant light level
  - Desired temperature ($T_{set\text{-}point}$)
- Other
  - Zone light level (general lighting)

Outputs (time-series data, written to a text file):

- Intra-zone temperatures
  - Temperatures at each occupant's thermostat (T_{sensor})
  - Return air temperature (T_{return})
  - Inside surface temperature of window ($T_{surface}$)
- Flow rates
  - Diffuser flow rates (x5)
- Energy use estimates
  - Lighting electricity demand
  - Cooling electricity demand

The weather and HVAC system data both represent inputs to the modeled system that are beyond our control. The occupant choices are usually also not controlled by the central control system, except in the cases of over-rides, as studied here.

It should be noted that the model can easily be configured to provide a selection of many different output variables. For the purposes of the control objectives considered here, the outputs listed above are deemed important, but if one wished to control for different objectives, then the model outputs could be easily modified.

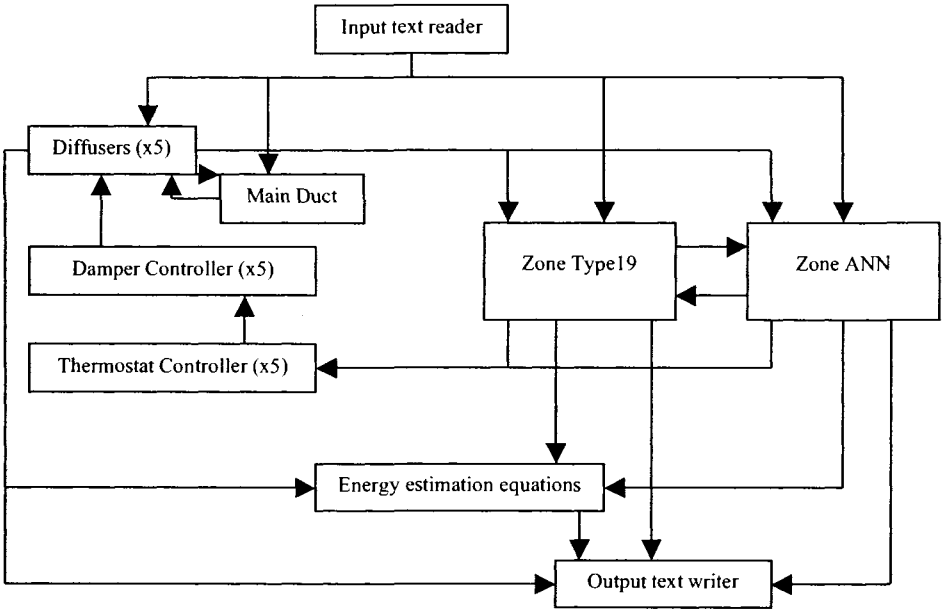
The intra-zone temperatures are essential outputs of the model, as they relate to recorded temperatures in the space and thus allow the model to be tested for accuracy and calibrated. These intra-zone temperatures are also used in determining the energy use estimates.

4.2.3 Modules within the Model, Description of Basic Internal Logic

The internal logic of the model is shown in Figure 4.6. Aside from the text readers and writers, and the equations written within the 'deck' file itself, six different types of modules (called 'types' in TRNSYS) are used, and three of them are duplicated five times (one for each workstation), so the entire internal structure of the model is defined by eighteen modules (called 'units' in TRNSYS). The thermostat controller modules and damper controller modules work together to calculate the five damper positions given the occupant set-points and the temperature readings at each workstation. The duct module and diffuser modules work together to calculate the airflow rate from each supply air diffuser given the supply air pressure and the five damper positions. The bulk air module (the standard TRNSYS Type 19 module) is used alongside an ANN module to calculate

the intra-zone temperatures given the weather inputs, occupant inputs (light levels, occupancy), the supply air temperature and the diffuser airflow rates.

Figure 4.6: Internal logic of model



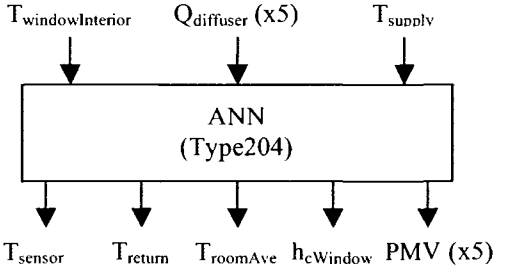
4.2.4 Zone modules

The crux of the modeling challenge with this system is that it produces non-uniform temperature distributions within the office space. The approach used in this model, developed by the author and L Zhou, as discussed in detail in Appendix 2, is thus perhaps also useful for other systems that produce non-uniform temperature distributions within the zone. It uses an ANN module to provide information about the temperature distributions, while still relying on the standard heat-balance approach embodied in the Type19 module to determine the bulk air temperature.

4.2.4.1 ANN module

The ANN is pre-trained to mimic a CFD module using a database of CFD inputs and outputs, so the use of the ANN module is akin to a fully-coupled TRNSYS-CFD simulation, but with much faster run-times (seconds per simulation rather than hours). The accuracy of the ANN module is determined by the size and quality of the CFD input-output database. As discussed in Appendix 2, the CFD model and ANN for this case were created by L Zhou, and an ANN module was written by the author for use in TRNSYS, using the weights and biases for the CFD-trained ANN. The module has the input and output variables shown in Figure 4.7. The PMV value and the heat transfer coefficient values are not used in the present study, but were included since they may be of interest in future studies with this model. The role of the ANN module is to determine the temperature distribution, particularly the relative values of the return air temperature, the sensor temperatures at each workstation (all five of which are assumed to be the same in this study), and the average room air temperature.

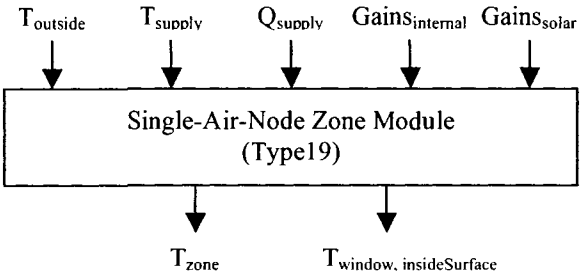
Figure 4.7: Inputs and outputs from the ANN module



4.2.4.2 Bulk-air module (TRNSYS Type 19)

The bulk air module used is Type19, which is available in the standard TRNSYS library. In the context of the current model, the most important inputs and outputs from this Type 19 module are shown in Figure 4.8.

Figure 4.8: Most significant inputs and outputs of single-air-node zone module



The module uses a single node for the room air and furniture. At each time-step (the simulation runs at three minute time-steps to simulate the diffuser controls accurately), it calculates a heat balance for that node, before updating the node air temperature based on the heat balance, the thermal capacitance of the air and furniture, and the previous temperature. To do so, it must simultaneously calculate the heat balance on each interior surface and each exterior surface. So the module (with the mode used in this model) does not assume that the zone temperature set-point is met, and then calculate a cooling or heating load based on that assumption – it instead allows the temperature to fluctuate, allowing for realistic simulation of the diffusers and their control.

Some of the key parameters for this module are provided in the following tables. Table 4.1 shows some of the key zone variables and Table 4.2 provides some details of the wall constructions.

Table 4.1: Type 19 zone air node parameters

Volume of air	116.13 m^3
Room capacitance	400 kJ/C
Occupants	5
Activity level	4
Lighting	116 W
Computer load	48 W
K1, K2, K3 (infiltration parameters)	0.106, 0.017, 0.049

Table 4.2: Type 19 wall parameters

	Area (m^2)	Description
South Wall	14.7	8" heavyweight concrete with 2" insulation
East Wall	23.7	8" heavyweight concrete
North Partition	14.7	Frame, partition with 0.75" gypsum board
West Partition	23.7	Frame, partition with 0.75" gypsum board
Floor	38.7	8" heavyweight concrete floor deck
Ceiling	38.7	8" heavyweight concrete deck with false ceiling

The dimensionless infiltration parameters noted in Table 4.1 are used in Equation 4.1, within the Type 19 module, to determine the infiltration load, where ρ and V are the density and volume of the room air. The parameter K1 is used to weight the constant component of the infiltration, K2 weights the buoyancy-driven component, and K3 weights the wind-driven component. The TRNSYS default values for K2 and K3 were used in this model. The value for K1 was tweaked slightly from the default value through the calibration process described in Section 4.3. Note that this equation is intended for use in fully-mixed zones – the zone temperature stratification may augment the buoyancy-driven infiltration, so this aspect may be underestimated slightly in the model.

$$\text{Infiltration mass flow rate} = \rho * V * (K1 + K2*(T_{amb}-T_{zone}) + K3*V_{wind}) \qquad \text{(Eq 4.1)}$$

4.2.4.3 Interaction between ANN and bulk-air modules

These two modules act in tandem to calculate the sensor temperatures at each workstation and the return air temperature. As discussed in Appendix 2, the Type19 module is used to determine the bulk temperature of the room and the ANN is used to estimate the magnitudes of the temperature variations within the space. So, the point temperature at the sensor is calculated as shown in Equation 4.2. The determination of T_{return} is similar.

$$T_{sensor} = T_{room\_Type19} + (T_{sensor\_ANN} - T_{room\_ANN}) \qquad \text{(Eq 4.2)}$$

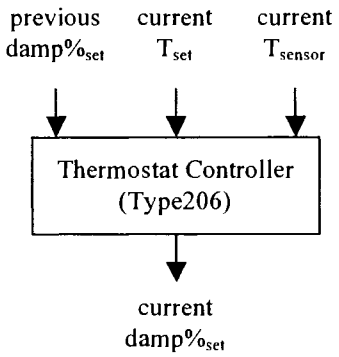
4.2.5 Modules for diffuser system

To appropriately model the diffuser system, new modules were created in TRNSYS. The controller modules are similar to other controller modules that are available in the standard TRNSYS library, but they were programmed separately in part to allow them to be simplified, and to make it easier to modify them during the course of model development. The diffusers module and the main duct module consider the interactions between the five diffusers, and are custom-made for this diffuser system model.

4.2.5.1 Controller modules

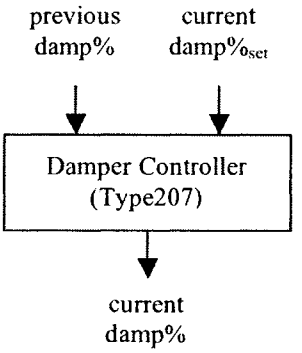
The thermostat controller module sets the damper position set-point, using the current temperature set-point, the current sensor temperature, and the previous damper position set-point. The user can specify the relationship between the temperature difference and the change in damper set-point through a module parameter.

Figure 4.9: Thermostat controller module



The damper controller module uses the set-point determined by the thermostat controller and the previous damper position to determine the current damper position. The damper controller module also has a parameter that specifies the relationship between the damper position difference and the change in damper position.

Figure 4.10: Damper controller module

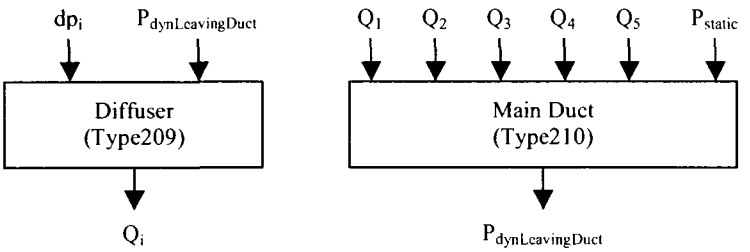


4.2.5.2 Diffusers module and Main Duct module

The diffuser module and main duct module use the five damper position values, along with the static pressure in the duct, to determine the airflow rates through each of

the diffusers. Parameters are used to set the pressure-drop relationships for each branch of the local duct system. The calculation assumes that the diffusers' exiting static pressure is approximately equal to the ambient pressure.

Figure 4.11: Diffuser module and main duct module



4.2.6 Considerations for the induction units in the space

In addition to the five ceiling diffusers, the space is also conditioned by four induction units. The supply air temperature for the induction units varies inversely with the outdoor ambient temperature, ranging between 15°C and 40°C. The airflow and temperature gradients due to the induction units are considered within the CFD model, and thus are considered within the ANN module. Within the Type 19 zone module, the induction units are modeled as internal loads, with the thermal gain or loss being a function of the induction supply air flow rate and the difference between the induction air supply temperature and the bulk temperature of the space. However, since the real-time temperature values for the induction supply air are not being monitored, this temperature had to be assumed based on the given range of temperatures and a few spot measurements. The assumed temperatures are reasonable, but this is one area where the model could be made more accurate (the induction air temperatures should ideally be an input to the model, rather than an assumed value based on the season). Another

complication with these induction units is that they are controlled to modulate their output temperature somewhat by increasing or decreasing the fluid flow through a small cooling coil on the induced air side. This modulation was neglected in the model, with the induction units just treated as a constant heat source or sink during the simulation. So this is another area where the model could be made more accurate.

4.2.7 Energy estimations

The system being modeled does not include the primary energy consumers – the office space with the diffuser system is just one office space among many in a large complex. Modeling the entire complex is beyond the scope of this project, so simplified methods are used to estimate the electricity use attributable to the cooling and lighting of the system under consideration. The electrical load from the lights is estimated based on the illumination level of each of the units, and an estimated power draw per unit when it is fully illuminated. The effect on the building's cooling energy use is somewhat more difficult to estimate. Equation 4.3 is used, with the cooling system COP assumed to be 3.0.

$$\text{Cooling System Elec Load} = Q_{supplyTot}*(T_{return}-T_{supply})*cp_{Air}/COP_{chillerSys} \quad \text{(Eq 4.3)}$$

Only cooling and lighting energy are estimated since they are the only two elements significantly affected by changes in the occupants' temperature and lighting set-points. The effects of set-point changes on the building's fan power consumption in this particular case is complex but essentially negligible, since the diffusers system for this

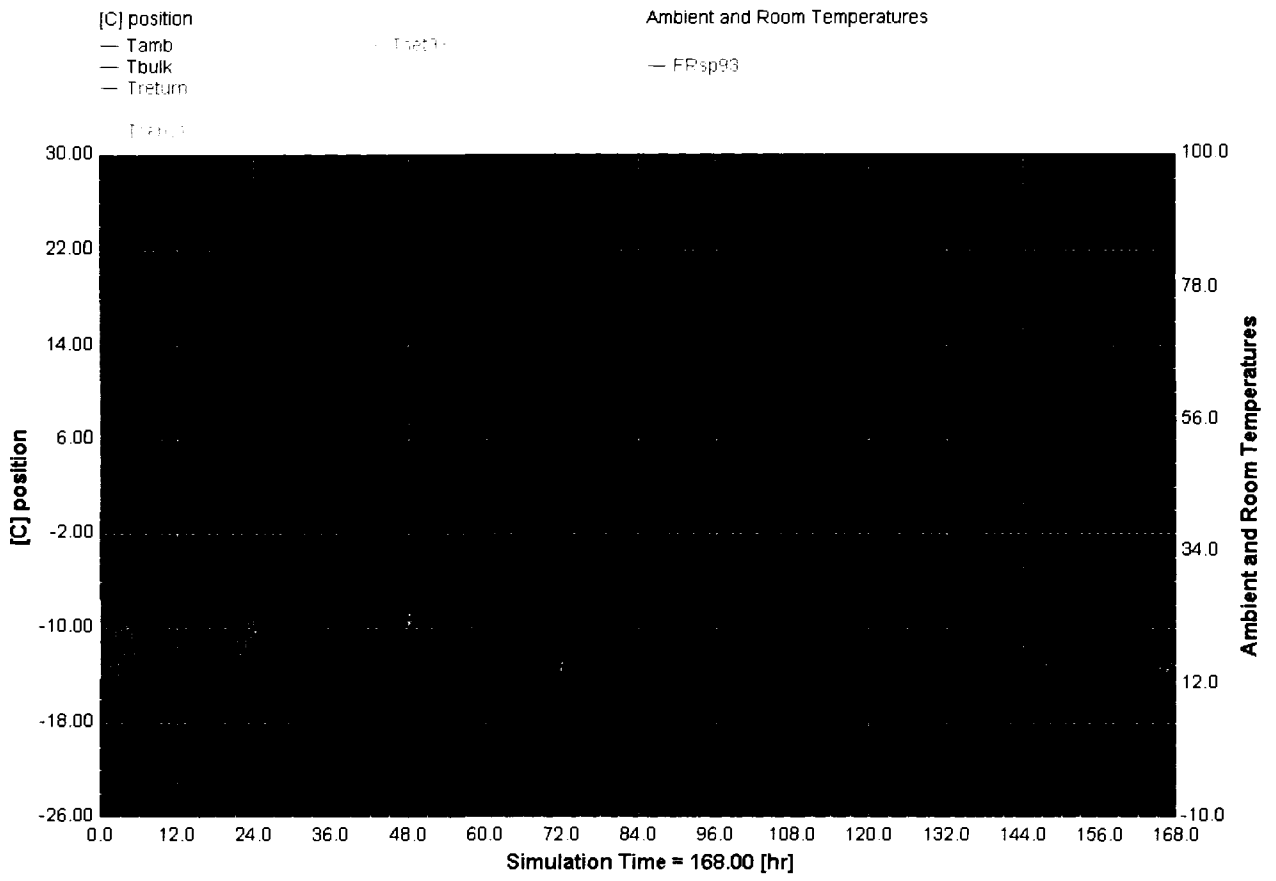
zone was patched onto the existing HVAC system in such a way that the air entering the zone is dampered to maintain a lower relatively stable intake static pressure for the zone, regardless of the damper positions or flow rates through the diffusers.

These estimations are not intended to provide reliable data about the system's energy use for reporting purposes, but rather are used here to compare the energy use implications of different system configurations during the control optimization process.

4.3 Model Verification and Calibration

Figure 4.12 shows an example graphical output from the TRNSYS model. The graph shows a simulation of one week, which takes approximately 15 seconds to run. The three lines at the top show temperatures in the space – the green line is the sensor temperature at a workstation (in this case, all of the workstations are presumed identical), the blue line is the average bulk air temperature in the space, and the purple line is the return air temperature. (Note that when the diffuser system is shut down at night or on weekends, the model assumes that the air in the zone is fully mixed, so the three lines are identical.) The red line is the outdoor air temperature. The orange line is the inside surface temperature of the window. The pink line at the bottom is the airflow rate through one of the diffusers (using the scale on the right axis), which rises and fluctuates during the day to maintain a sensor temperature near 22°C.

Figure 4.12: Example graphical output from TRNSYS model



The estimated lighting loads are constant during the day, at 116 W (52 W from occupant lighting, 64 W from general lighting). The estimated cooling loads are a function of the airflow rate through the five diffusers (which add up to approximately 125 L/s during the daytime peaks), and the difference between the supply air temperature and the return air temperature. During the daytime, this estimated cooling system electrical load can reach up to approximately 1.3 kW. (It should be noted that it is particularly difficult to estimate the air conditioning load during the winter months. The diffuser system still provides approximately 125 L/s of 13°C air to the space, but this cooling is offset by heating from the supply air to the induction unit. The building uses a system of heat pumps to make this process of simultaneous heating and cooling somewhat more

efficient than it may at first appear. But it does make for a difficult challenge to estimate the energy implications of the diffuser system configurations during the winter months. The cooling system electrical load estimation used in this model (Equation 4.3) is only applicable when the induction units are not being used for heating. This is suitable for the case study, which considers only July 6th. However, a more detailed consideration would be necessary to estimate the effects of diffuser control during the winter months.)

To test the accuracy of the model, its temperature and airflow-rate outputs have been compared with data that was collected from the actual system for a week in October 2006. (Note that the verification and calibration was performed only for the temperature and airflow-rate outputs, and not for the estimated cooling energy use.) Temperature and flow-rate values were collected every fifteen minutes by using a program to automatically take screenshots of the system's output screen, and this data was later tabulated. Hourly weather data was collected for the nearest weather station (Ottawa airport), through the weathernetwork.com.

Figures 4.13 and 4.14 show some of the comparisons between the measured data and the model output, when the model received the measured input data. Although unable to exactly mimic the detailed fluctuations in the measured data (it comes close to doing so with the diffuser flow rates, but not at all close with the sensor temperatures), the patterns are similar, and the errors in temperature outputs are generally within 1°C, while the errors in airflow rates are within 5 to 10 L/s but tend to average out fairly well when one considers how much the flow rates fluctuate.

Figure 4.13: Sensor temperature for workstation 97, before automated calibration

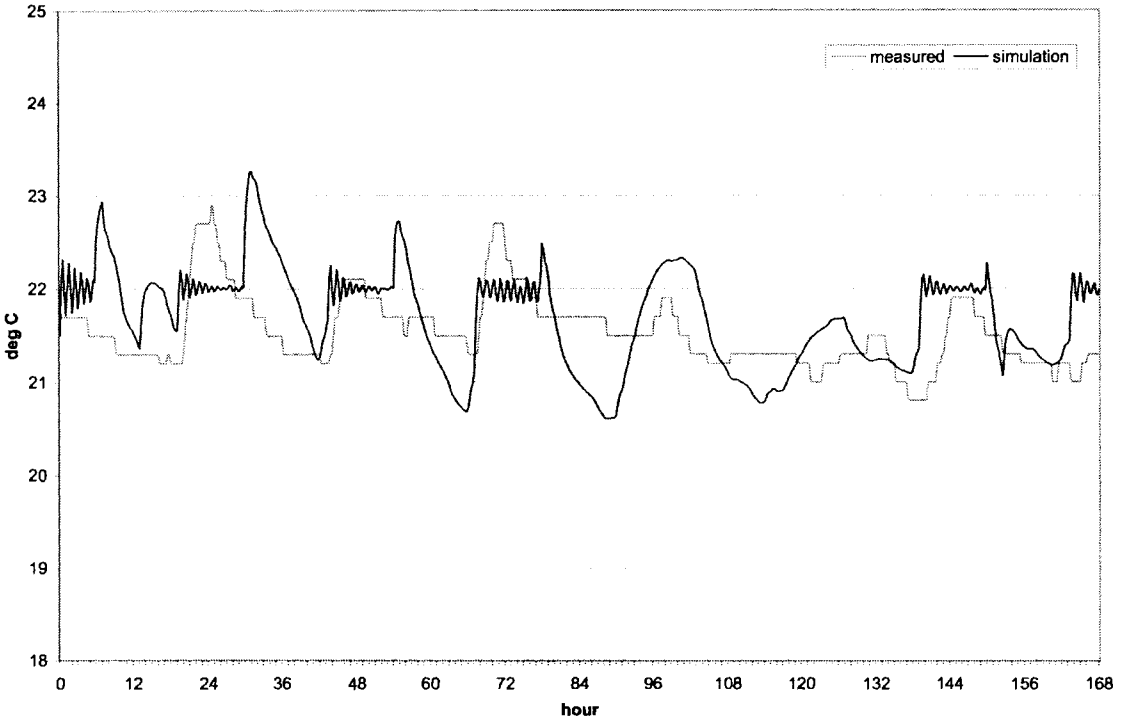
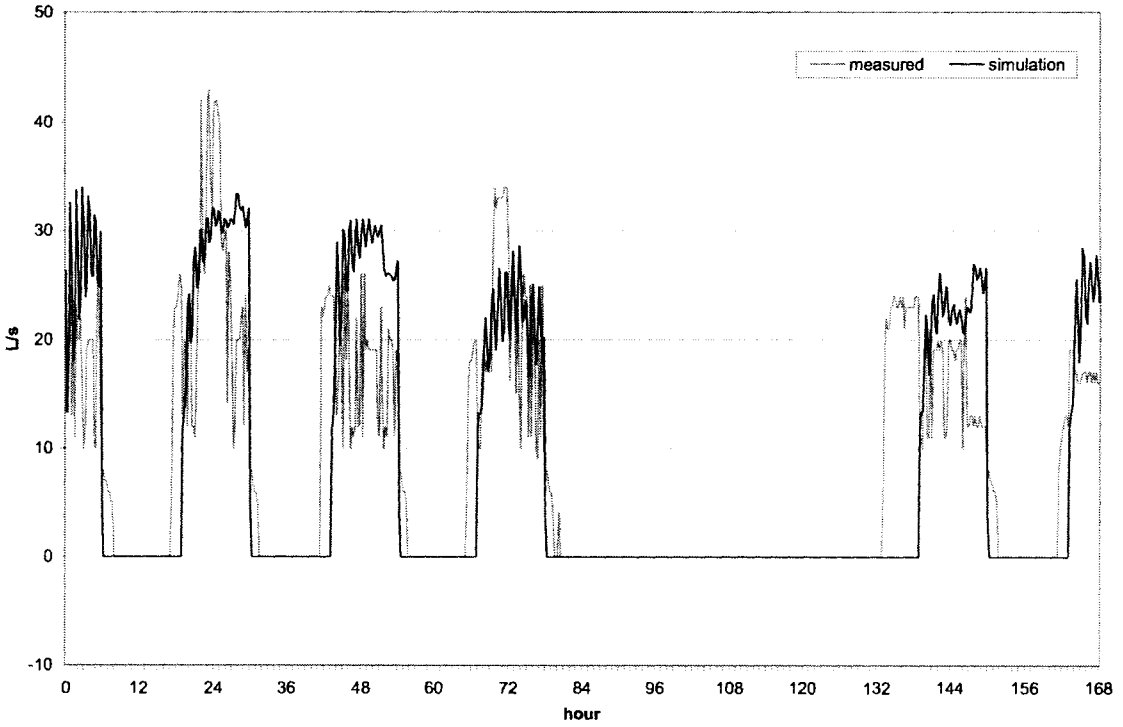


Figure 4.14: Flow rate for workstation 97, before automated calibration



These figures do, however, show that there is some room for the model to improve. As noted in the previous section, there are some aspects of the model's structure that could be improved upon (such as a more detailed consideration of the induction units' temperature modulations), but for the most part, the modules and their input-output structure within the TRNSYS model provide a reasonably accurate representation of the physical phenomena at play in the actual system. But further improvements can be made by making some of the model's parameter values more accurate.

In the process of creating the model, many parameter values have been assigned. Some of these values are fairly certain, others required some judgment. When one is modeling an existing system, calibration against system data is generally an ongoing exercise, as one performs rough checks on model outputs against rules of thumb and against actual system outputs to ensure that the results are in the correct range. But the process does take on a particular character towards the end of the modeling, when the least certain parameter values are tweaked back and forth within a reasonable range for each parameter, until the model outputs reflect the system data to some desired level of accuracy. This is what is generally referred to as 'model calibration'. At this phase in the creation of the model discussed herein, the following seven parameters were found to be both difficult to assign precise values to, and to significantly influence the model outputs:

- The room's thermal capacitance (air and furniture)
- The ratio of solar transmittance
- Infiltration coefficient K1 – the first constant in Equation 4.1
- Induction air supply rate (decreased slightly from the estimated actual flow rate, to consider the effect of the induction unit's cooling coils on the induced air)

- Internal heat gains from general lighting
- Internal heat gains from each occupant's lighting
- Internal heat gains from computers

A reasonable value can be assigned to each of these parameters, but with some uncertainties. The process of model calibration involves tweaking these parameter values within a range (the size of which depends upon the nature of the parameter, and the level of certainty about its value). In most modeling cases, this calibration process is performed manually by iteratively making small changes to the parameter values and re-running the simulation to see the results. But getting more accurate results can require a lot of simulations. After some manual calibration of the model discussed here, the following parameter values were found to produce good results:

- The room's thermal capacitance = 0.1 kJ/hr
- The ratio of solar transmittance = 0.7
- Infiltration coefficient = 0.1
- Induction air supply rate = 0.1 m^3/s
- Internal heat gains from general lighting = 216 kJ/hr
- Internal heat gains from each occupant's lighting = 36 kJ/hr
- Internal heat gains from computers = 36 kJ/hr

These are the values that were used for the outputs shown in Figures 4.13 and 4.14. The average errors between the measured outputs and the simulation outputs, in fifteen minute intervals, are shown in Table 4.3.

Table 4.3: Average error values after manual calibration

Point	Units	Ave Error
T_{sensor} WS93	°C	0.47
T_{sensor} WS94	°C	0.62
T_{sensor} WS95	°C	0.84
T_{sensor} WS96	°C	0.47
T_{sensor} WS97	°C	0.45
Flow Rate WS93	L/s	12.04
Flow Rate WS94	L/s	4.26
Flow Rate WS95	L/s	7.26
Flow Rate WS96	L/s	11.40
Flow Rate WS97	L/s	4.50

Another approach to calibration is to treat it as an error minimization problem and use an optimization algorithm to help find the most appropriate values for these parameters. This automated approach to model calibration is not yet common, but it has been used in a number of studies (Reddy 2006, Sun and Reddy 2006), and it is a promising technique. It is of particular interest in the context of simulation-based control, since an automated calibration process could be employed periodically by the central controller to keep the model updated and to have it continually improve its accuracy. But automated calibration does involve some theoretical and practical challenges, including the fact that when there are many parameters, the optimization problem is usually mathematically underdetermined. Appendix 3 discusses a possible approach to dealing with some of these problems by using uncertainty metrics to help guide the optimization search. The appendix also describes the automated calibration process for the model under study, which found the following set of parameter values to be reasonable (based on the expected value and the level of uncertainty for each parameter) and to produce less error between the simulation outputs and the measured outputs:

- The room's thermal capacitance = 0.11 kJ/hr
- The ratio of solar transmittance = 0.64
- Infiltration coefficient = 0.106
- Induction air supply rate = 0.11 m^3/s
- Internal heat gains from general lighting = 232 kJ/hr
- Internal heat gains from each occupant's lighting = 37.2 kJ/hr
- Internal heat gains from computers = 34.8 kJ/hr

The error metric that was used in the optimization process was based on the squares of the differences between the temperature sensor values and the diffuser airflow rates in the model outputs and the measured data. Given that the objective function is determined by running a simulation, the optimization process is both uncertain (it is able to find values close to the optimum, but is unable to state with certainty whether or not the obtained value is the global optimum), and computationally time-consuming. After 1000 simulations, the parameter values listed above were the best set found, but they produce an error metric which is just 1.7% better than that produced with the values obtained after manual calibration. The automated method does, however, have other benefits, which are discussed further in Chapter 7.

A comparison of model outputs versus measurement (similar to the graphs in Figures 4.13 and 4.14) for the model after the automated calibration technique is shown in Figures 4.15 and 4.16. This is the final version of the model, as used in the studies discussed in the upcoming chapters.

Figure 4.15: Sensor temperature for workstation 97, after automated calibration

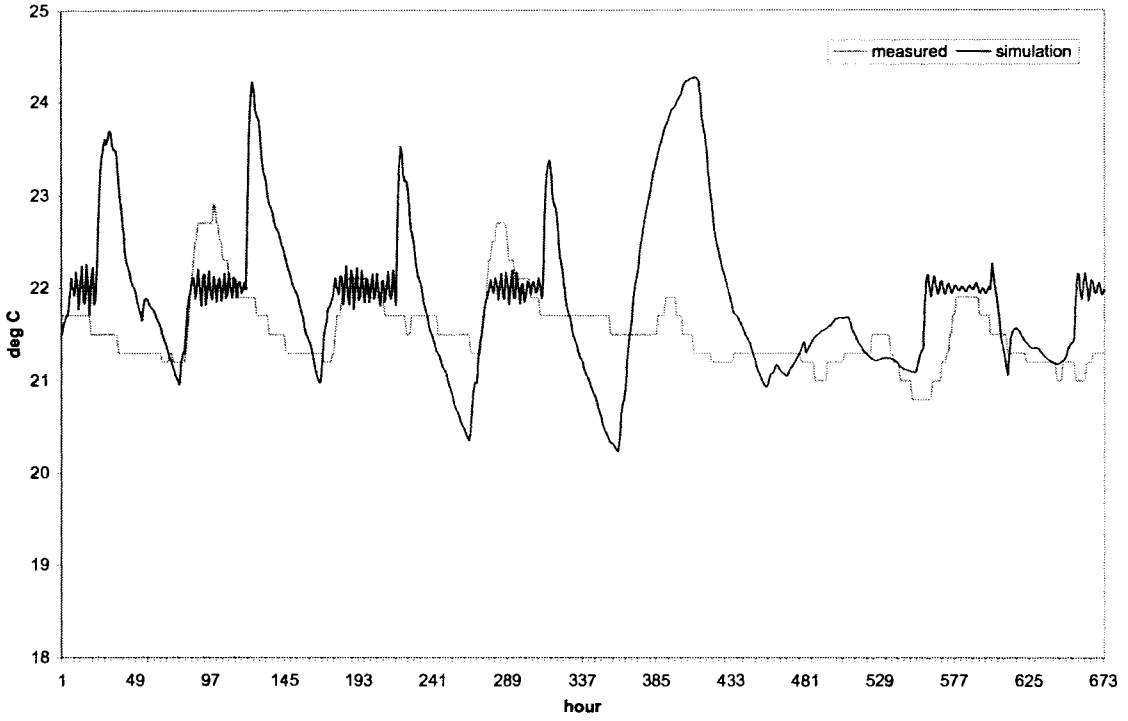
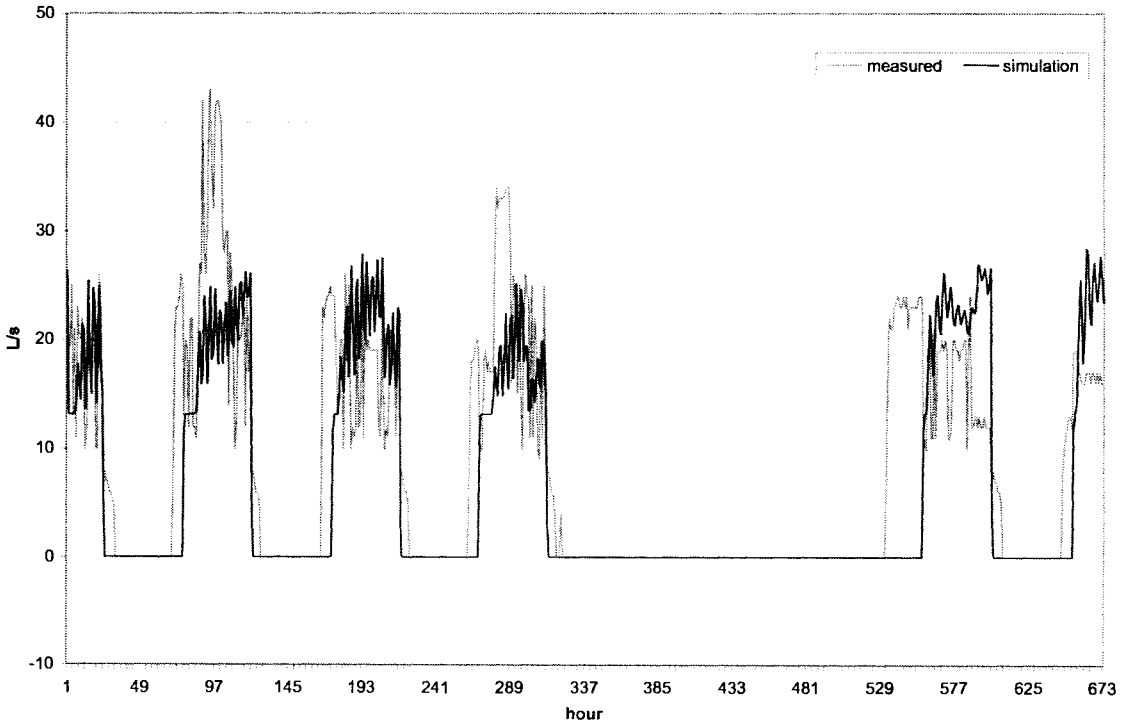


Figure 4.16: Flow rate for workstation 97, after automated calibration



4.4 Demand Response Conditions

In the control studies presented in the next two chapters, the model is used to simulate just one day – the date is July 6 (in the representative-year TMY weather file). It is the hottest day of the year, reaching almost 35°C in Ottawa. The temperature variation over the day is shown in Figure 4.17, and the solar radiation variation is shown in Figure 4.18. All five workstations are occupied, with full lighting and computer use.

Figure 4.17: Outdoor temperature during day of trim

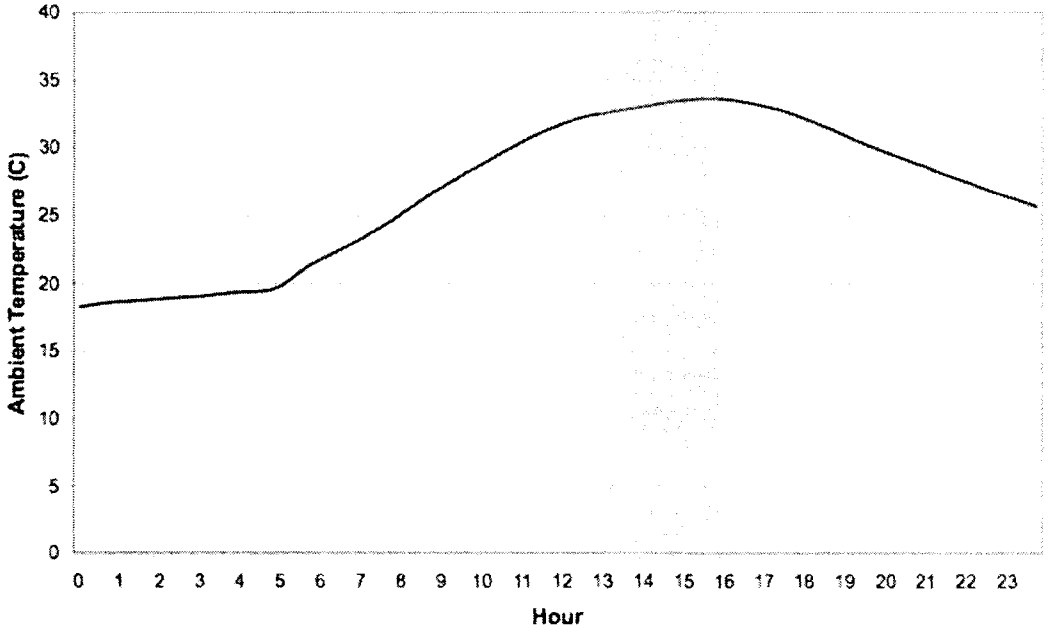
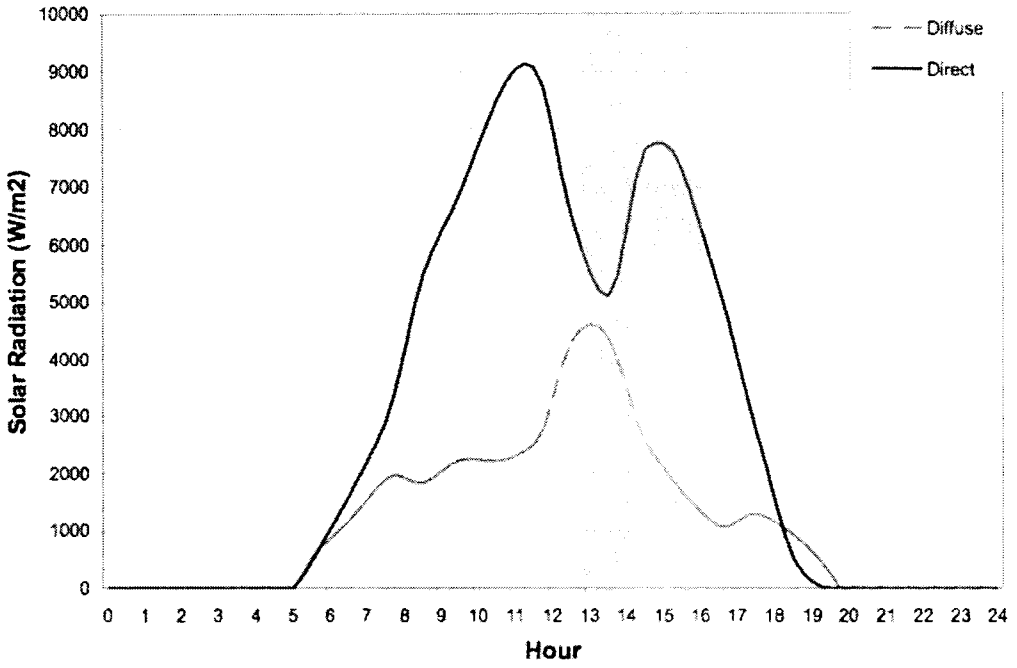


Figure 4.18: Solar radiation during day of trim



The electricity grid is nearing capacity. At noon, it is decided that the building will trim its demand between 1:00 pm and 4:00 pm. (This decision could be in response to increased grid prices, or to a signal from a central grid manager.) The goal of the trim is to minimize the building's peak demand during this three-hour period, where the peak is measured by averaging the demand over 15-minute intervals and finding the maximum among these. The trim is intended to help avoid blackout, so it is only as good as its worst 15-minute interval. After 4:00pm, the peak grid stress is over, and we are not concerned with the rebound in demand that may happen afterward. The trim is carried out by overriding the occupant set-points in the PEC system. The maximum set-point allowed is 26°C, and the minimum allowed is 22°C. At noon, the occupants' temperature set-points are all set to 22°C. The set-point changes can begin at 12:30 pm, providing the system time to respond before the trim measurements start. As noted in the introduction, the base

case, without any demand response action, is as shown in Figures 1.1 and 1.2. The base case maximum demand is 1301 W.

Note the use of the diffuser airflow-rate set-point in Figure 1.1 (and in the related graphs throughout the next two chapters), the diffuser airflow rate will vary slightly around this set-point, but will remain very close to it. Also note that in response to the large loads, the airflow-rate reaches its maximum and is unable to maintain the sensor temperature set-point exactly (the temperature rises very slightly between 2 pm to 4 pm).

4.5 Summary

This chapter has described the office space and ventilation system considered in the case study, along with the TRNSYS model of it, and provided some discussion of the verification and calibration of this model against measured data. This model is used in the demand response control studies presented in the next two chapters, both as the model within the simulation-based supervisory controller (for the results in Chapter 6), and as the 'actual building' model in the virtual test environment.

Chapter 5

Results: Rule-based approaches

Before considering the results from the simulation-based control studies in the next chapter, three rule-based strategies are considered. This provides some basis for comparison, and also informs the knowledge-based starting points used in some of the cases with the simulation-based controller. We look first at a jump trim, then a linear trim, followed by the logarithmic trim as suggested by Braun and Lee (2006a).

5.1 Jump trim

The simplest approach to trimming demand by changing the occupants' temperature set-points is to increase them immediately to the maximum allowable value, in this case 26°C, and maintain them at this value for the length of the trim. With the PEC system model and the given weather conditions, the results of this approach are shown in Figures 5.1 and 5.2.

Figure 5.1: Jump trim

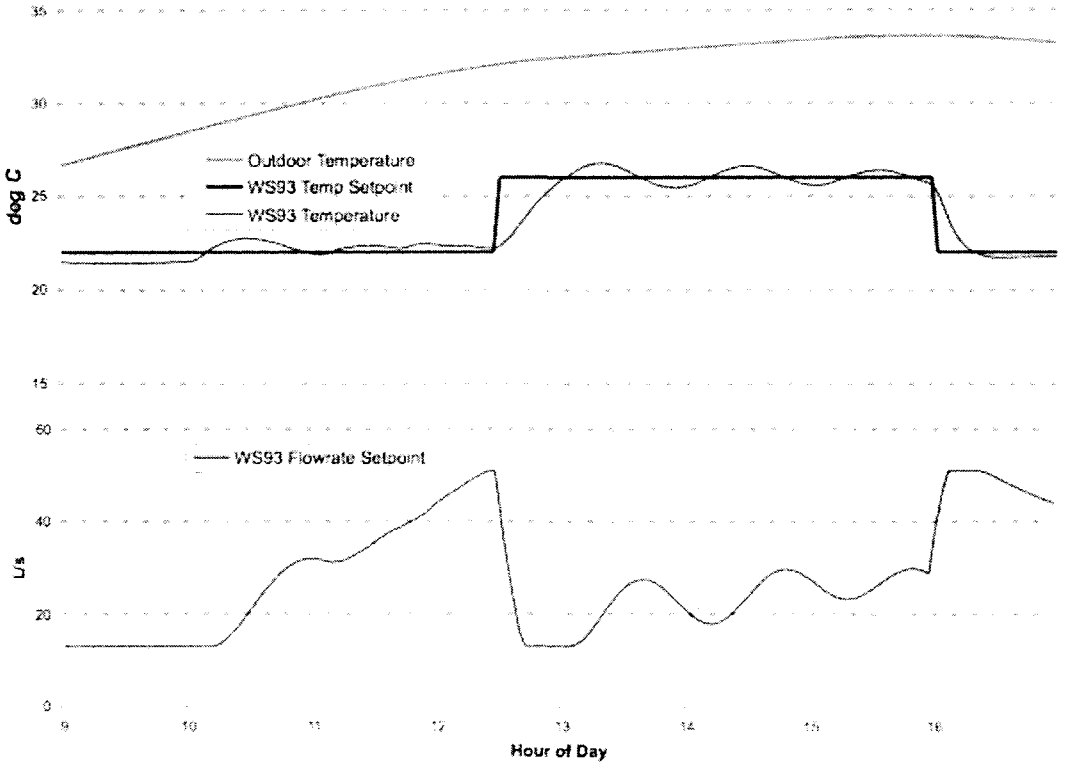
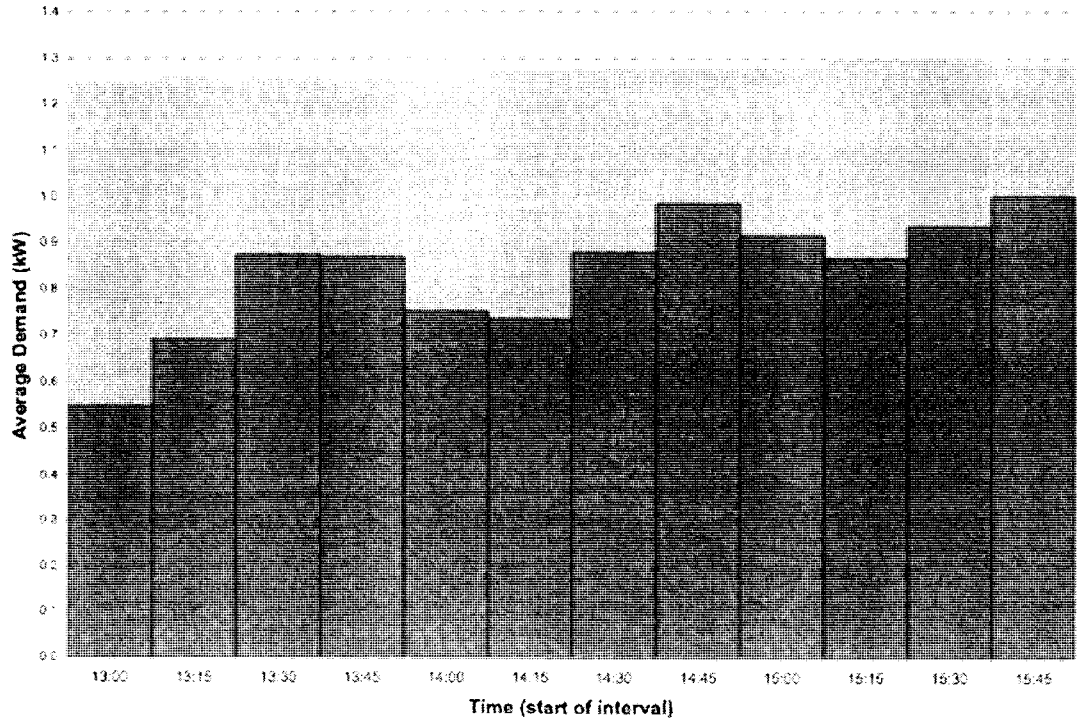


Figure 5.2: Demand curve with jump trim

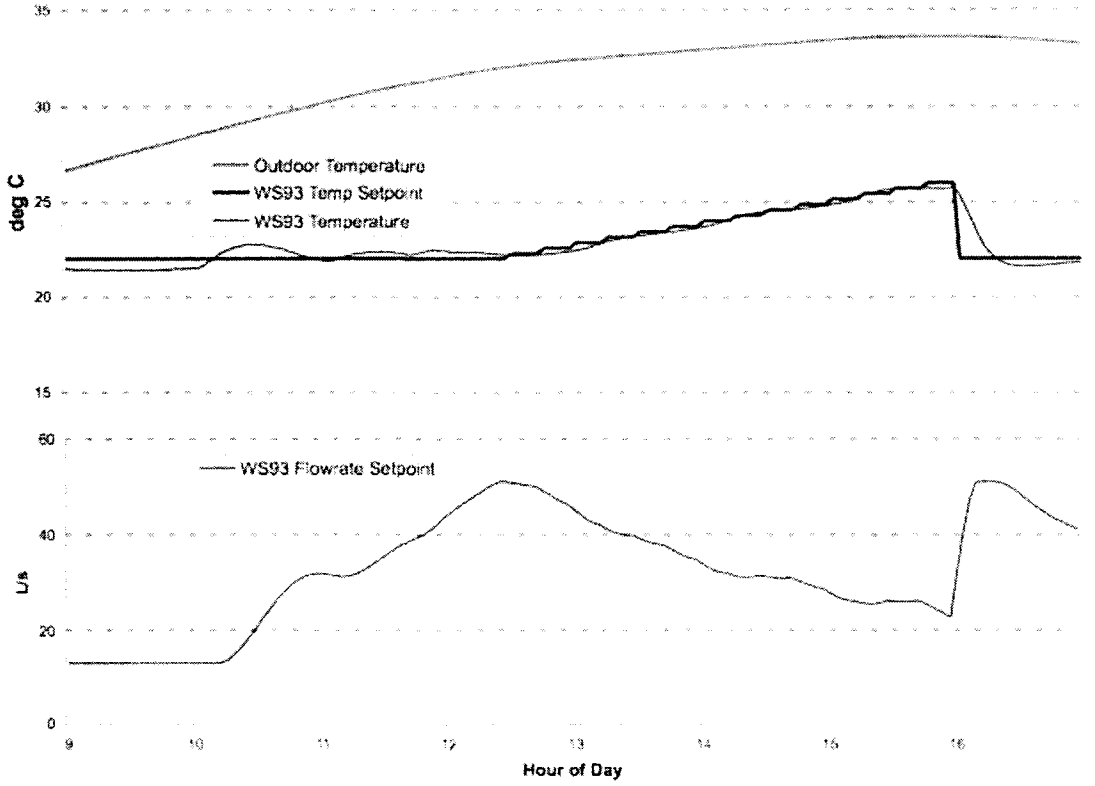


As shown in Figure 5.2, the electricity demand by the PEC system drops off quickly to produce a very low demand during the beginning of the trim period, but then it increases over the course of the following three hours. The maximum demand for the period is during the last fifteen-minute interval, and is equal to 1001 W. This represents a 23.03% trim when compared with the status-quo case. This control approach is simple and fairly effective, but the results suggest that a better trim might result from ramping up the temperature somewhat instead of just jumping from 22°C to 26°C at the beginning of the trim period.

5.2 Linear trim

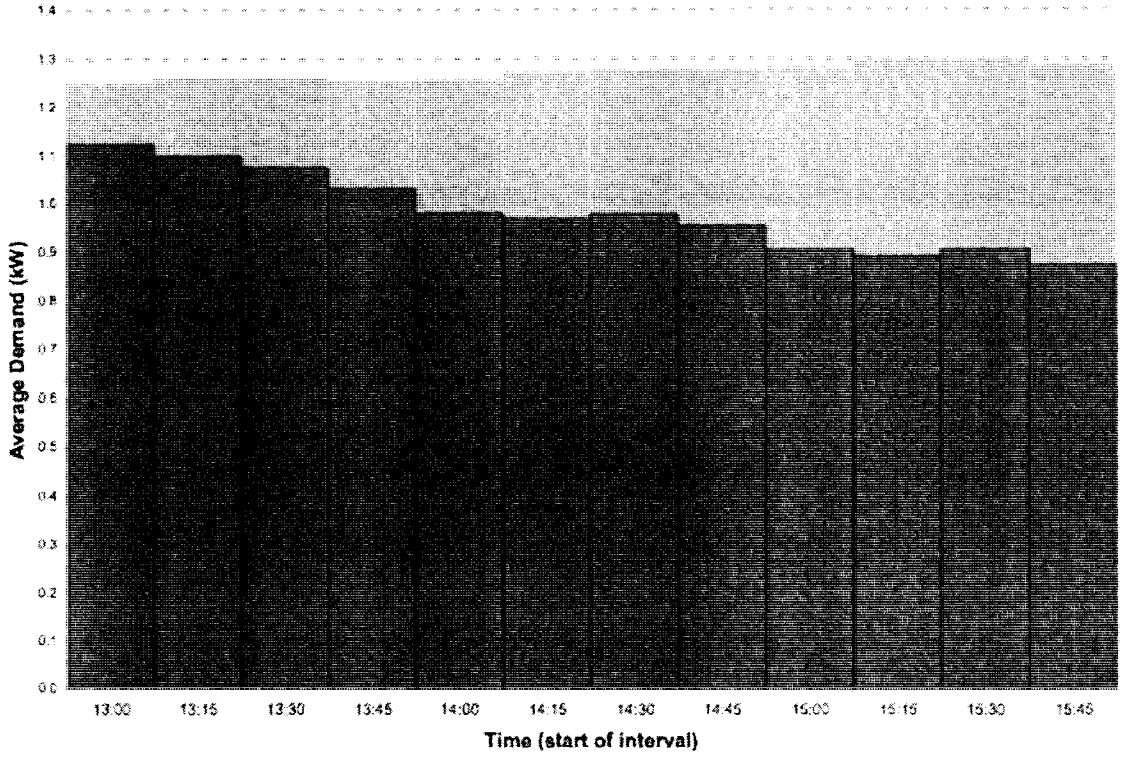
Figure 5.3 shows the results of using a simple ramping approach, which increases the temperature linearly from 22°C to 26°C between 12:30 pm and 4:00 pm.

Figure 5.3: Linear trim



This strategy helps to decrease the demand towards the end of the trim period, as shown in Figure 5.4, where the demand during the last 15-minute interval is much less than during the 'jump trim' case. But the linear ramping does not produce enough of a decrease in demand during the beginning of the trim period, so the maximum demand occurs during the first 15-minute interval, at 1123 W. This represents just a 13.69% trim.

Figure 5.4: Demand curve with linear trim

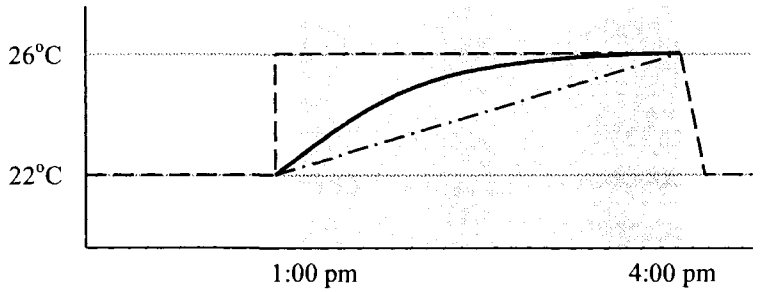


These results suggest that a ramping strategy might be more successful if it ramped the temperature set-point up more aggressively at the beginning of the trimming period, and then less aggressively at the end.

5.3 Logarithmic trim

The 'jump trim' approach resulted in too much demand reduction at first, at the expense of demand later in the trimming period, whereas the 'linear trim' approach did not provide enough of a demand reduction at the beginning of the trimming period. A more complex approach might thus consider using a multi-stage ramping, or some type of smooth ramping curve, as shown in Figure 5.5.

Figure 5.5: Complex temperature ramping

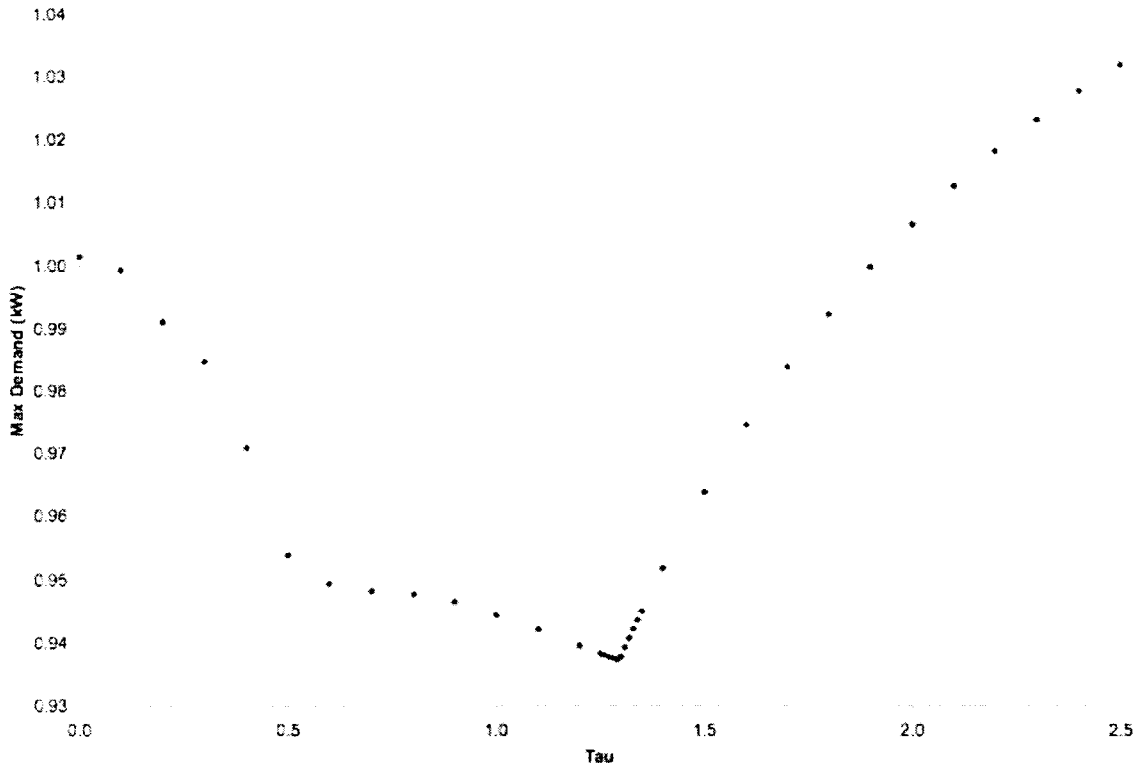


As discussed in Chapter 2, Lee and Braun (2006a, 2006b) have shown that a good control strategy for demand trimming with temperature set-point modulation is to ramp the temperature up according to Equation 5.1, where t is the length of time since the beginning of the trim, κ is the full length of the trim period, and τ is a system-specific constant that determines the shape of the curve. As τ approaches zero, the equation approximates a jump trim, and as τ approaches infinity, the equation approximates a linear trim.

$$\frac{T_{set} - T_{min}}{T_{max} - T_{min}} = \frac{1 - \exp(-t/\tau)}{1 - \exp(-\kappa/\tau)} \qquad \text{(Eq 5.1)}$$

To use this approach, an appropriate value must be used for τ. Because this case study is using a detailed model rather than a actual building, the best value for τ was found simply by testing various values of τ, as shown in Figure 5.6, and using the best one, rather than adopting one of the approaches outlined by Lee and Braun (2006a). The optimal value (to the desired level of precision) for τ was found to be 1.29.

Figure 5.6: Finding the optimal value for τ ('Tau')



With this value of τ, the ramping equation produces very good results when tested with the PEC model. The results are shown in Figures 5.7 and 5.8. The maximum demand is 937 W, which is a 27.97% trim compared with the status-quo case. This is a 4.67% greater trim than the 'jump trim' case, and a 14.28% greater trim than the 'linear trim' case.

Figure 5.7: Logarithmic trim

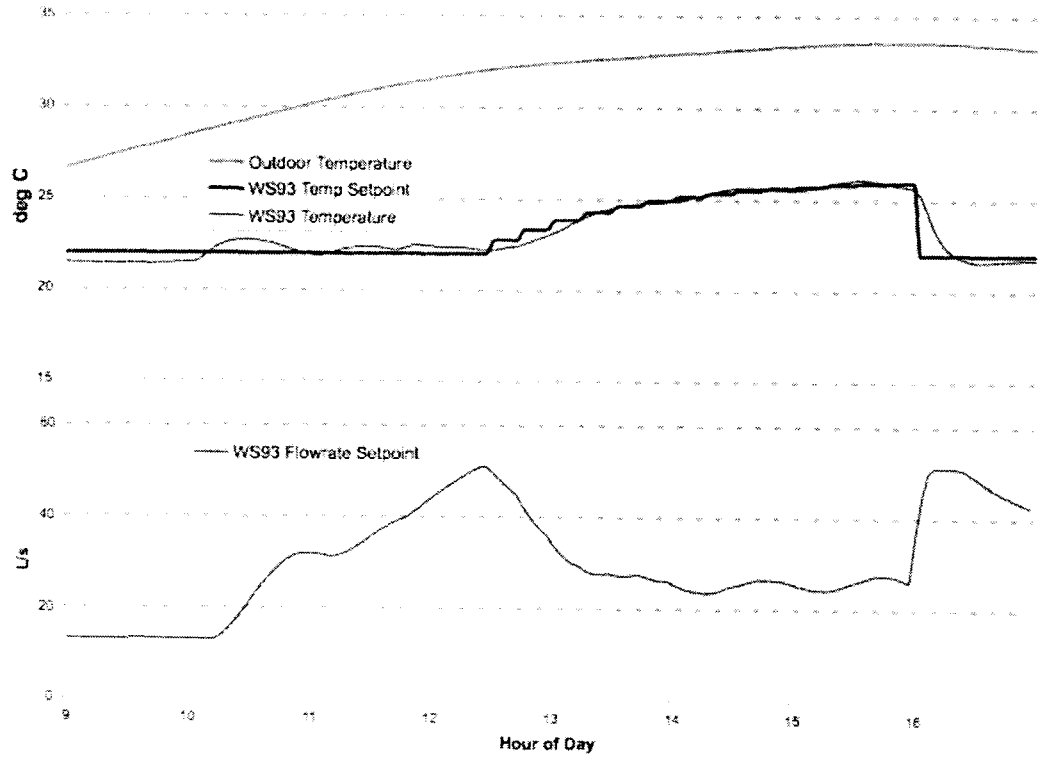
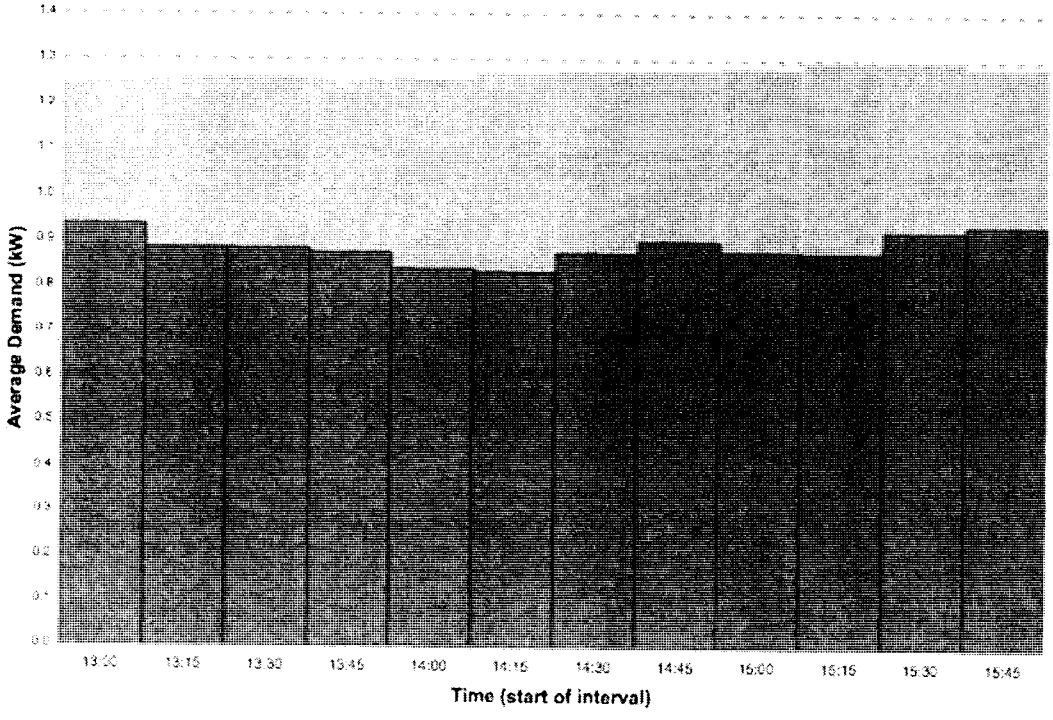


Figure 5.8: Demand curve with logarithmic trim



Note how close the results in Figure 5.8 are to forming a straight line. A truly optimal ramping strategy would result in a constant demand across the trimming period. The logarithmic approach comes very close to producing that optimal result.

5.4 Summary

Table 5.1 provides a summary of the rule-based temperature-ramping strategies evaluated here with the PEC model. The 'jump trim' strategy performs quite well given its simplicity. The 'linear trim' strategy provides some trimming, but not nearly as much as the other two strategies. The 'logarithmic trim' strategy is the most effective.

Table 5.1: Summary of results with rule-based demand trim strategies

	Max Demand (W)	Trim Percentage
Base Case	1301	-
Jump Trim	1001	23.03%
Linear Trim	1123	13.69%
Log Trim	937	27.97%

A truly optimal temperature ramp is particular to both the building system under consideration and the weather conditions. But it is unlikely to perform much better than the 'logarithmic trim' discussed above.

Chapter 6

Results: Simulation-based approaches

Using the results of the rule-based approaches in Chapter 5 as a baseline for comparison, this chapter evaluates the performance of simulation-based control under a variety of circumstances, with the software framework and modified genetic algorithm described in Chapter 3 and Appendix 1. Ideal circumstances are considered first, helping to approximate an upper bound for the performance of simulation-based control for this particular building system and demand-response scenario. Two non-ideal circumstances are then considered: imperfect model calibration, and imperfect initialization and prediction. These investigations of non-ideal conditions rough out the magnitude of the impacts of these imperfections on possible real-world applications of simulation-based control, and demonstrate how the software framework and virtual testing arrangement can be used for more detailed studies of these impacts in future research.

An important realization that came about during the course of this research was just how important proper model initialization and good predictions are for the performance of simulation-based control. Any inaccuracies in the conditions being fed to SimCon mean that it is solving an optimization problem that is different than the reality it is intending to solve for, so what it finds for its optimal solution may be different than what is optimal in reality. And in cases like our current case study, where there is not much room for improvement over a rule-based approach, this effect of inaccuracies can become quite pronounced.

Model initialization is a particularly challenging concern for SimCon, because of the nature of most simulation tools. In most cases, the user cannot specify point-in-time values for many parts of the model, including temperatures of the thermal mass. Most simulation programs instead rely on a warming-up period, where somewhat arbitrary values for these unknown initial conditions are used at the beginning of the warm-up period, which uses weather data similar to what would have happened in the period before the start of the simulation. The idea is that if the warm-up period is long enough and/or if the values provided at the beginning of the warm-up period are nearly accurate, then the values at the end of the warm-up period (which is the beginning of the actual simulation) are good approximations of the initial values. This approach does not cause any particular problems for simulation-based design optimization. But it does cause a challenge for simulation-based control, since it must deal with longer computation times and/or specify accurate conditions for the warm-up period.

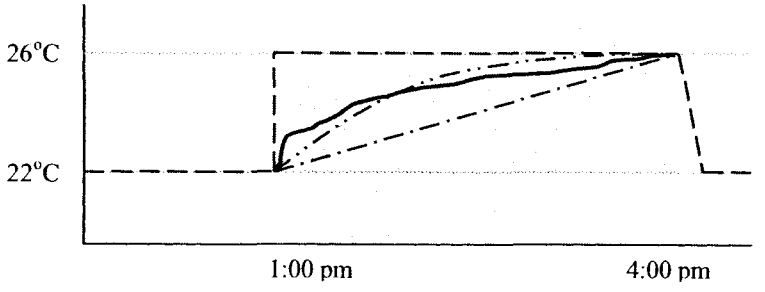
With this in mind and with the desire to study the effects of imperfect initialization and prediction, two slightly different versions of the setup are considered here. The first uses an identical warming-up period and predictions for the 'simulated building' model as is used for the 'actual building' model. This represents the ideal case. The second uses a shorter warming-up period with slightly different warming-up conditions, and thus has slightly different initial conditions, and also uses slightly inaccurate predictions. The length of the warming-up period, the warming-up conditions, and the predictions can all be modified, allowing investigation of the impacts of inaccuracies.

Section 6.2 considers the ideal case, first without any computation-time constraints, and then with the 15-minute optimization time constraints inherent in the problem definition. Section 6.3 also uses perfect initialization and prediction, but considers the effects of poor model calibration by using different parameter values in the 'simulated building' model than are used in the 'actual building' model. Section 6.4 then furthers the discussion of warm-up times and initial conditions, and evaluates the impacts of imperfect initialization and prediction.

6.1 Optimization problem definition for simulation-based supervisory control

The goal of the simulation-based supervisory control strategy for this problem is to come as close as possible to producing a truly optimal temperature ramp for the system. This section looks at how the problem is set up within the simulation-based control software framework.

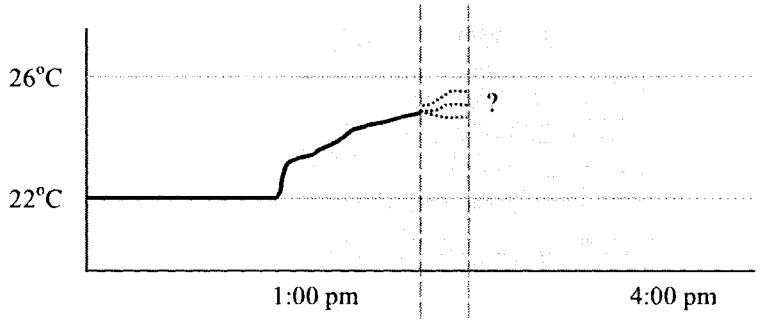
Figure 6.1: Possible shape of optimal ramping for this system and conditions



With the rule-based approaches described in the previous section, the temperature set-point followed a pre-defined ramping path during the trim period. However, with simulation-based supervisory control, the path of the temperature set-point is not known

at the outset. At each time-step, the temperature set-point for the next time-step is determined by the supervisory controller, based on the results of simulation-based optimization.

Figure 6.2: Each time-step presents a new decision to be made by the control system



This decision is based on the current state of the system and the predicted future conditions. And in order to effectively determine the set-point for the upcoming time-step, the supervisory controller must also postulate what the set-points will be for future time-steps. The resulting optimization problem is shown in Figure 6.3.

Figure 6.3: Optimization problem faced at each time-step

<table>
<tr><td>minimize [max elec demand] = f(C,S), where f(C,S) approximated by simulation</td></tr>
<tr><td>C = { [weather conditions], [initial system state conditions], [other conditions] }

<table><tr><td>[weather conditions]</td><td>Ambient temperature (from current to 3.5 hours in future)
Direct solar radiation (from current to 3.5 hours in future)
Diffuse solar radiation (from current to 3.5 hours in future)
Etc.</td></tr><tr><td>[initial system state conditions]</td><td>Initial zone temperature
Initial temperature set-point
Initial diffuser flowrate
Etc.</td></tr><tr><td>[other conditions]</td><td>Number of hours remaining in demand trim</td></tr></table></td></tr>
<tr><td>S = { temperature set-points for the upcoming 3.5 hours }</td></tr>
</table>

In this case study, a 15 minute time-step is used by the supervisory controller module. So at 12:15pm, the supervisory controller begins the calculations necessary to determine the set-point value for 12:30pm to 12:45pm. And at 12:30pm it begins the calculations to determine the set-point value for 12:45pm to 1:00pm.

It is worth noting the size of the possible solution space in Figure 6.3. Since each of the set-points s_i in S are continuous variables, the search space is infinite in theory. But even if we limit ourselves to consideration of a single decimal place for each set-point, we still have 40 possible values for each set-point, and there are 14 set-points in S (since there are four set-points per hour, over 3.5 hours), so there are $40^{14} = 2.68 * 10^{22}$ possible configurations of S. If we consider two decimal places of precision, the size of the search

space is 2.68 * 10^{36}. This provides a serious challenge for the simulation-based optimization algorithm, since it can only run approximately 400 simulations of the PEC model within the given 15 minute window.

6.2 Demonstration example with a simple parametric algorithm

To demonstrate the use of SimCon, a very simple algorithm has been set up for use in GenOpt, as embodied in the 'command file' in Figure A1.8 in Appendix 1. This uses a minor variation of the EquMesh algorithm, which is a standard algorithm within the GenOpt library and is used for parametric studies – it simply creates a grid over the search space and tests every point on the grid. (The only change made to the original algorithm was so that it reported the minimum point at the end of its parametric study.) The algorithm only tests two values for each of the first eight set-point variables, 23°C and 25°C, and just one value (25°C) for each of the last 6 set-point variables. So there are $2^8 = 256$ permutations that are tested by the algorithm, which requires approximately 9 minutes of computation time.

6.2.1 Single SimCon run using a simple parametric algorithm

The conditions used in this single control-time-step example SimCon run are shown in Table 6.1. These conditions data were written to text files in the appropriate manner, and SimCon was called – it set up the optimization problem for that timestep, ran GenOpt, processed the results, and output the control decision of 25°C. This can then be used by the central control system in the building (or virtual building, in these studies), setting the temperature set-point value to 25°C for the upcoming 15-minute interval. It

should be noted that in the process, SimCon also calculated the optimal value for each time-step in the future horizon (all of which were 25°C in this case), and wrote this full result to memory so that it might be used to inform future SimCon runs (if the algorithms are configured to take advantage of this).

Table 6.1: Conditions used in example SimCon run

Previous-time-step (warm-up) and Current Conditions						
Time-step	*Tamb*	*Direct*	*Diffuse*	*Tset*	*FRset*	*Tzone*
Current	33.14	2477	586	22	0.0352	22.5
Previous	32.96	2185	961	25	-	-
2-before	32.71	1722	1278	25	-	-
3-before	32.46	1754	1410	23	-	-
4-before	32.15	2036	1359	25	-	-
5-before	31.65	2448	1118	22	-	-
6-before	31.16	2418	751	22	-	-
Predicted Future Conditions						
Next hour	33.50	2451	587	-	-	-
2-after	33.60	2096	426	-	-	-
3-after	33.57	2096	426	-	-	-
4-after	33.36	2096	426	-	-	-
Other Conditions						
Time remaining in trim = 1.5 hrs						

6.2.2 Example two-model run

The example in the previous section was just for a single time-step. But to test the simulation-based control configuration, it is better to run SimCon within a two-model virtual testing environment, as described in Chapter 3. Two copies of TRNSYS are used on the same computer. One version of the PEC model is slowed down and acts as the 'actual' building, and it has a supervisory control module that calls SimCon. Another version of the PEC model is used as the model within the SimCon framework.

Figures 6.4 and 6.5 show the results (from the perspective of the 'actual' building) of a sample run with the two-model approach, where SimCon was configured with the

simple parametric algorithm used in the previous section. The same scenario and conditions were used as with the tests of the rule-based approaches. Since the GenOpt algorithm was only choosing between 23°C and 25°C for the set-point values, SimCon output 25°C to the 'decisions' file at each time-step that it was asked to determine a set-point, except for the two time-steps near the beginning of the trim, where it was set to 23°C. The result was that the maximum demand was 1243 W, which represents a 4.44% trim. This is slightly better than the base case, but much worse than the jump trim and the logarithmic trim, and certainly not optimal.

Figure 6.4: Example run with two-model approach with SimCon

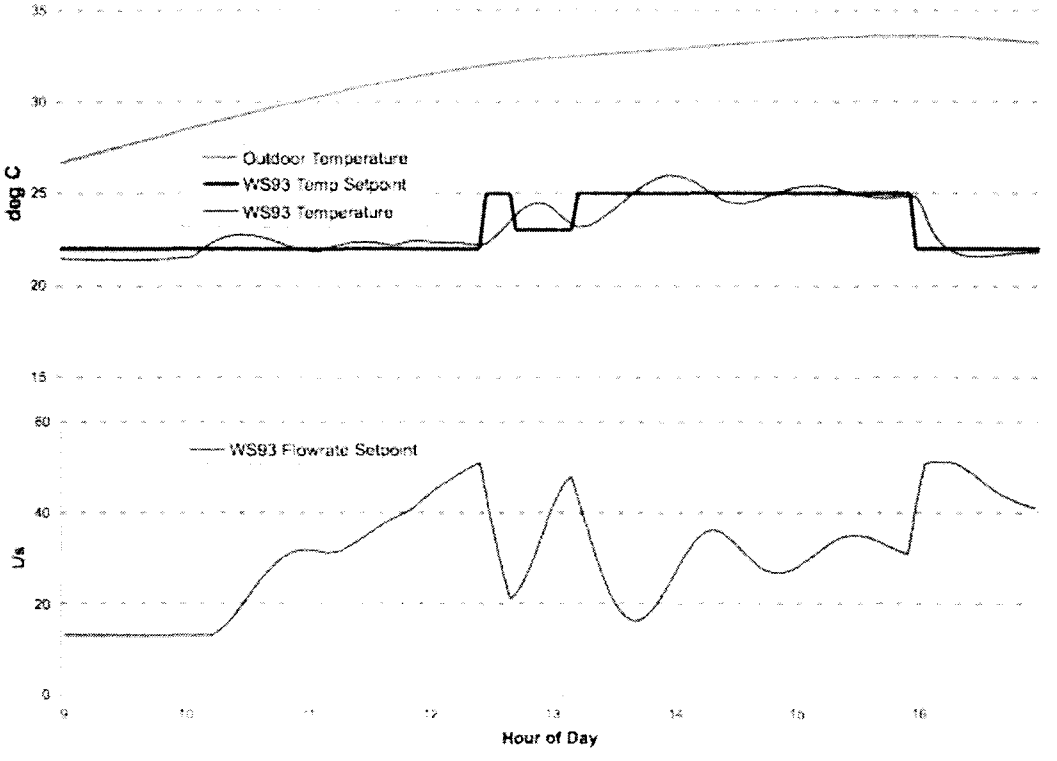
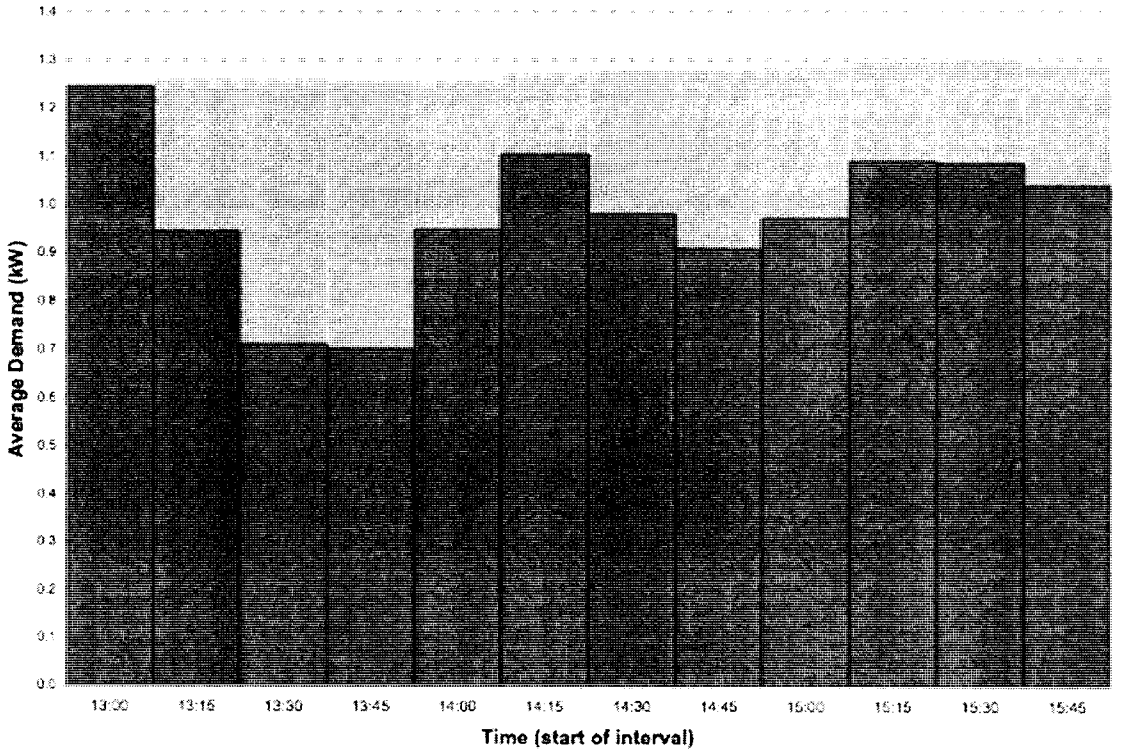


Figure 6.5: Example demand curve (using simple parametric algorithm)



The simulation-based control approach does not guarantee optimal results, nor does it even guarantee better results than rule-based approaches. How well it performs depends, to a great extent, on the optimization algorithms used. The remaining sections of this chapter discuss results with the modified genetic algorithm. Other possibilities are discussed in Chapter 7, which points towards further development.

6.3 Ideal Case – Perfect model, initialization and prediction

The logarithmic trim discussed in Chapter 5 produced results that were very near optimal, which leaves very little room for improvement through the use of simulation-based control. The logarithmic trim produces a maximum demand that is 27.97% below the base case. Section 6.3.1 shows that the optimal trim is better than this by a couple of

percent, and section 6.3.2 shows that the ideal operation of SimCon with the modified genetic algorithm, within the time-constraints of the problem, produces a trim that is better than the logarithmic trim, but slightly worse than the optimal.

6.3.1 With no computational constraints

In an attempt to approximate the truly optimal set of set-point values for this problem, the time-constraints were dropped and the modified genetic algorithm was allowed to run over a number of nights. The resulting set of set-points is shown in Figure 6.6, and the demand map is shown in Figure 6.7. Although there still does seem to be some room for improvement (since the demand line is not perfectly straight), it is getting very close to the optimal. It is interesting to note the variations from the logarithmic trim: it raises the set-point faster at first in order to decrease the demand during the 1:00pm time-step, which was one of the two high points in the logarithmic trim, and then decreases it in order to flatten out the demand and thus improve the later times of the trim. This configuration produces a maximum demand of 909 W, which is a 30.12% trim.

Figure 6.6: Ideal case, no computational constraints – set-point control

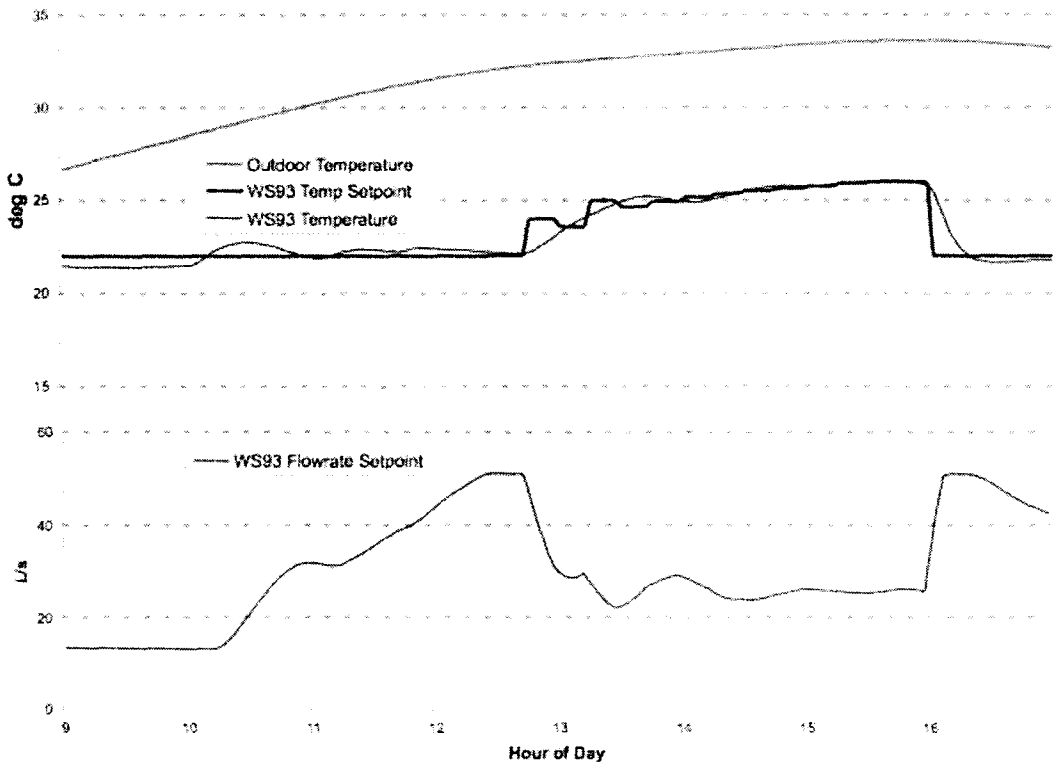
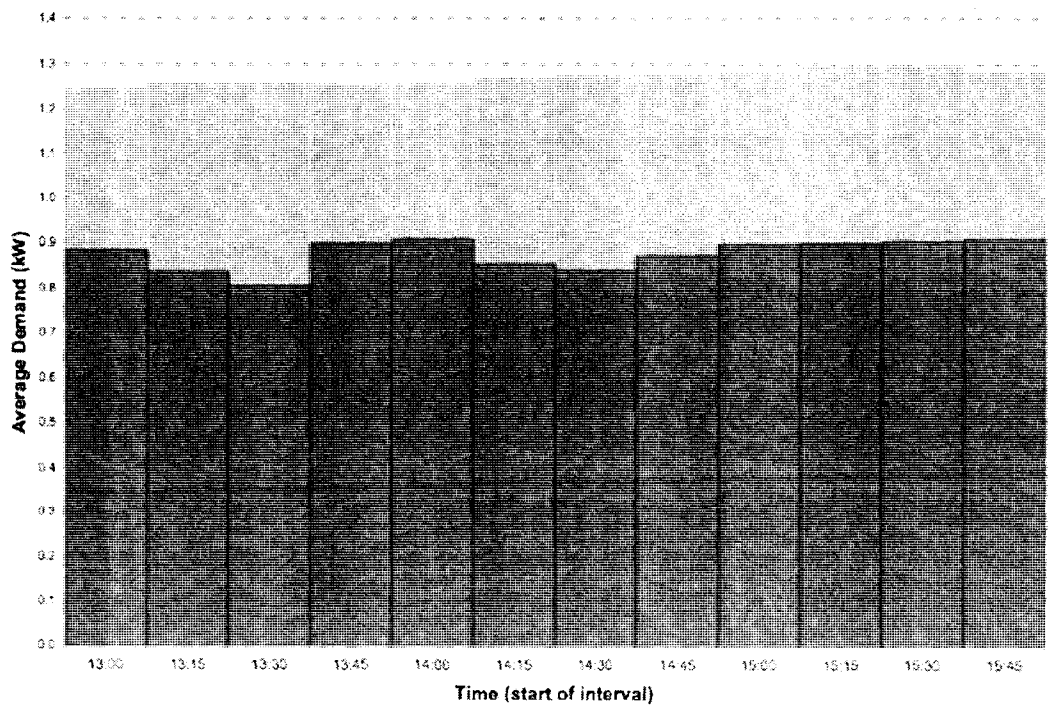


Figure 6.7: Ideal case, no computational constraints – demand map



6.3.2 With time constraints

With the time constraints inherent in the problem, it is expected that SimCon would not perform as well as the ideal case without computational time constraints. However, with perfect model calibration and perfect prediction and initialization, along with rule-based starting-point suggestions based on the logarithmic rule, it can also not do any worse than the logarithmic case (since the optimization algorithm can do no worse than its initial points). And as expected, the application of SimCon did produce a result between these two bounds. It produced a relatively significant improvement on the logarithmic case, with a 29.46% trim (maximum demand of 918 W), which is 1.49% better than the logarithmic case. The genetic algorithm parameters used here are: maximum 400 simulations, population size of 15, an elite ratio of 0.2, a mutation ratio of 0.2 and a crossover ratio of 0.8.

Figure 6.8: Ideal case, with time constraints – set-point control

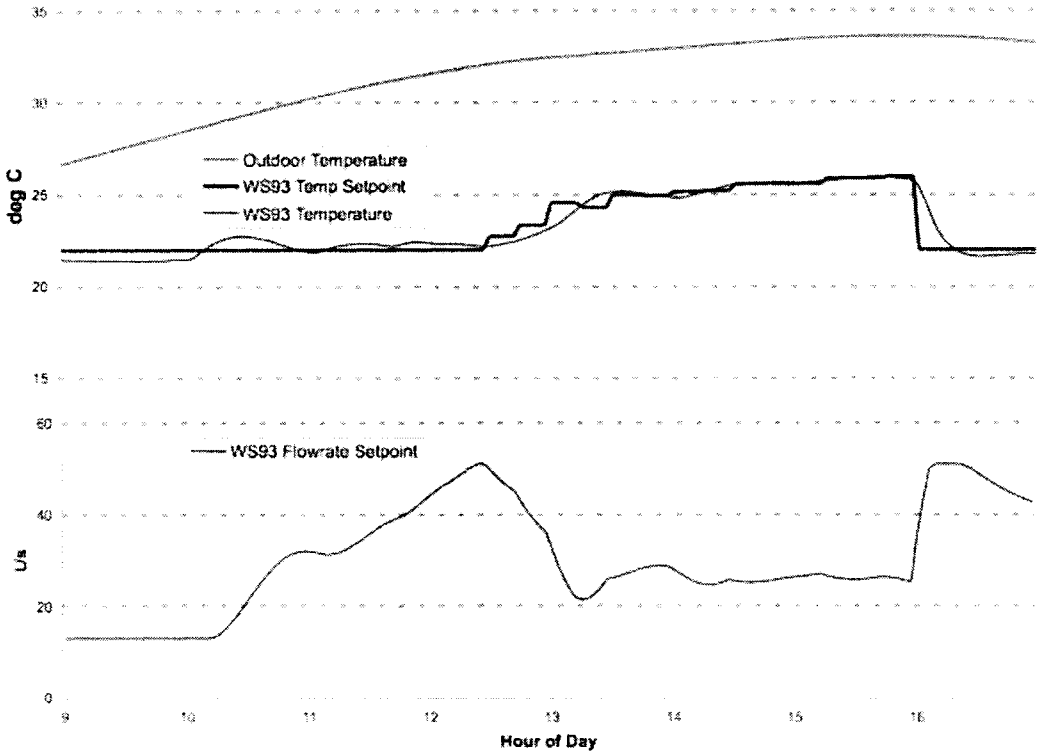
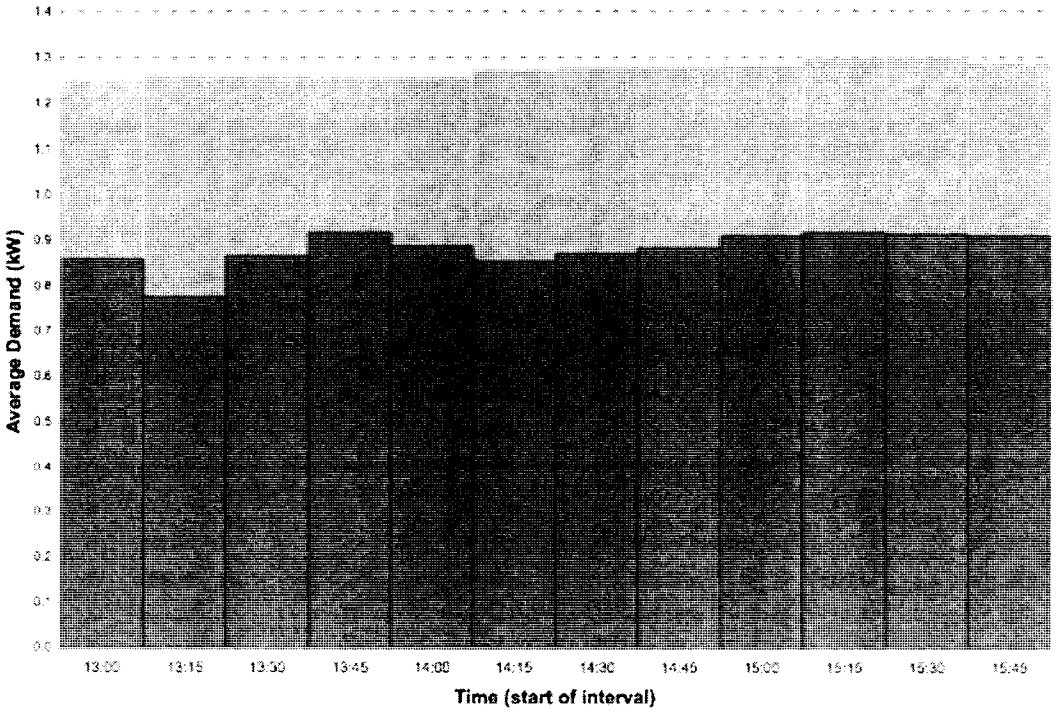


Figure 6.9: Ideal case, with time constraints – demand map



It should be noted that since genetic algorithms are stochastic, slightly different results would be obtained each time it is run, so on average the results may be slightly better or slightly worse than the results shown here. The results will, however, always lie between the lower bound, provided by the logarithmic trim results, and the theoretical upper bound approximated by the no-time-constraints optimization. This lower bound is dependent on the use of suggestions from previous time-steps and using logarithmic rule-based suggestions for starting points. If different rule-based suggestions were used (e.g. a linear ramp), then the lower bound of SimCon's results would be defined by those rules, and it would not perform as well on average as it would with better rule-based suggestions. And if it was not making use of suggestions from previous time-steps, then it is possible for it to perform worse than the rule-based suggestions – for example, one time-step's optimization may find a good configuration that uses an unexpected set-point for the upcoming time-step and relies heavily on a future set-point for this configuration to work out properly, if that information is not passed forward then that important future set-point may not be found during the optimization process on that time-step.

In general, the performance of the simulation-based control configuration studied here suffers without the use of learning or rule-based suggestions. Figures 6.10 and 6.11 show the results of a test with logarithmic starting points but no learning, which produced a 28.04% trim (only very slightly better than the logarithmic case). Figures 6.12 and 6.13 show the results of a test without rule-based starting-points, nor learning from previous time-steps, nor any rule-based constraints, so the GA just used random initial configurations from across the search space. It performed much worse than the other cases, producing just a 5.91% trim.

Figure 6.10: Ideal case, time constraints, no learning – set-point control

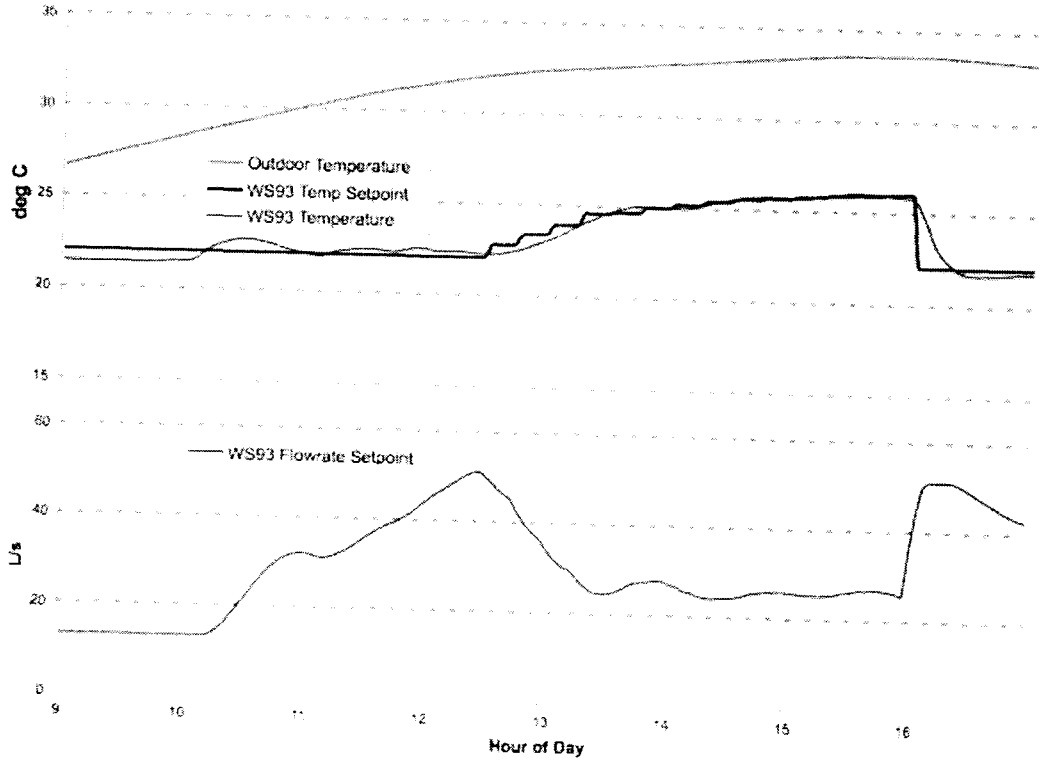


Figure 6.11: Ideal case, time constraints, no learning – demand map

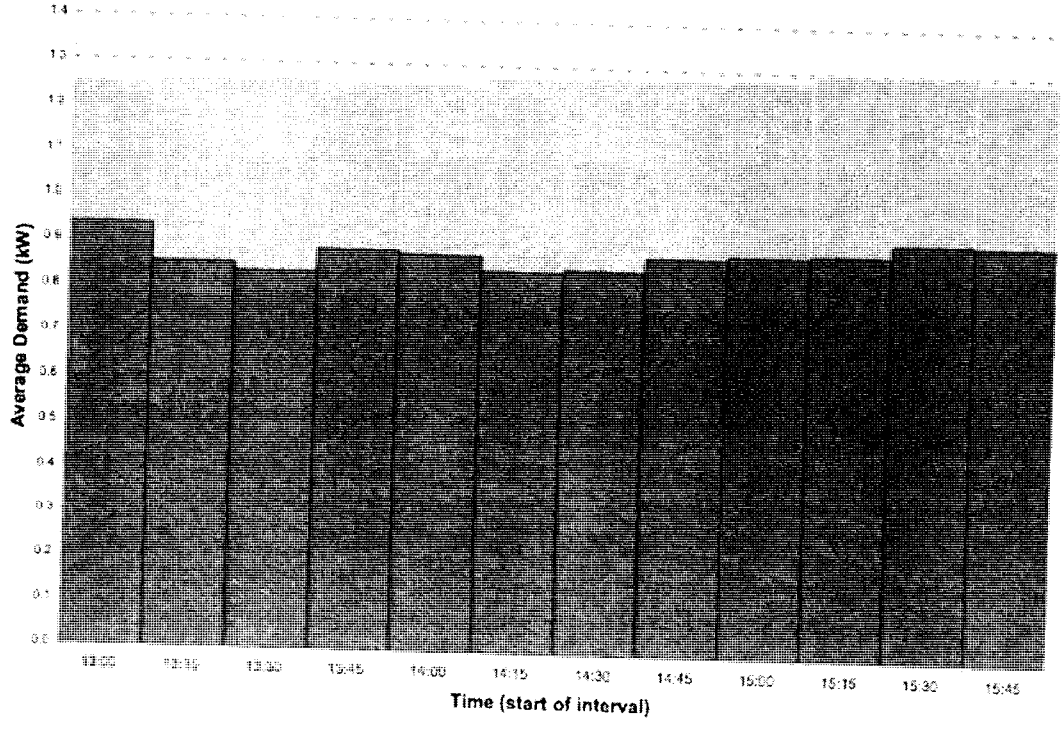


Figure 6.12: Ideal case, time constraints, no rules, learning or constraints

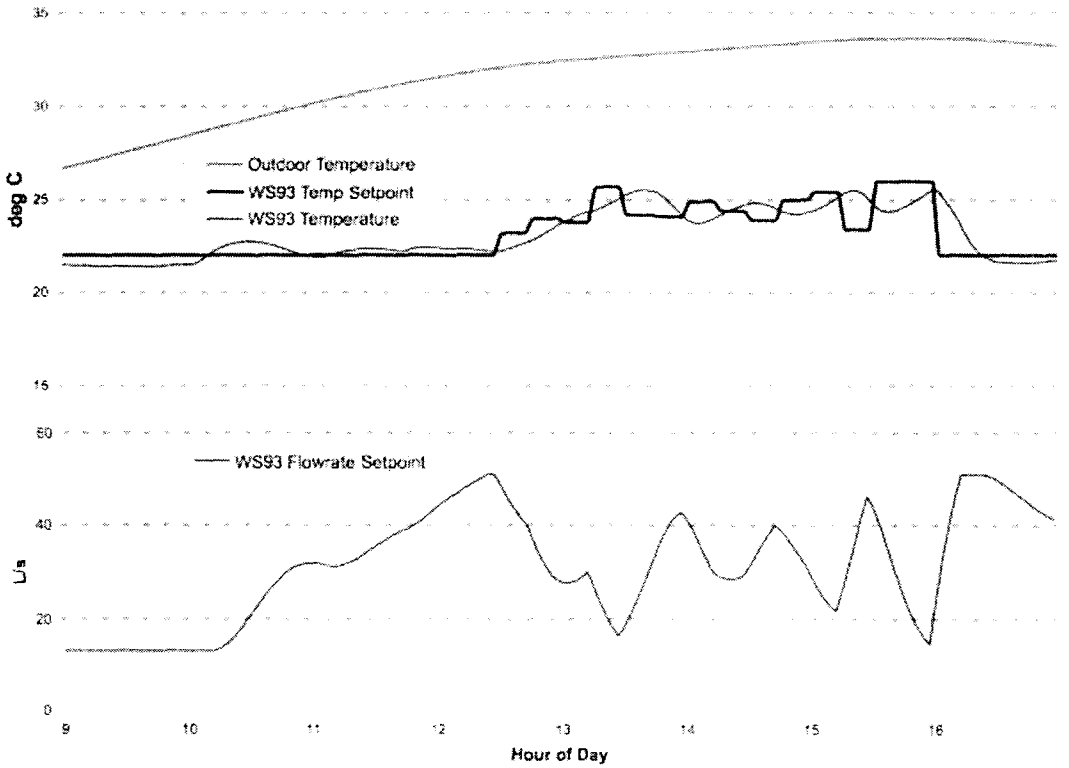
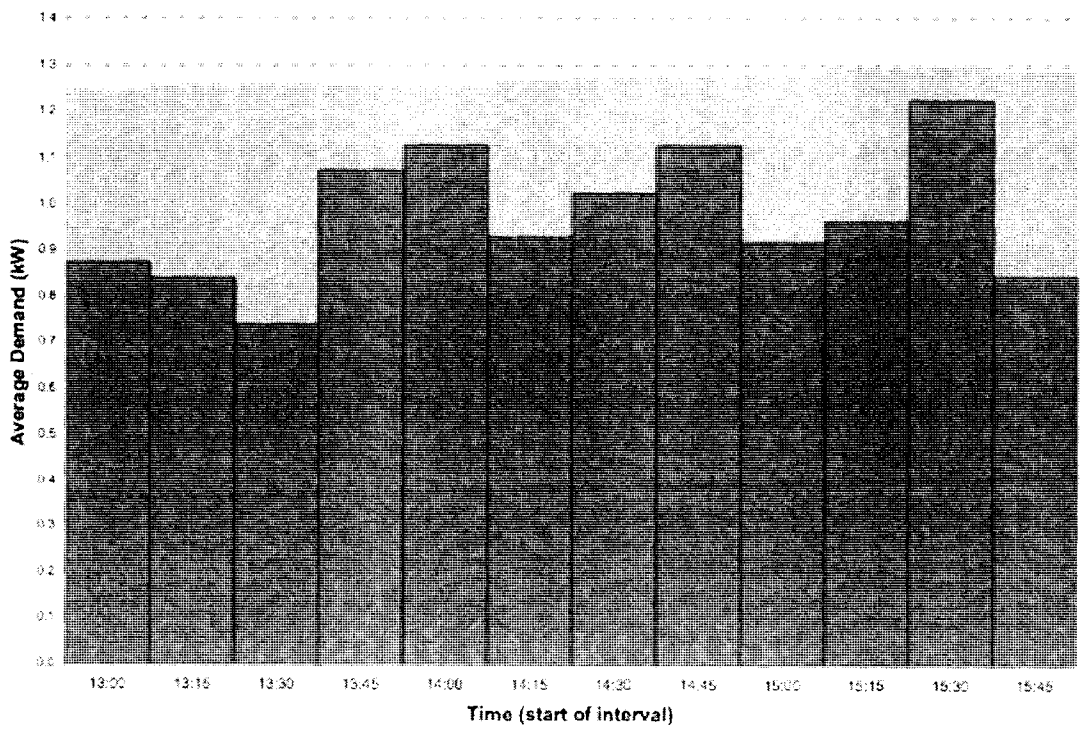


Figure 6.13: Ideal, time constraints, no rules, learning or constraints – demand



6.4 Imperfect-Model Case

This section and the next begin to explore the impacts of imperfections. In such cases, it is possible for the simulation-based approach to actually do worse than the suggested starting points that it uses, since it can under-evaluate some points or over-evaluate others because of its inaccuracies.

The software framework allows for a more complete study of the impacts than are discussed here. In the case of imperfect model calibration, the 'simulated building' can be given different parameter values than the 'actual building' (in the two-model test scheme), so a wide variety of mismatches can be modeled and analyzed. A large sampling would provide information about the effects of different levels of deviation, and would be able to determine the relative impacts of each variable, which would help in determining what variables might require closer consideration and/or better data.

Just one case is considered here, to demonstrate how it is done and to rough out the impact level that could be expected for a poorly calibrated model. Perfect initialization and prediction are assumed, with the calibration parameter values as shown in Table 6.2.

Table 6.2: Calibration parameters used to simulate a poorly calibrated model

Parameters	*'Actual Bulding'*	*'Sim. Building'*
Induction supply	0.11 m^3/s	0.10 m^3/s
Background lighting	64.4 W	79.4 W
Occupant lighting	10.3 W/occ	10.6 W/occ
Computer use	9.7 W/occ	10.9 W/occ
Infiltration coeff. K1	0.106	0.111
Room capacitance	400 kJ/C	450 kJ/C
Solar transmittance	0.64	0.60

The application of SimCon (with the same GA parameters as above, and with logarithmic starting-points and previous-time-step learning) in this case resulted in a 27.13% trim, which is slightly worse than the rule-based logarithmic trim. The set-point control is shown in Figure 6.14 (only small deviations were made from the logarithmic suggested starting points), and the demand map is shown in Figure 6.15.

Figure 6.14: Imperfect calibration – set-point control

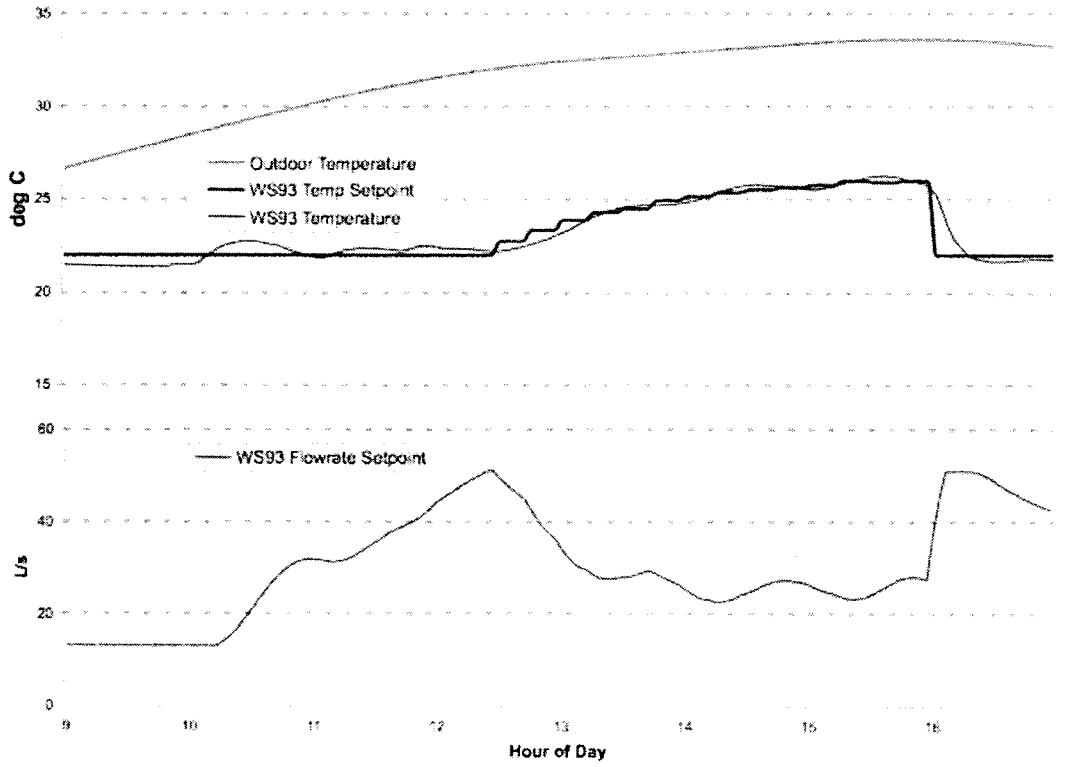
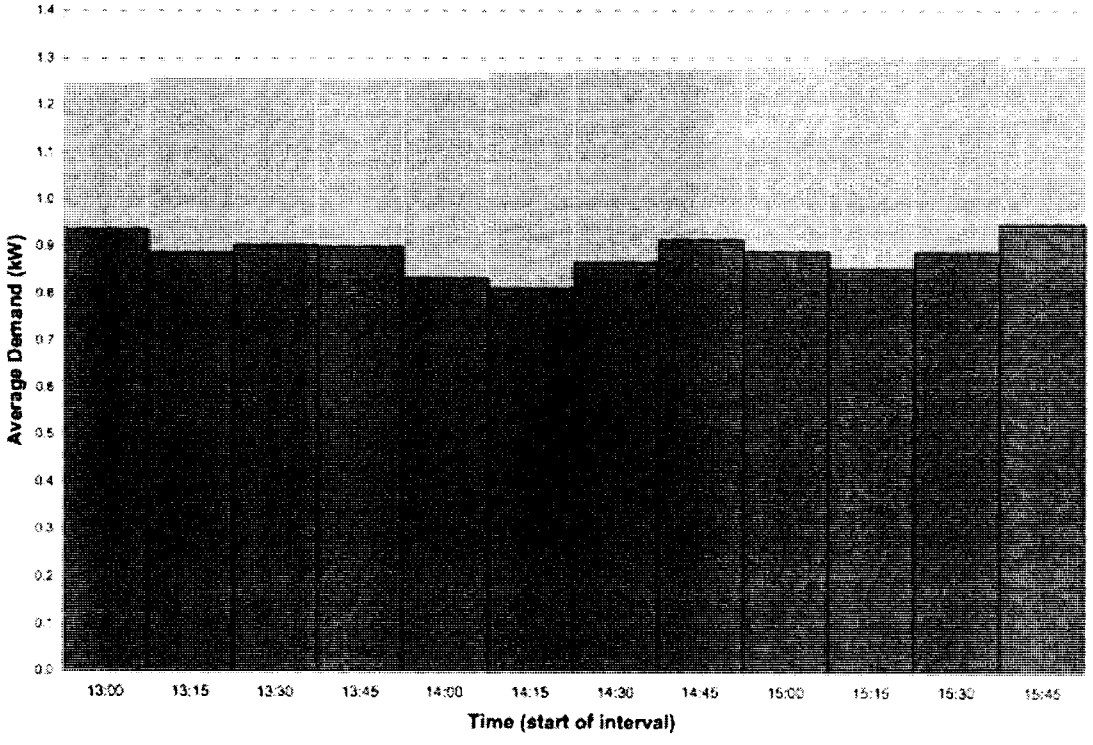


Figure 6.15: Imperfect calibration – demand map



6.5 Imperfect-Initialization-and-Prediction Case

The performance of the simulation-based supervisory controller was found to be remarkably sensitive to inaccuracies in the 'conditions' information sent to it, both for predicted future conditions and for current and historical conditions (for the model's warm-up period). Even small deviations from the 'actual' conditions were found to produce inconsistent and generally poor performance of SimCon. Figure 6.16 shows slight deviations for T_{amb} caused by stepped inputs for the historical conditions (to the left of the dotted line) and the predicted future conditions (to the right of the dotted line). Similar deviations were used in the 'conditions' inputs for the direct and indirect solar gains. Figure 6.17 shows the resulting load with a logarithmic trim, for the 'simulated building' model, and compared with the load in the 'actual building' model.

Figure 6.16: Slight imperfections in initial conditions and predictions

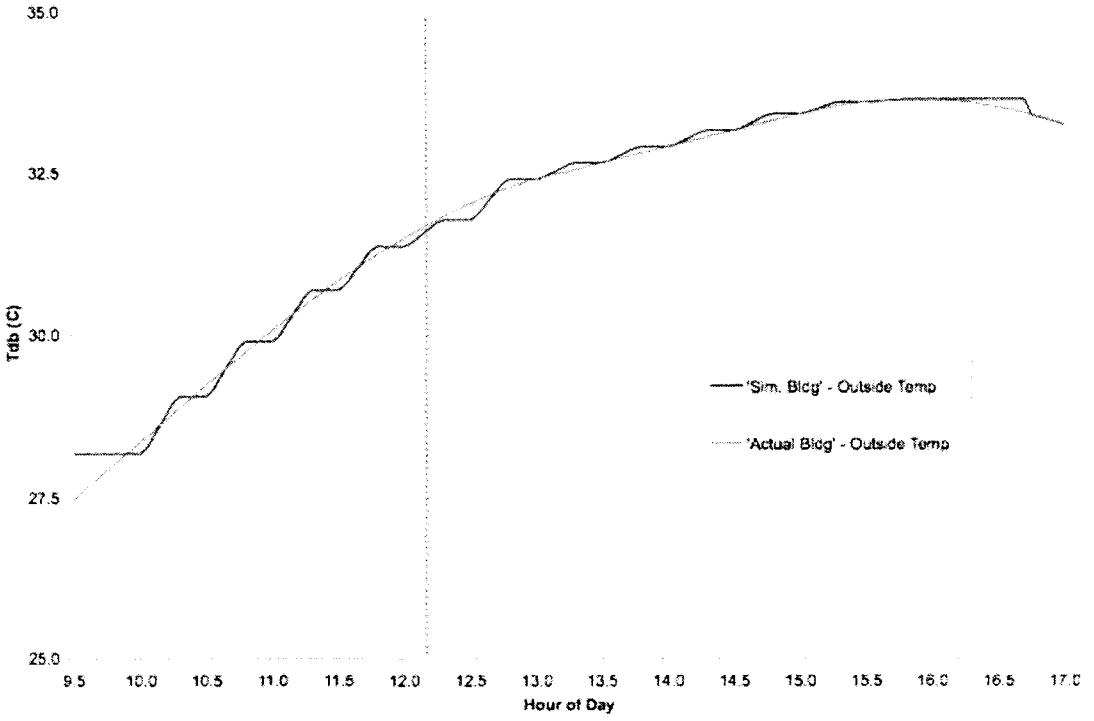
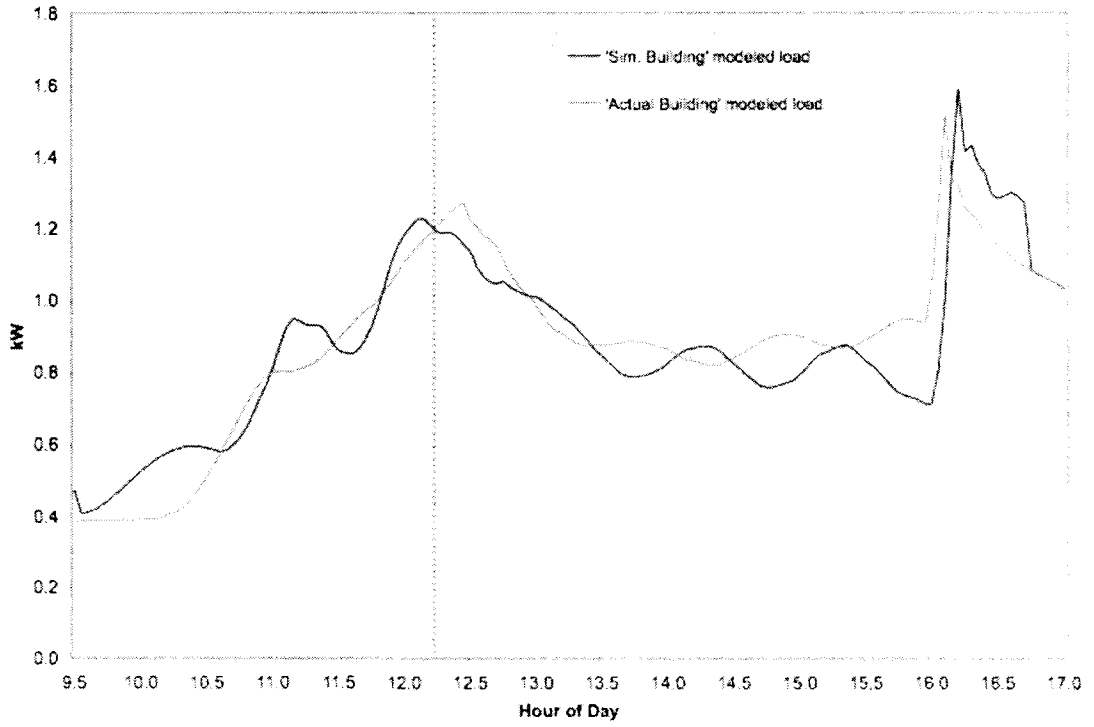


Figure 6.17: Resulting load shape with logarithmic set-point control



This level of error at each time-step quickly adds up in two-model tests over the full trim period. Figures 6.18 and 6.19 show the results of a two-model run using the stepped conditions shown above. The decisions made by SimCon are rational responses to the 'conditions' it is given, but they produce poor responses for the 'actual' building. This run produced just a 10.13% trim.

Figure 6.18: Minor inaccuracies in initial conditions and predictions – set-points

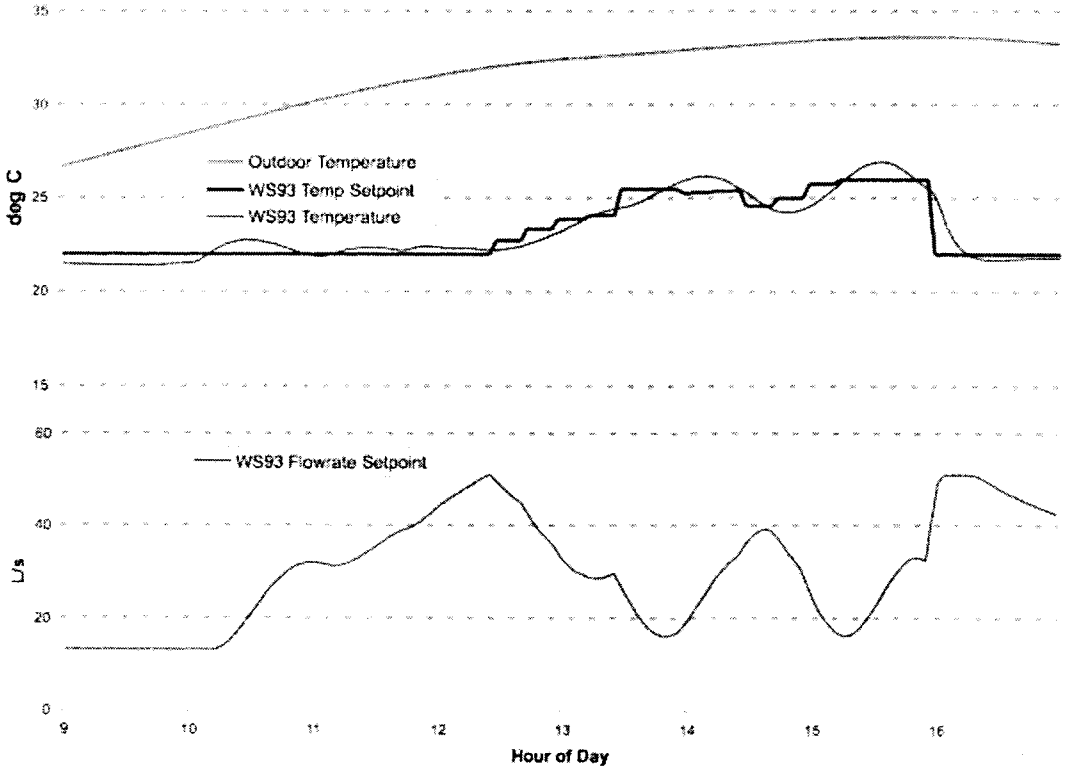
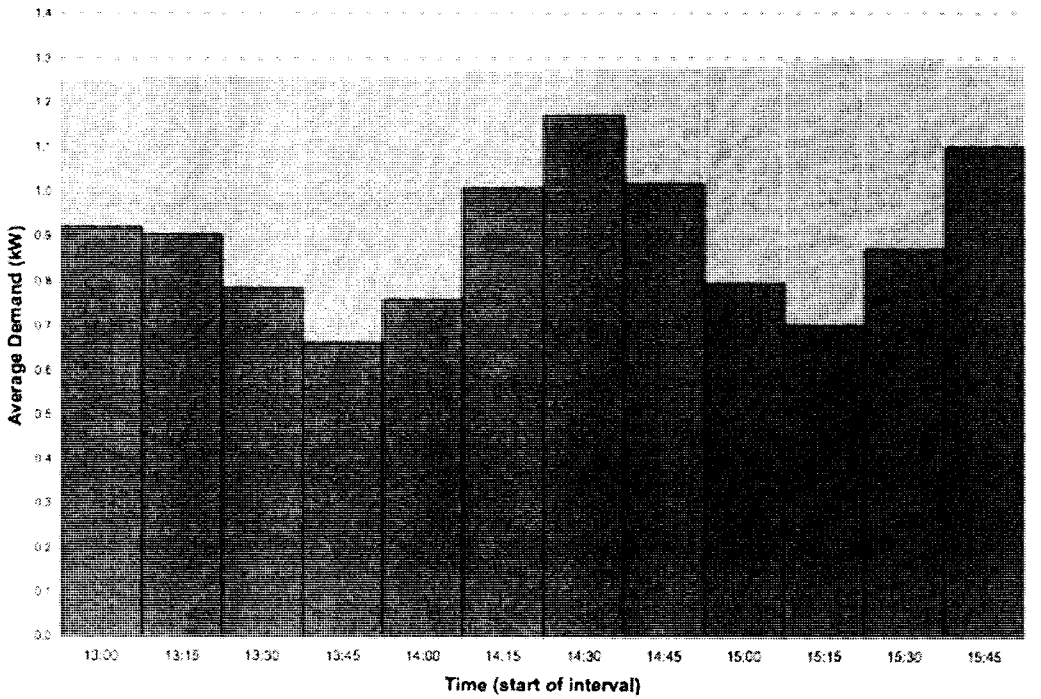


Figure 6.19: Minor inaccuracies in initial conditions and predictions – demand



Other systematic inaccuracies can be modeled and tested. Figures 6.20 and 6.21 show the results of a run with the same stepped conditions as above but also with a 5°C underestimation of the ambient temperature for the entire lead-up period. In this case, SimCon actually performs better than the case just presented, producing a 20.32% trim (this variation is perhaps just because of the stochastic nature of the algorithms, or perhaps in this case the two sources of error somehow evened each other out). Figures 6.22 and 6.23 show the results of a run with the same stepped conditions but with a 5°C underprediction of the ambient temperature for the entire future horizon – this produced an 8.09% trim. In general, it seems that after a certain level of inaccuracy, the results become quite sporadic. And these initial tests suggest that the level of inaccuracy at which that happens is quite low. The further characterization of impacts of inaccuracy will be an important part of future research in this area.

Figure 6.20: 5°C under-estimation of T_{amb} during model warm-up – set-points

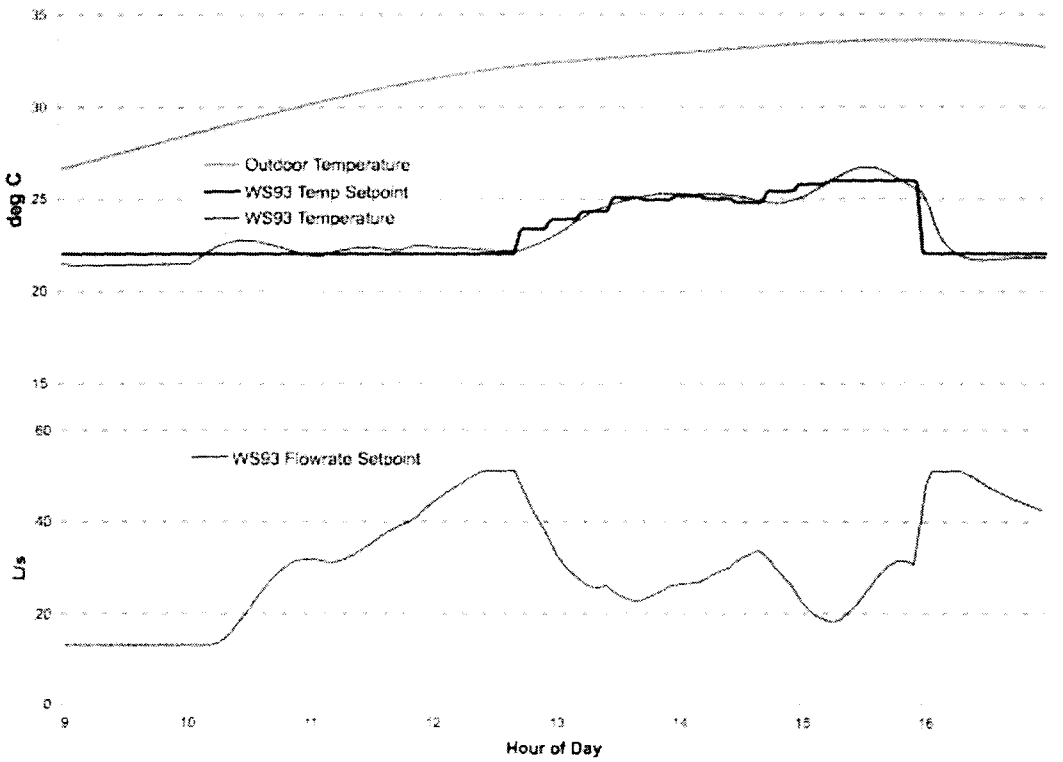


Figure 6.21: 5°C under-estimation of T_{amb} during model warm-up – demand map

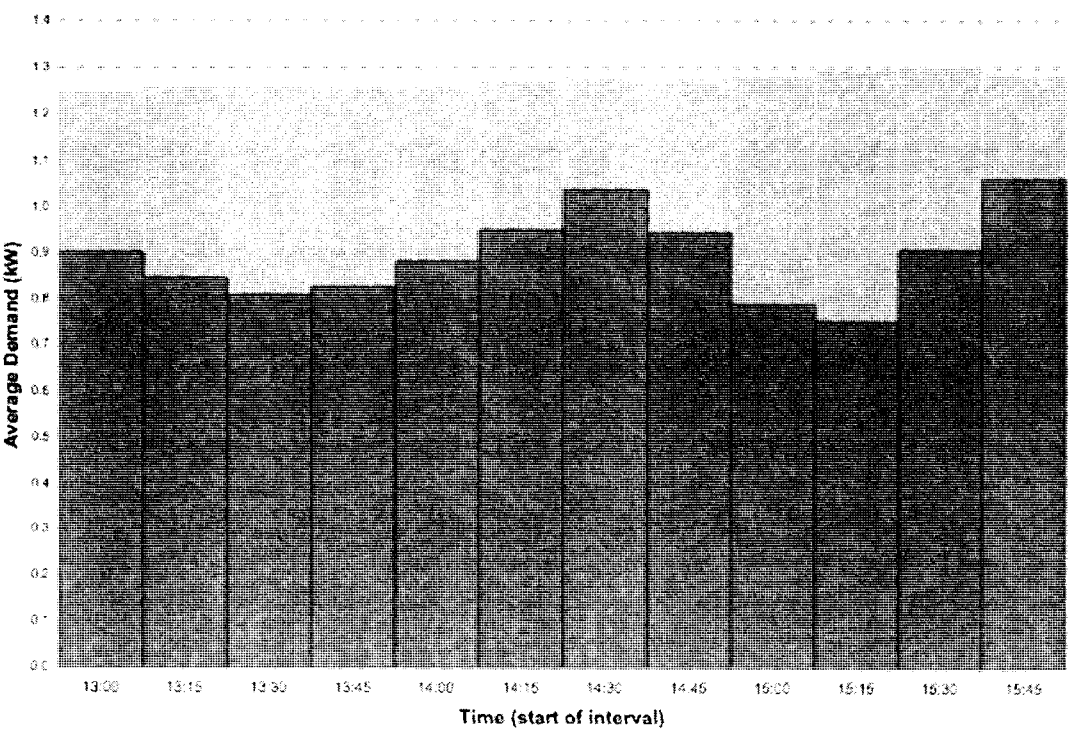


Figure 6.22: 5°C under-prediction of T_{amb} over future horizon – set-point control

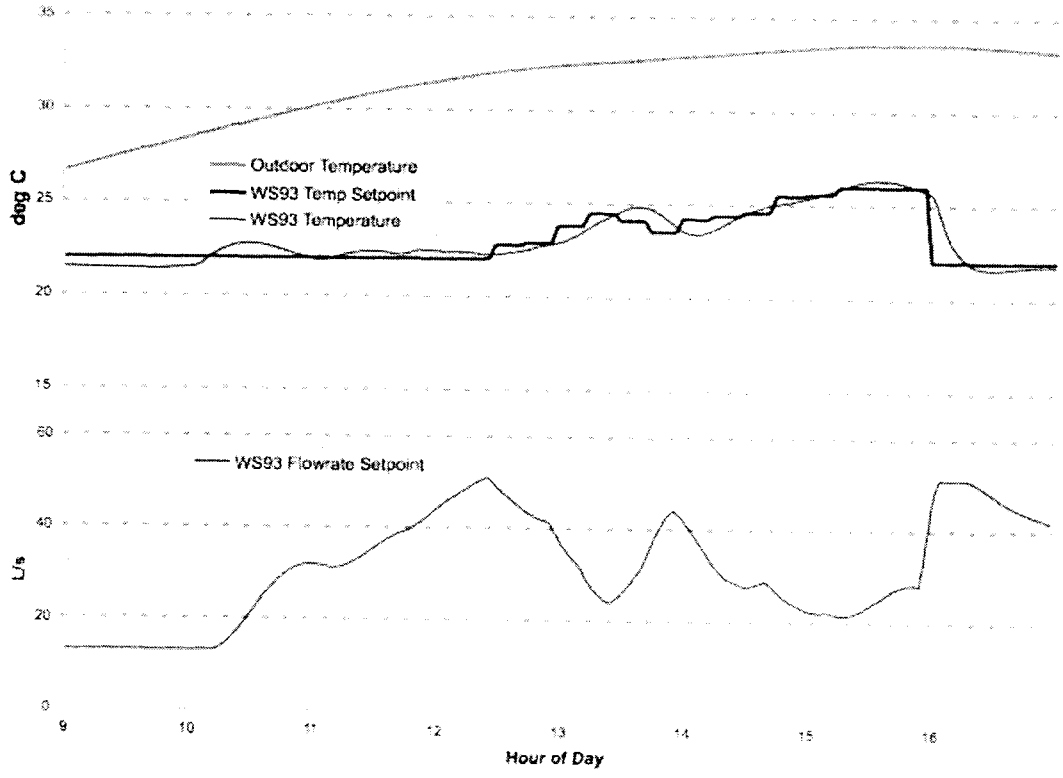
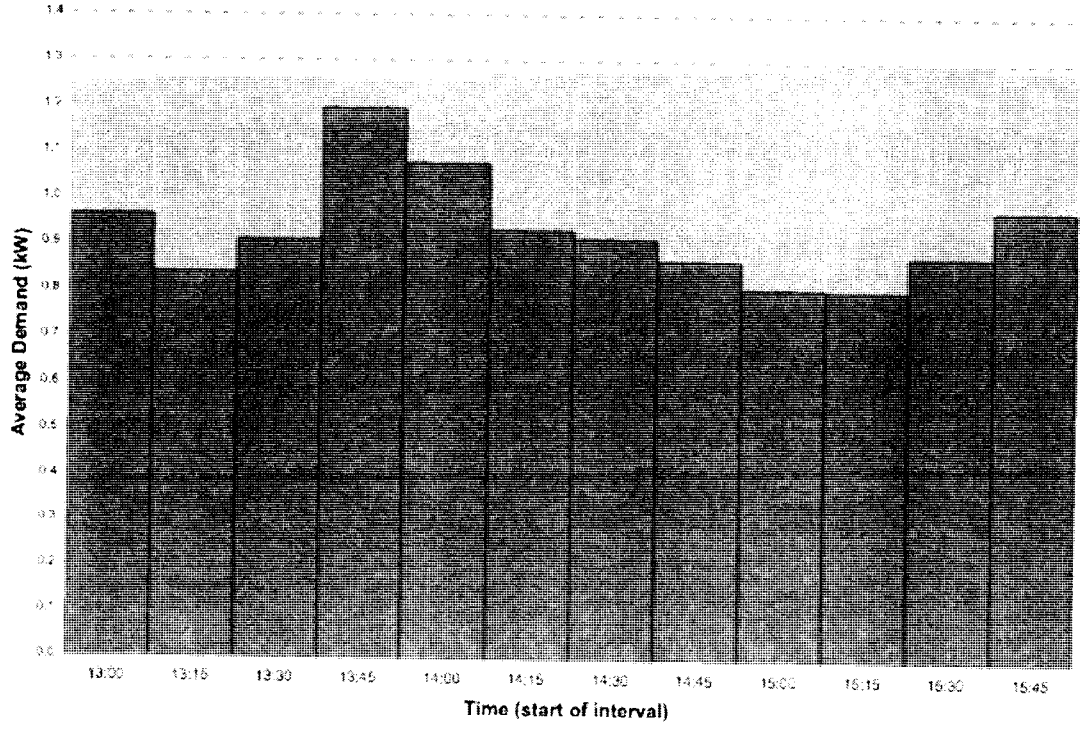


Figure 6.23: 5°C under-prediction of T_{amb} over future horizon – demand map



6.6 Summary

The performance of the simulation-based supervisory control framework with the modified genetic algorithm has been evaluated for some ideal and non-ideal circumstances. By first relaxing the computational time constraints, SimCon was used to approximate optimal control, which was found would produce a trim of 30.12%, compared to the 27.97% trim produced by the logarithmic rule-based approach. Under the ideal circumstances of perfect model calibration and perfect initialization and prediction, but with realistic time-constraints, SimCon with a modified genetic algorithm with rule-based initial-point suggestions and previous-time-step learning produced at 29.46% trim. Its performance was not nearly as good without logarithmic starting-point suggestions or previous-time-step learning. Similarly, its performance decreased with imperfect model calibration and imperfect initialization and prediction.

Table 6.3: Summary of results

			Max Demand (W)	Trim Percentage
		Base Case	1301	-
Rule-Based		Jump Trim	1001	23.03%
		Linear Trim	1123	13.69%
		Log Trim	937	27.97%
Simulation-Based	Ideal	No Time Constraints	909	30.12%
		With Time Constraints	918	29.46%
		No Learning	936	28.04%
		No Rule Suggestions	1225	5.91%
	Non-Ideal	Imperfect Calibration	948	27.13%
		Imperfect Initialization	1169	10.13%

Chapter 7

Discussion

We begin with a brief discussion of what the results suggest about rule-based and simulation-based approaches to zone temperature ramping control, and then turn our attention to what the results and case study process can tell us about simulation-based supervisory control in general. This leads into a discussion of possible improvements and extensions to the simulation-based control framework and algorithms, and outlines some possibilities for future research.

7.1 Logarithmic trim for temperature ramping

Although it may have been easily predicted at the outset, the results show quite conclusively that simulation-based supervisory control can perform little better than a well-parameterized logarithmic trim in the zone temperature ramp demand response control application. And in addition to the vastly more complex set-up process associated with simulation-based control, the results have also shown that it can easily perform worse than rule-based logarithmic control for this case. So for practical application, a simple logarithmic trim seems the appropriate choice.

And it should be noted that the rule-based approach has generally outperformed the simulation-based approach in this case study, even with the rule-based approach adhering to the same discrete time-step constraints that had to be used by the simulation-based approach. In practice, the rule-based approach could be used to reset set-point

values much more often than once every 15 minutes, and as the set-point changes get closer to being continuous, the rule-based approach would likely perform a bit better.

7.2 Complexity level appropriate for simulation-based control applications

The case study does not consider, however, possible ways that the scenario could be more complex, for example if light dimming is being applied alongside the zone temperature ramping, or if more detailed consideration is to be given of occupant comfort under these transient thermal conditions (Newsham et al, 2006). For such situations, simulation-based control may be worth considering, or perhaps the use of control rules based on simulation-based control studies. This remains an open question for future research.

In general, the relative performance of rule-based and simulation-based control in this case study suggests that there might be a certain level of problem complexity below which simulation-based control is simply not appropriate. It is difficult to assess what this level might be, but future research should be considering more complex problems (with more control variables and/or more complex control objectives), if we are to test practical applications of simulation-based supervisory control.

But using this simple problem, with its known and relatively simple but high-performing rule-based strategy and intuitive optimality check (knowing that the optimal control sequence would produce a flat demand line), has provided a good test of the software framework and algorithms, and has provided some fodder for further analysis, discussion and development, particularly concerning the results with model and prediction inaccuracies.

7.3 Imperfect models, initializations and predictions

At the outset of this research, the challenge of optimization within tight time-constraints was considered to be of primary concern. It remains important, but the case study results have highlighted the need for more consideration of model accuracy, initial conditions variables and predictions. If not dealt with effectively, these aspects could undermine the performance of any practical or theoretical application of simulation-based control. More research is required in these areas, but there are some good starting points to go on.

7.3.1 Continuous model calibration

As discussed in Appendix 3, the automated error-minimization approach to model calibration allows for the model to be continuously tuned to become more accurate over time. And since the tracking of actual conditions and simulation outputs can be used both for automated calibration and for good organization-layer inputs to the optimization layer, setting up a good data tracking system can do double duty in a simulation-based control framework.

7.3.2 Initialization

There are many tradeoffs that must be thought through here. Longer warm-up times for the simulations can allow greater accuracy and/or require fewer sensors (although more data points, since their history must be kept), but this also means longer simulation times and thus fewer objective-function calls within each time-step.

Possible improvements in this area are somewhat more dependent on changes in simulation tools themselves than on any changes that can be made in the optimization or organization layers. A key improvement would be the ability of simulation programs to warm up just once for a group of parametric or optimization simulations, rather than warming up each time. This could amount to just saving a system snapshot at a user-specified time (in TRNSYS, writing the S-array to memory may suffice), and then starting subsequent simulations from that same point.

7.4 Reducing computation time: possible algorithm improvements

Although the impacts of inaccuracies have shown themselves to be very important, the challenge of time-constrained simulation-based optimization remains an interesting and important area of study. This dissertation has presented a software framework for the development and evaluation of simulation-based control optimization algorithms, but has only begun to scratch the surface of algorithm development and testing. The modified genetic algorithm contains the main elements of organization-layer to simulation-layer communication for which this framework was envisioned – namely learning from previous time-steps, rule-based starting-point suggestions and dynamically modifying constraints – but there is much more that can be done, both in the basic optimization algorithm and in the use of information from the organization layer.

7.4.1 Optimization algorithms, independent of organizational layer

Genetic algorithms do have the benefits of allowing multiple starting-point suggestions alongside other random starting points, but they are perhaps not ideal for this

control-optimization application, since they do not converge to solutions very quickly. Other optimization algorithms might be more appropriate for simulation-based control optimization, and could be developed for use within this supervisory control framework. (For example, the generalized pattern search algorithms available in the standard GenOpt library should be explored for use in this context.)

Optimization processes are often considered as having two stages – the first stage scans the entire domain in search of a promising area or areas, and then the second stage focuses in to find the best point in this area. This division into global and local searches is not always applicable (e.g. some hill-climbers skip the global search, relying on the initial point(s) they are given; and some broader-searching algorithms, such as GAs with high mutation rates, never really settle into a local search), but it does provide a nice way of viewing the process, and underlines part of the challenge here. Hybrid algorithms are likely worth developing for this application, with different approaches used for the two different stages, and the optimization algorithm could be made to switch from one to the other after a certain number of simulations. Even if good starting points can be provided (as discussed in the next section), it is still worth including a broad search of the domain in case there are unexpected configurations with good potential for energy savings – for this, a rough grid or random points spread across the domain would be best. For the second stage, another smaller grid could be used, followed by a gradient-based approach (with the gradient estimated by using the calculated points).

Another possible approach is similar to a technique that is currently becoming more common for optimization with CFD or with very complex simulations. In this technique, Latin hypercube sampling is used to choose points in the search space, and

then these points are simulated and the outputs are used to train an ANN. The ANN is then used with an optimization algorithm, running much faster than the simulation software or CFD. And depending on its transfer functions, the ANN might also be optimized analytically. Latin hypercube sampling is not appropriate if the variables are relatively co-dependent, so it must be used with care, but in cases where the number of simulations are limited (as in the present case) it is certainly worth considering, especially if the ANN optimal value is only used as a suggestion and incorporated into a GA which tests it with a simulation. In fact, the whole process could be intertwined with a GA: hypercube sampling to set most of the initial population; the results of the simulations for the first generation are used to train the ANN; the ANN-optimal points incorporated as individuals every generation thereafter; and the points simulated by the GA continue to be used to further train the ANN, making it more accurate over the course of the optimization process. (A similar ANN-entwined GA was developed and used in the calibration process discussed in Appendix 3, with some success, but it needs to be further tested and incorporated into the simulation-based control framework.)

7.4.2 Better use of inputs from the organizational layer

There are more interesting ways of using the organizational layer to inform the optimization algorithm. The organizational layer could search for more complex patterns in the previous results and pass these on to the optimization algorithms: it could find correlations strengths between variable values in optimization outputs, noting conditions where variables should take on similar values (e.g. if in the best values found by the various algorithms thus far under the given conditions for the illustrative example, the

variable values tend to change very little over the time horizon – knowing this could help the optimization algorithm to limit its search); and the organizational layer could perform sensitivity analyses on the previous results to determine which variables have the greatest effect on the objective function, and the optimization algorithm could focus on these variables first before tweaking the other variable values. SimCon has been programmed such that the optimization instruction algorithms are in a separate, extendable library, so that further research and development can be done in this area.

It is worth noting that with an increased use of learning techniques, and tests over longer times than our short one-day case study, SimCon algorithms could begin to take on a much different nature – they could begin to take on some of the benefits of an ANN approach. And aside from the possible ways that entire SimCon configurations can emulate some of the ANN's learning attributes, there are numerous possibilities for the incorporation of standard ANNs at various levels within the SimCon structure: ones created during the optimization process (e.g. in the GA-ANN-intertwined process discussed above); one created by the organizational layer from previous simulation results to mimic the building model (the 'resultsTracking' database currently only records optimization outcomes, but it could be made to record all of the simulation inputs and outputs), which can be used to run quick optimization routines within the organizational layer to provide suggested initial points for the optimization algorithm within GenOpt; and another created by the organizational layer to map conditions directly to decisions using previous results, again to provide suggested initial points to the optimization algorithm within GenOpt.

Other possible improvements to the use of organization-layer information include: finding ways of treating future time-steps differently the further into the future they are (e.g. to decrease their relative precision or constrain their search ranges as a way of placing more focus on the more-immediately upcoming time-steps); and refining old or developing new rules based on results from past optimization (possibly through the use of decision trees and Genetic Programming).

7.4.3 Moving away from the black box

All of these things could significantly improve the efficacy of simulation-based design and/or control optimization. However, they all still walk around the simulation program, treating it essentially as a black box, generating and testing possible points. Such an approach is inherently limited – it precludes the use of any analytical approaches that could vastly improve the efficacy of optimization algorithms. Some planned upcoming work is to look at possible ways of revealing more of the simulation's underlying structure to optimization algorithms, so that they can use this to parse the problem into sub-regions or use analytically-derived gradients, both of which could dramatically improve computational efficiency. Such a development could also allow for the backing-out of set-point values given a desired objective, as was used in the inverse-model studies of Lee and Braun, that were noted in the literature review.

7.5 Links to other operations-phase uses of building simulation

One of the most alluring aspects of simulation-based control is that joins with a number of other possible applications of simulation in building operations. Model-based

building system diagnostics has been an active field of research for a number of years, and is showing some promise as a technology that may begin to see widespread use in the near future . The interaction between the idea of a model-based controller with automated calibration and a model-based diagnostics tool is interesting – in the first case you want an accurate model of how the building is actually performing, but in the second case you want an accurate model of how the building *should* be performing. But the models must share the same structure, and could be treated as two sides of the same model within an integrated controls-and-diagnostics system, differing only in terms of some parameter values. An interesting future research project would be to set up such a system, considering the calibration techniques suggested in Appendix 3.

Another use of simulation in building operations is for retrofit analysis. Having a well-tuned physics-based model of the building could be of great use in evaluating possible retrofit strategies. This is one of the advantages of simulation-based control over a strictly ANN-based control.

7.6 Using SimCon for the development and evaluation of standard control sequences

Although this dissertation has focused on the study of real-time simulation-based control, it should be acknowledged that another possible use of SimCon (and perhaps a much more practical use in the short and mid term, and possibly even in the long term) is for the development and evaluation of standard rule-based control sequences for complex systems. By relaxing the computational time constraints and providing ideal circumstances (perfect model calibration, initialization and prediction), SimCon could be used to approximate optimal control for any given (simulated) set of conditions. Iterating

over various sets of conditions and analyzing the SimCon outputs could allow researchers to devise rules for near-optimal control, and to analyze the trade-offs between rule complexity and performance.

Chapter 8. Conclusions and Recommendations

The outcomes of this research can be divided in two parts: those pertaining to the particular case study at hand; and those pertaining to the software framework and simulation-based control in general. We deal first with the particulars, then with some general thoughts and recommendations, and end with some ideas for future research.

8.1 Conclusions and recommendations pertaining to the case study

It was found that the logarithmic trim was very nearly ideal in this case, and should be highly recommended for use in other similar cases (although further research is required to study the case involving light ramping or other considerations alongside the temperature set-point changes). The optimal set-point configuration, however, is actually quite complex and likely highly case-dependent, and it would be difficult to imagine a rule-based approach that would capture it. The logarithmic approach produced a 27.97% trim, while the optimal was estimated to be slightly greater than 30.12%. The application of simulation-based supervisory control with a modified genetic algorithm with rule-based initial-point suggestions and previous-time-step learning, under the inherent time-constraints of the problem but with the otherwise ideal conditions of perfect model calibration, initialization and prediction, produced a 29.46% trim. Without rule-based initial suggestions or previous-time-step learning, simulation-based control with a genetic algorithm does not perform nearly as well. And the performance of the simulation-based control configuration drops off as circumstances move away from the ideal: with imperfect model calibration but perfect initialization and prediction, SimCon performed

slightly worse than the logarithmic approach, producing a 27.13% trim; and with perfect model calibration but imperfect initialization and prediction, the results from SimCon became more sporadic and much further from the optimal. It should be noted that since the genetic algorithm is stochastic, the results can vary from test to test. But in the ideal case, the trim is always between the logarithmic trim and the true optimum (because of the combination of logarithmic starting-point suggestions and previous-time-step learning). In the non-ideal cases, these results can vary much more, depending on what imperfections are included in the test.

8.2 Recommendations pertaining to simulation-based supervisory control

The primary purpose of the case study was to provide fodder for the testing of software and algorithms, in the hopes of learning more about the challenges and potential benefits of simulation-based supervisory control, and to further the development of a software framework for its study and application. In addition to highlighting the significant logistic challenges associated with keeping track of the complex array of variables, text files and algorithms (which are only partially mediated by the rudimentary interface created for SimCon), this case study has underlined the importance of model initialization. Time-constrained optimization remains a significant intellectual and practical challenge, as does model calibration, but without proper model initialization, wildly inaccurate conclusions can be reached by the simulation-based controller. Further research is required in this area. And, as noted in the literature review, future research into simulation-based control for buildings should more closely consider what might be learned from model-predictive control research in other industries.

In general, this research has done more to open up possibilities for future research than it has answered any particular questions. The software framework described and applied in this dissertation allows for the investigation of simulation-based supervisory control under ideal conditions or under conditions of model inaccuracy or imperfect initialization or prediction. It also provides a framework for the development and testing of algorithms for simulation-based building control optimization. And it opens up possibilities for studies of the technical energy-savings potential of dynamic building systems, for the development and/or evaluation of standard rule-based control sequences for complex systems, and for the consideration of optimal control in the building design phase. Simulation-based optimizing supervisory control remains an interesting and important field of research, and it is hoped that this software framework and case study might help in furthering its development.

8.3 Further research possibilities

8.3.1 Variable controller time-steps and continuous control possibilities

One notable aspect of simulation-based control that has not yet been considered in detail is the nature of the controller time-steps. Because of the computational time requirement for optimization, simulation-based supervisory control must use discrete time-steps, and a trade-off must be made allowing more time for the optimization to converge and getting results quickly enough to produce reasonably smooth control. Further research is needed to better understand the nature of this trade-off, and to determine appropriate time-step lengths for applications. And currently within the

framework, all control decisions are being considered at each time-step, but in more complex cases with more control variables, it may be worth considering different time-steps for different variables. And the research in this dissertation has conceived of the control decisions as being used for a constant set-point over each time-step, but it may be more appropriate in some cases (including in the case study considered here) for the control decisions to be considered as modifying the gradient of a continuously-changing set-point value. The current version of the software framework does not allow for anything other than a single discrete control time-step, so research into variable time-steps and continuous control possibilities would likely require some modifications to the framework.

8.3.2 Using SimCon in situ for a real-time application

Testing this approach with the actual PEC system considered in this case study is technically quite feasible, because the control system is already connected to a computer system through WebPEC. Similar arrangements could be configured for many other case study applications. The supervisory control program written within Excel for this case study could be used as an interface between the control system and SimCon, just as it is in the two-model testing arrangement – it would just be a matter of writing a software patch that would send the sensor data from control system to the Excel spreadsheet and then send the set-point data from the Excel spreadsheet to control system. The computer used for this testing must have the proper software installed (the simulation program, Java, GenOpt and SimCon), and should be expected to be used for nothing other than running the test.

Long-term testing with the actual system would provide a good way of exploring some of the potential for system-learning available with the framework. One could test to see how much the supervisory system improved over time by incorporating optimization suggestions drawn from the results tracking database. The model could be periodically updated through automated calibration using data available in the results tracking database. And ways of using the model for system diagnostics could also be explored.

8.3.3 Technical potential of highly-interactive dynamic building systems

Previous research on various forms of dynamic solar shading (see, for example, Lee et at, 2004) has shown that they can produce very significant energy savings. And it is becoming a commonly-held belief among researchers that great efficiency gains can be gathered through better integration and control of facades, lighting and HVAC. SimCon could be used to provide an approximation of optimal control for technical-potential studies in this area.

8.3.4 Interactions between control optimization and occupant control

Personal control tends to lead to better occupant satisfaction and can also lead to decreased energy use in some cases (see Newsham et al 2004, and their subsequent studies). But one may be able to find ways of decreasing energy consumption further without decreasing occupant satisfaction very much. Or one may be able to identify set-point configurations that could result in an environment that the occupant would find more comfortable, and these configurations could be suggested to the occupant. This type of give and take between the central system and the occupant is far more complicated

than the simple over-ride that might be called for in the demand-response case (as considered in this case study), but it is also more interesting.

9. References

Beausoleil-Morrison I. 2000. The Adaptive Coupling of Heat and Air Flow Modeling within Dynamic Whole-Building Simulation. PhD thesis, University of Strathclyde, Glasgow.

Borrelli F., Bemporad A., Fodor M., Hrovat D. 2006. An MPC/hybrid system approach to traction control. IEEE Trans. Control Systems Technology. Vol 14, no. 3. pp. 541-552.

Charron R., Athienitis A. 2006. The Use of Genetic Algorithms for a Net-Zero Energy Solar Home Design Optimisation Tool. Proceedings of PLEA2006 (Conference on Passive and Low Energy Architecture), Geneva, Switzerland.

Clarke J.A., Dempster W.M., Negrao C. 1995. The Implementation of a Computational Fluid Dynamics Algorithm within the ESP-r. System Proceedings of Building Simulation '95, Madison, pp. 166-75.

Clarke J.A., Crockroft J., Conner S., Hand J.W., Kelly N.J., Moore R., O'Brien T., Stachan P. 2002. Simulation-assisted control in building energy management systems. Energy and Buildings 34 pp. 933-940.

CBI: Commercial Buildings Initiative. 2008. www.zeroenergycbi.org

DABO: Diagnostic Agent for Building Operators. 2008. CANMET Energy Technology Center - Varennes. Energy http://cetc-varennes.nrcan.gc.ca/en/b_b/bi_ib.html

Flake B.A. 1998. Parameter Estimation and Optimal Supervisory Control of Chilled Water Plants. PhD thesis, University of Wisconsin – Madison.

Henze G.P., Kalz D.E., Felsmann C., Knabe G. 2004. Impact of Forecasting Accuracy on Predictive Optimal Control of Active and Passive Building Thermal Storage Inventory. HVAC&R Research Vol 10, No 2 pp. 153-177.

Henze G.P., Kalz D.E., Liu S., Felsmann C. 2005. Experimental Analysis of Model-Based Predictive Optimal Control for Active and Passive Thermal Storage Inventory. HVAC&R Research Vol 11, No 2 pp. 189-213.

Henze G.P., Krarti M. 2005. Predictive Optimal Control if Active and Passive Building Thermal Storage Inventory: Final Report. DOE Award Number: DE-FC-26-01NT41255

Kelly G.E. 1988. Control System Simulation in North America. Energy and Buildings 10, pp. 193-202.

Kummert M., André P. 2005a. Simulation of a Model-Based Optimal Controller for Heating Systems under Realistic Hypotheses. In Proceedings of the 9th International

Building Performance Simulation Association (IBPSA) Conference 2005, Montreal, Canada.

Kummert M., André P., Argiriou A. 2005b. Performance Comparison of Heating Control Strategies Combining Simulation and Experimental Results. In Proceedings of the 9th International Building Performance Simulation Association (IBPSA) Conference 2005, Montreal, Canada.

Lee E.S., Yazdanian M., Selkowitz S. 2004. The Energy-Savings Potential of Electrochromic Windows in the US Commercial Buildings Sector. LBNL-54966, http://btech.lbl.gov/pubs

Lee K., Braun J. 2004. Development and Application of an Inverse Building Model for Demand Response in Small Commercial Buildings. Proceedings of SimBuild 2004.

Lee K., Braun J. 2006a. Development of Methods for Determining Demand-Limiting Setpoint Trajectories in Commercial Buildings using Short-Term Data Analysis. Proceedings of SimBuild 2006, Boston.

Lee K., Braun J. 2006b. Evaluation of Methods for Determining Demand-Limiting Setpoint Trajectories in Commercial Buildings using Short-Term Data Analysis. Proceedings of SimBuild 2006, Boston.

Lee K., Braun J. 2007. Reducing Peak Cooling Loads through Model-Based Control of Zone Temperature Setpoints. Proceedings of the 2007 American Control Conference.

Liu S., Henze G. 2005. Calibration of Building Models for Supervisory Control of Commercial Buildings.In Proceedings of the 9th International Building Performance Simulation Association (IBPSA) Conference 2005, Montreal, Canada.

Liu S., Henze G.P. 2006. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory Part 1. Theoretical foundation. Energy and Buildings 38. pp. 142-147

Mahdavi A. 2001. Simulation-based control of building systems operation. Building and Environment 36. pp. 789-796.

Mahdavi A., Pröglhöf C. 2005. A Model-Based Method for the Integration of Natural Ventilation in Indoor Climate Systems Operation. In Proceedings of the 9th International Building Performance Simulation Association (IBPSA) Conference 2005, Montreal, Canada.

Mahdavi A., Spasojević B., Brunner K.A. 2005. Elements of a Simulation-Assisted Daylight-Responsive Illumination Systems Control in Buildings. In Proceedings of the 9th International Building Performance Simulation Association (IBPSA) Conference 2005, Montreal, Canada.

Mara T.A., Garde F., Boyer H., Mamode M. 2001. Empirical validation of the thermal model of a passive solar cell test. Energy and Buildings, 33, pp. 589-599

Mayne D., Rawlings J., Rao C., Scokaert P. 2000. Constrained model predictive control: Stability and optimality. Automatica, vol 36(6), pp. 789-814.

Morari M., Lee J.H. 1999. Model predictive control: past, present and future. Computers and Chemical Engineering, 23, pp. 667-682.

Nassif N., Stainslaw K., Sabourin R. 2005. Optimization of HVAC Control System Strategy Using Two-Objective Genetic Control Algorithm. HVAC&R Research, Vol 11 No 3 pp. 459-486.

Negrao C. 1995. Conflation of Computational Fluid Dynamics and Building Thermal Simulation. PhD thesis, University of Strathclyde, Glasgow.

Newsham G.R., Veitch J.A., Arsenault C.D., Duval C.L. 2004. Lighting for VDT Workstations 1: Effect of Control on Energy Consumption and Occupant Mood, Satisfaction and Discomfort. Research Report, Institute for Research in Construction, National Research Council Canada, B3208.3 RR-165

Newsham G., Donnelly C., Mancini S., Marchand R., Lei W., Charles K., Veitch J. 2006. The Effect of Ramps in Temperature and Electric Light Levels on Office Occupants: A Literature Review and a Laboratory Experiment. 2006 ACEEE Summer Study on Energy Efficiency in Buildings.

Priolo C., Sciuto S., Sperduto F. 2001. Efficient Design Incorporating Fundamentals Improvements for Control and Integrated Optimisation: Final Report. http://lesowww.epfl.ch/anglais/techint/Edificio.pdf

Reddy T.A. 2006. Literature Review on Calibration of Building Energy Simulation Programs: Uses, Problems, Procedures, Uncertainty, and Tools. ASHRAE Transactions, Vol. 112 (2), pp. 226-240.

Sun J., Reddy T.A. 2006. Calibration of Building Energy System Simulation Programs using the Analytic Optimization Approach (RP-1051) International Journal of HVAC&R Research, 12(1), pp. 177-196

Wang S., Jin X. 2000. Model-based optimal control of VAV air-conditioning system using genetic algorithm. Building and Environment 35. pp. 471-487.

Wang W., Zmeureanu R., Rivard H. 2005. Two-Phase Application of Multi-Objective Genetic Algorithms in Green Building Design. Proceedings of IBPSA-2005, Montreal.

Wetter M., Wright J. 2003. Comparison of a Generalized Pattern Search and a Genetic Algorithm Optimization Method. Proceedings of IBPSA-2003, Eindhoven, Netherlands.

Wetter M. 2004a. GenOpt 2.0: Generic Optimization Program. http://gundog.lbl.gov/GO/

Wetter M. 2004b. GenOpt User Manual, Version 2.0.0. Technical Report LBNL-54199. http://gundog.lbl.gov/GO/

Wetter M., Wright J. 2004. A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. Building and Environment, Vol. 39, pp. 989-999.

Wetter M., Haves P. 2008. Building Control Virtual Test Bed. https://gaia.lbl.gov/bcvtb

Wright J., Alajmi A. 2005. The Robustness of Genetic Algorithms in Solving Unconstrained Building Optimization Problems. Proceedings of IBPSA-2005, Montreal.

Xu P., Haves P., Braun J.E., and Hope L.T. 2004. Peak Demand Reduction from Pre-Cooling with Zone Temperature Reset in an Office Building. Proceedings of the ACEEE 2004 Summer Study on Energy Efficiency in Buildings.

Xu P., Haves P. Kim M. 2005. Model-Based Automated Functional Testing – Methodology and Application to Air-Handling Units. ASHRAE Transactions 2005, OR-05-13-4.

Xu P. 2006. Evaluation of Demand Shifting Strategies with Thermal Mass in Two Large Commercial Buildings. Proceedings of SimBuild 2006, Boston

Zhai Z., Chen Q., Haves P., Klems J. 2002. On approaches to couple energy simulation and computational fluid dynamics programs. Building and Environment, Vol 27, pp. 857-864.

Zhou L., Huang H., Shakeri A., Rastan S., Stach B., Pero K., Morofsky E., Haghighat F. 2005. Indoor Environment in an Office Floor with Nozzle Diffusers: a CFD Simulation. Proceedings of Building Simulation Conference 2005, Montreal.

Zhou L., Shakeri A., Rastan S., Pero K., Morofsky E., Huang H., Kutrowski E., Haghighat F. 2006 A Comparative study of Underfloor Air Distribution System and ceiling System: CFD Simulations. Proceedings of Building Simulation Conference (IBPSA Canada) 2006, Toronto.

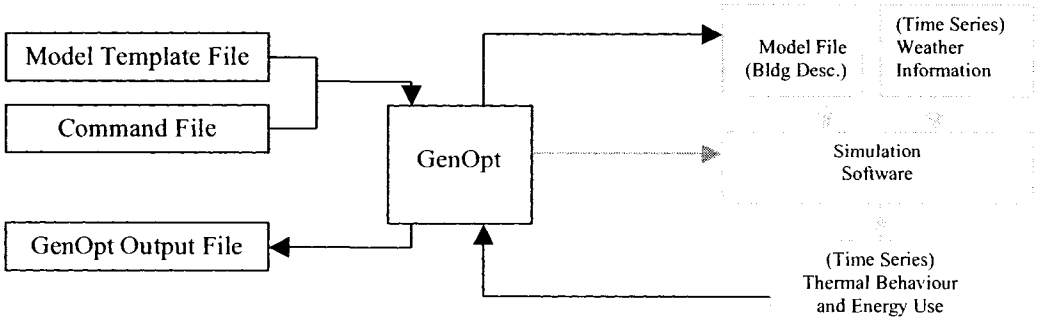
A1. Details of Software Framework

The software framework was developed based on the conceptual outline provided in Chapter 3. This appendix covers some of the details of how the software works, and is intended for the reader interested in better understanding the underlying logistics of the case studies presented in this dissertation, or those who may be interested in using or extending the software framework in further research.

A1.1 Review of GenOpt basics

Since GenOpt plays a central role in the software framework, it is worth first noting the basic structure of this existing software, since it strongly influences the structure of the rest of the framework. Figure A1.1 shows the basic operation of GenOpt. The command file instructs GenOpt on what optimization algorithm to use, and provides information about variable constraints and other information required by the algorithm, such as the desired precision of the output or the maximum number of iterations.

Figure A1.1: GenOpt basics

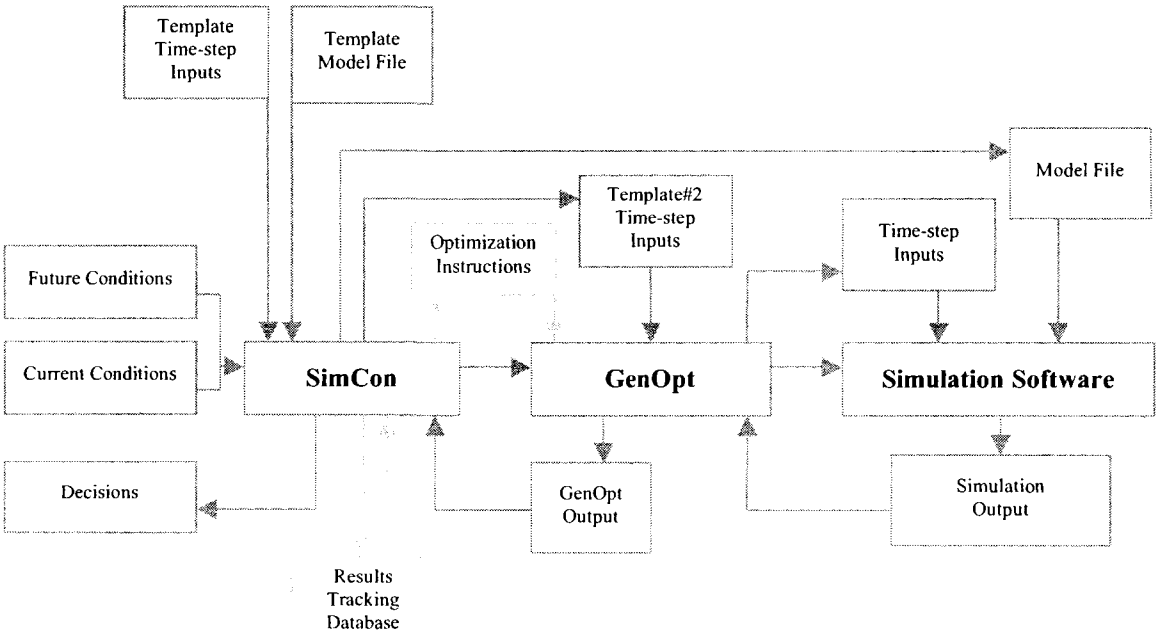


To evaluate the objective function at a particular point, GenOpt generates a new building model, calls the simulation program and reads the value of the objective function from the simulation output file. It creates the new building model by using the template file supplied by the user – this file is identical to what would be read by the simulation program, except that the optimization variables are demarcated by '%' on either side of a variable name. GenOpt recognizes the %x% in the template file and replaces it with the value of x for the point under consideration. (See Wetter, 2004b for GenOpt details.)

A1.2 Overall structure and model template files

A number of different text files are used to pass information between the programs used in the three different layers. A schematic is shown in Figure A1.2.

Figure A1.2: Overall flow of text files and instructions in the software framework



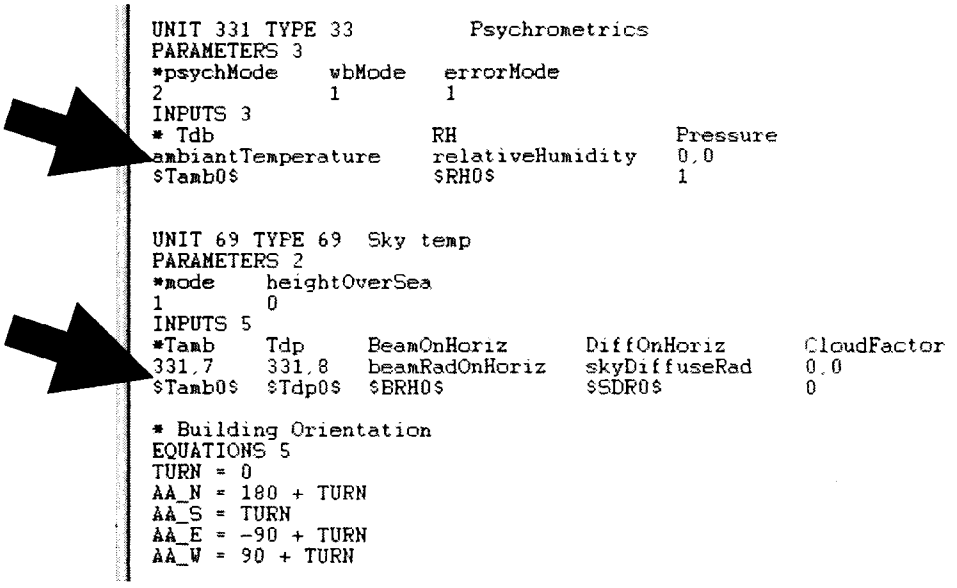
The flow of information is as follows:

- SimCon receives files containing information about current and predicted conditions, and writes this information into the building model that it sends to the simulation layer and to an input file template that it sends to GenOpt. It also writes a file called 'optimization instructions' that is sent to the GenOpt layer. (This 'optimization instructions' file performs a similar function to the 'command' file that GenOpt usually uses, but it was created as a separate file for this framework to allow greater flexibility.) It later reads the GenOpt output file and writes the control decisions file that is sent back to the building's central control system. It also writes conditions and results to the results tracking database, which it can use later to find promising points to write into the 'optimization instructions' file.
- GenOpt reads the time-step input template and optimization instructions, uses these to write input files that are read by the simulation layer, calls the simulation program and then reads the simulation outputs. GenOpt continues to write new input files, representing different control configurations, until it either finds an optimal configuration or runs out of time, and then writes an output file that gets read by the SimCon layer.

The template files referred to in this schematic ("Model With Initial Conditions, Template" and "Time-step Inputs, Template") are essential to its operation, so it is worth considering how these are configured. The particulars of the model and input file templates are dependant on the case under consideration, but as shown in Figures A1.3

and A1.4, there are parts of any model file template and input file template that contain variable names between '$' or '%' markers.

Figure A1.3: Partial screenshot of "Model With Initial Conditions, Template" text file



```
UNIT 331 TYPE 33          Psychrometrics
PARAMETERS 3
*psychMode     wbMode    errorMode
2              1         1
INPUTS 3
* Tdb                    RH                    Pressure
ambiantTemperature      relativeHumidity      0,0
$Tamb0$                 $RH0$                 1


UNIT 69 TYPE 69  Sky temp
PARAMETERS 2
*mode     heightOverSea
1         0
INPUTS 5
*Tamb     Tdp      BeamOnHoriz       DiffOnHoriz       CloudFactor
331,7     331,8    beamRadOnHoriz    skyDiffuseRad     0,0
$Tamb0$   $Tdp0$   $BRH0$            $SDR0$            0

* Building Orientation
EQUATIONS 5
TURN = 0
AA_N = 180 + TURN
AA_S = TURN
AA_E = -90 + TURN
AA_W = 90 + TURN
```

Figure A1.4: Partial screenshot of "Time-step Inputs, Template" text file

```
$Tamb1$, $RH1$, $RS1$, $RN1$, $RE1$, $RW1$, %b1%    , %w1%    , $WS1$
$Tamb2$, $RH2$, $RS2$, $RN2$, $RE2$, $RW2$, %b2%    , %w2%    , $WS2$
$Tamb3$, $RH3$, $RS3$, $RN3$, $RE3$, $RW3$, %b3%    , %w3%    , $WS3$
$Tamb4$, $RH4$, $RS4$, $RN4$, $RE4$, $RW4$, %b4%    , %w4%    , $WS4$
$Tamb4$, $RH4$, $RS4$, $RN4$, $RE4$, $RW4$, %b4%    , %w4%    , $WS4$
```

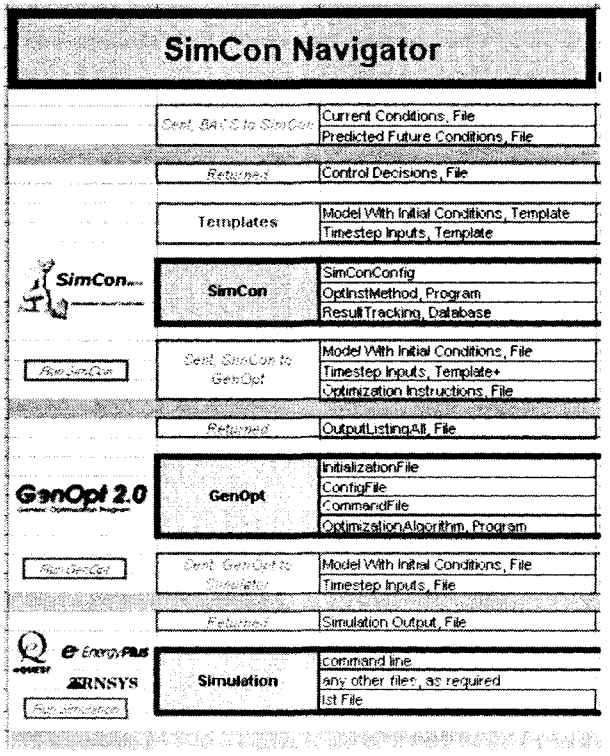
The variables demarcated by '%' are the control set-points, and their numerical values are entered into the text files by GenOpt each time a new configuration is to be tested with the simulation software. The variables demarcated by '$' are the initial and predicted future conditions, and their numerical values are entered into the text files by SimCon at the beginning of each time-step. (It should be noted that the consideration of initial conditions is very important in this scheme, since the simulation length for the

models under consideration is very short. The model template and SimCon layer must be properly configured to ensure that these are dealt with appropriately.)

A1.3 Rudimentary interface

In order to help the user keep track of the various text files when a case is being configured, a rudimentary user interface has been developed in Excel. The three basic layers are shown, along with the various text files that are used in the process. When the user clicks on a file name, it opens up in a text editor. The Excel interface plays no functional role in the program – it just assists in file organization. SimCon and GenOpt are both java programs that use text files for inputs and outputs, and they run independently of this interface.

Figure A1.5: Screenshot of a section of the rudimentary interface



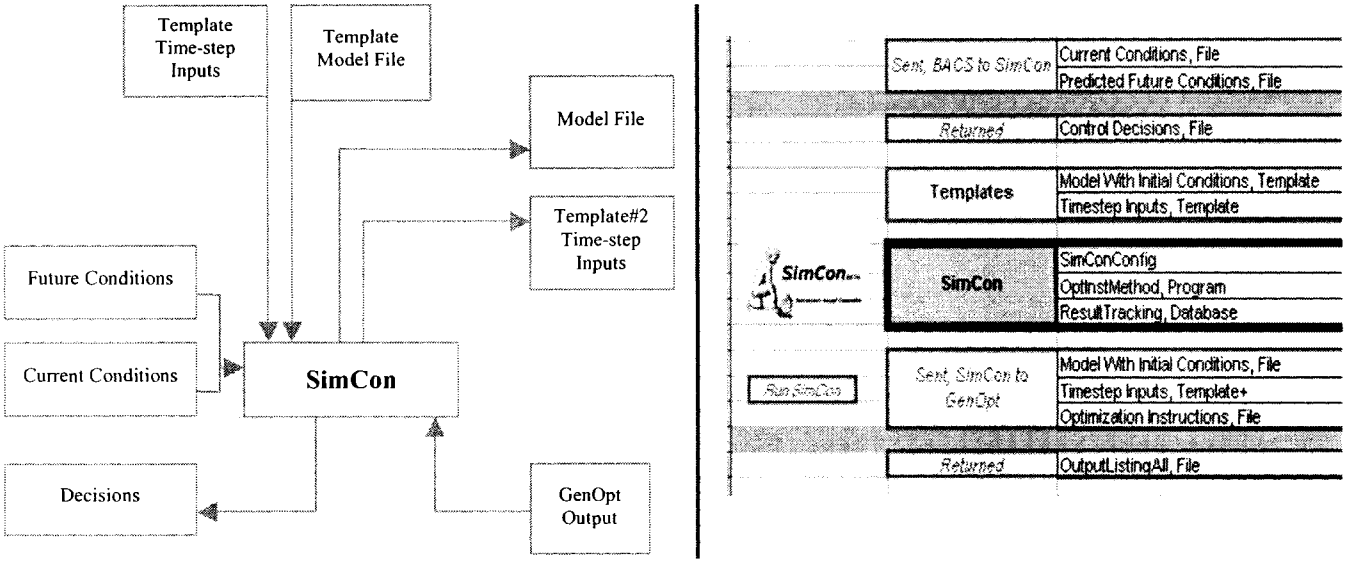
A1.4 Organizational Layer: SimCon

SimCon is a java program. Most of its code is dedicated to reading and writing text files. It first reads the SimCon configuration file, which tells it where to find the necessary input and output files. It then uses the model and input templates and the condition files to create the model and input files. It calls GenOpt and then reads its output file when it is finished, and writes the decision file. And in between these essential routines, SimCon also writes the conditions and optimization results to a database for results tracking (in the prototype, the 'database' is just a very large text file).

Before SimCon calls GenOpt, it calls another java program within the simcon package that is used to write the optimization instructions file. Because the user may wish to choose between one of many different ways of creating this file, or may wish to write their own algorithm for this aspect, it has been placed aside as a library, similar to how GenOpt deals with optimization algorithms. The library of java routines that can write optimization instructions files are contained in the java package {simcon.optinst}.

So the rudimentary interface has three files associated with the internal workings of the SimCon layer, as shown in Figure A1.6: a SimCon configuration file, a results tracking database, and a java file that produces the optimization instructions file.

Figure A1.6: SimCon configuration



The main purpose of the optimization instructions file is to pass information from the organizational layer to GenOpt's optimization algorithms (which can be user-created, as they are in this case). This file can be used to pass information about what a rule-based strategy might do under the current conditions, and the optimization algorithm can use this as a good starting point for its optimization search. It can also be used to provide knowledge-based suggestions on what parts of the search space are very unlikely to produce useful results. And by using the result-tracking database, this file can also be used to send information about previous optimization searches found under similar conditions, again providing the optimization algorithm with good starting points. Other uses of the instructions file and the result-tracking database are possible, as discussed in Chapter 7, and should be explored in future research.

A1.5 Optimization Algorithms within GenOpt

The structure of the optimization and simulation layers is shown in Figure A1.7 As noted earlier, GenOpt is structured such that the user may choose from a library of optimization algorithms, and may create their own to add to this library. There are a number of algorithms within the standard GenOpt library which may be of interest in simulation-based control, and they can be modified for use within SimCon (with or without the use of the optimization instructions file). For the current research study, a custom algorithm was developed, as discussed in Chapter 3, based on a genetic algorithm. The algorithm being used in a given case needs to be referenced and configured by the 'command file' that is used by GenOpt. An example 'command' file is shown in Figure A1.8 – this particular command file is used in the illustrative example discussed near the beginning of Chapter 6.

.

Figure A1.7: GenOpt and TRNSYS configuration

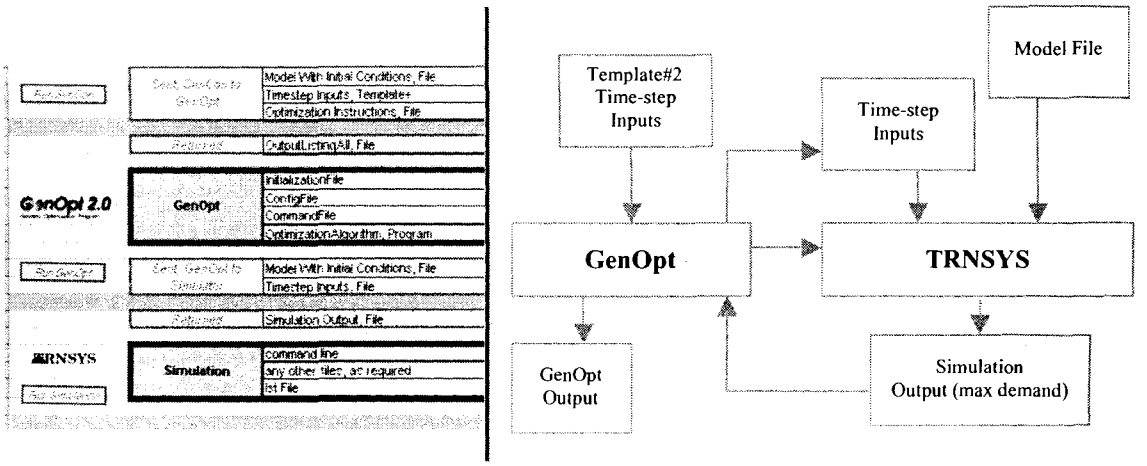


Figure A1.8: GenOpt command file for PEC model study, simple parametric algorithm

```
// GenOpt command file for PEC study, parametric algorithm

Vary{
 Parameter{   // temperature set-point for future time-step 03
    Name   = set1;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 04
    Name   = set2;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 11
    Name   = set3;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 12
    Name   = set4;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 13
    Name   = set5;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 14
    Name   = set6;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 21
    Name   = set7;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 22
    Name   = set8;
    Min    = 23;
    Ini    = 0;
    Max    = 25;
    Step   = 1;
  }
 Parameter{   // temperature set-point for future time-step 23
    Name   = set9;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 24
    Name   = set10;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 31
    Name   = set11;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 32
    Name   = set12;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 33
    Name   = set13;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
 Parameter{   // temperature set-point for future time-step 34
    Name   = set14;
    Min    = 25;
    Ini    = 0;
    Max    = 25.1;
    Step   = 0;
  }
}

OptimizationSettings{
 MaxIte = 2000;
 MaxEqualResults = 100;
 WriteStepNumber = false;
}

Algorithm{
 Main = EquMeshForSimCon2;
 StopAtError = true;
}
```

A1.6 Virtual Testing Environment

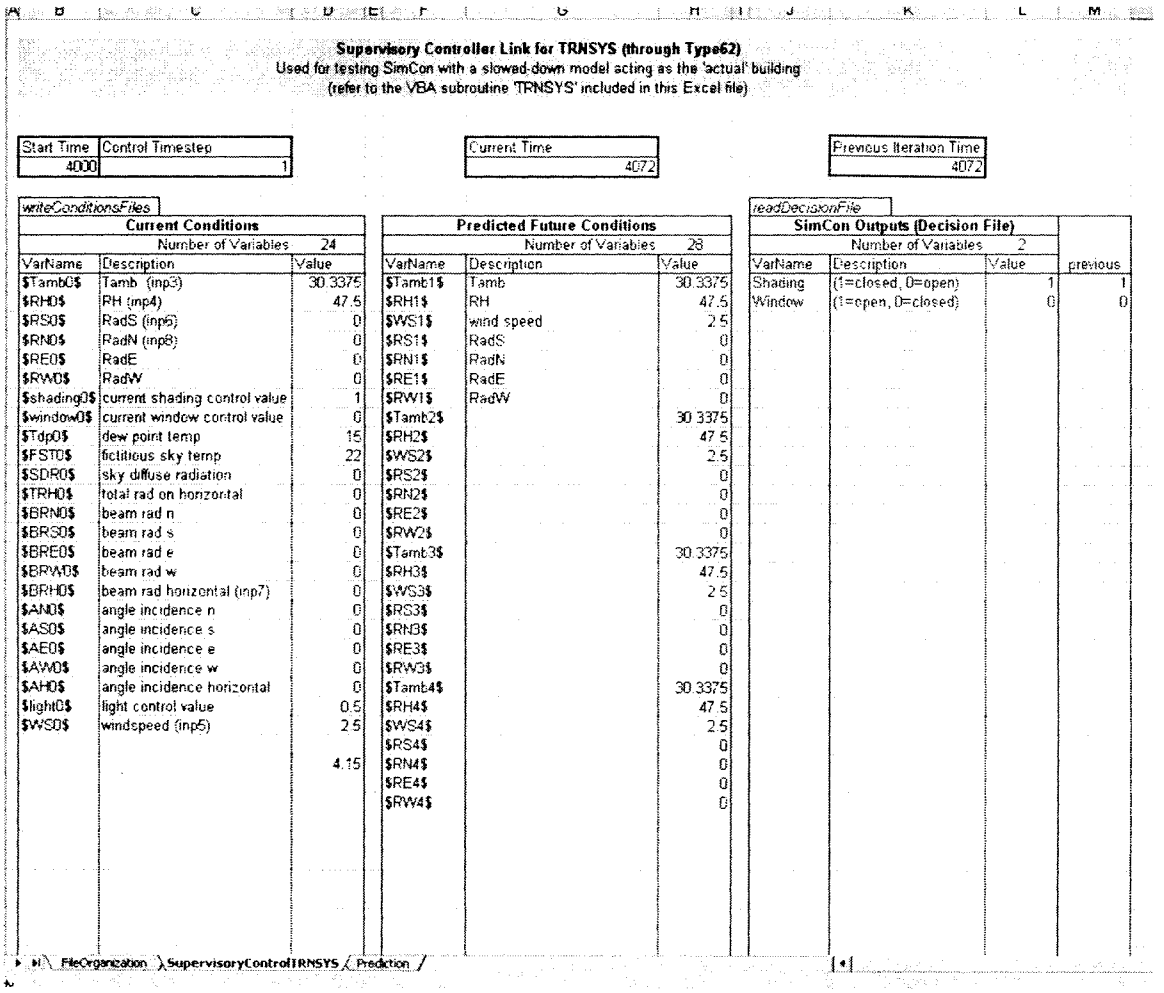
Note that a single call to SimCon only calculates control decisions for the one controller time-step for which that call was made. In order to test its application over time, it must be used in conjunction with either an actual building control system that calls SimCon every controller time-step, or a virtual building that mimics this situation. This virtual building can be a simulation model (generally a copy of the model used within the simulation layer of the framework) must have two special aspects: it must be slowed down to mimic the operation of the building at speeds that are near to real-time; and it must have a supervisory control module that writes the conditions text files, calls SimCon, and reads and acts upon the decisions text file.

Thus two different building models (both representing the same building) are being used simultaneously – the 'actual' building model that is taking the place of a physical building and control system, and the building model that is being repeatedly called by SimCon through GenOpt. Luckily, with TRNSYS at least, this does not pose any computational problems, as long as two separate installations of the software are present in separate locations on the computer, and the two sets of text files for the two models are kept separate. This same idea could also be used with two separate computers over a local network.

To slow the building model down, a new Fortran module was created for TRNSYS which simply tells it to sleep for a pre-defined period of time, doing so once every time-step within the TRNSYS simulation. And the supervisory control module was created by using the Type62 module in the standard TRNSYS library, which calls an Excel file – an appropriate Excel file was created to write the conditions text files given the information

from the 'actual' building simulation, run SimCon, and read the decision text file before sending this information back to the 'actual' building simulation. (This Excel file was merged with the SimCon Navigator interface, with the two on separate worksheets, to aid in organization.) A screenshot of part of the supervisory control module set up in Excel is shown in Figure A1.9.

Figure A1.9: Supervisory control module in Excel



Supervisory Controller Link for TRNSYS (through Type62)
Used for testing SimCon with a slowed-down model acting as the 'actual' building
(refer to the VBA subroutine 'TRNSYS' included in this Excel file)

Start Time	Control Timestep
4000	1

Current Time
4072

Previous Iteration Time
4072

writeConditionsFiles

Current Conditions — Number of Variables: 24

VarName	Description	Value
$TambO$	Tamb (inp3)	30.3375
RHO	RH (inp4)	47.5
RSO	RadS (inp6)	0
RNO	RadN (inp8)	0
REO	RadE	0
RWO	RadW	0
$shadingO$	current shading control value	1
$windowO$	current window control value	0
$TdpO$	dew point temp	15
$FSTO$	fictitious sky temp	22
$SDRO$	sky diffuse radiation	0
$TRHO$	total rad on horizontal	0
$BRNO$	beam rad n	0
$BRSO$	beam rad s	0
$BREO$	beam rad e	0
$BRWO$	beam rad w	0
$BRHO$	beam rad horizontal (inp7)	0
ANO	angle incidence n	0
ASO	angle incidence s	0
AEO	angle incidence e	0
AWO	angle incidence w	0
AHO	angle incidence horizontal	0
$lightO$	light control value	0.5
WSO	windspeed (inp5)	2.5
		4.15

Predicted Future Conditions — Number of Variables: 28

VarName	Description	Value
$Tamb1$	Tamb	30.3375
$RH1$	RH	47.5
$WS1$	wind speed	2.5
$RS1$	RadS	0
$RN1$	RadN	0
$RE1$	RadE	0
$RW1$	RadW	0
$Tamb2$		30.3375
$RH2$		47.5
$WS2$		2.5
$RS2$		0
$RN2$		0
$RE2$		0
$RW2$		0
$Tamb3$		30.3375
$RH3$		47.5
$WS3$		2.5
$RS3$		0
$RN3$		0
$RE3$		0
$RW3$		0
$Tamb4$		30.3375
$RH4$		47.5
$WS4$		2.5
$RS4$		0
$RN4$		0
$RE4$		0
$RW4$		0

readDecisionFile

SimCon Outputs (Decision File) — Number of Variables: 2

VarName	Description	Value	previous
Shading	(1=closed, 0=open)	1	1
Window	(1=open, 0=closed)	0	0

Appendix 2: Coupling Energy Simulation with CFD Using a Neural Network

The material discussed in this appendix stems from a collaborative project with L Zhou (then a PhD candidate, Concordia University), and some of it was presented at the 2006 IBPSA-US conference at MIT in June 2006. Further investigations are required to analyze the full implications of the coupling approach described herein. This appendix discusses the basics of the approach and some of its motivators, and describes how it has been applied to the PEC (Personal Environmental Controls) system model used in this dissertation.

A2.1 Motivations and Precedents

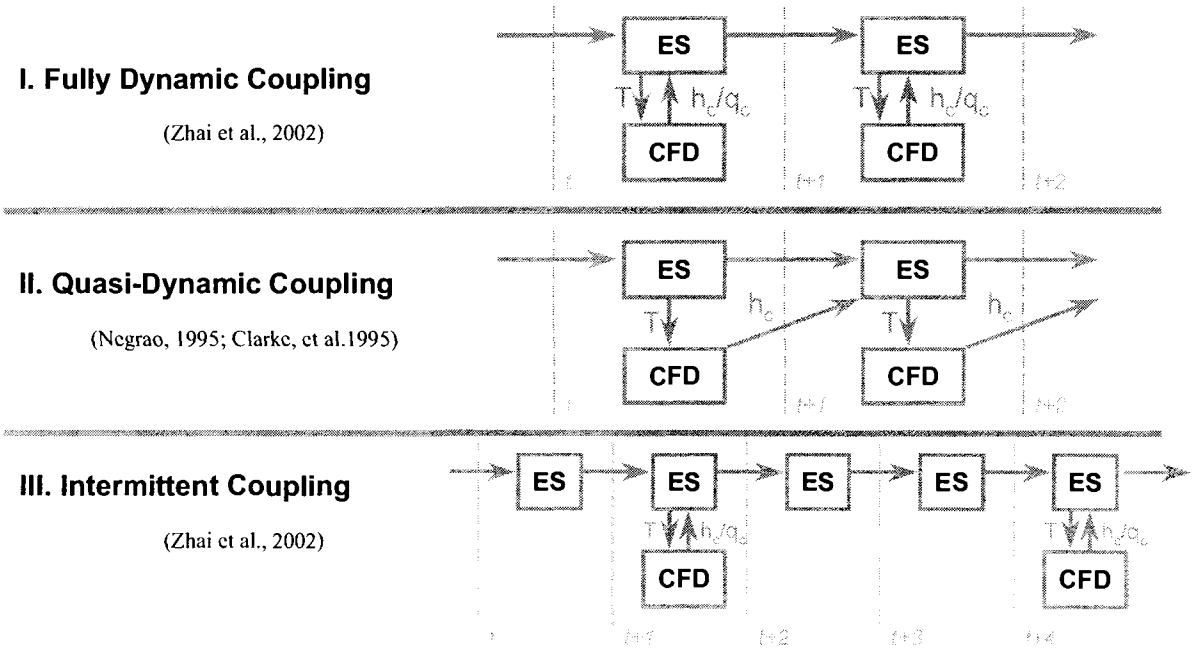
The Personal Environmental Controls system in Place-du-Portage 8B1 is an interesting system, and offers the potential for both comfort improvements and increased energy efficiency. It also happens to have a characteristic that makes it very difficult to model effectively with currently available energy simulation software – the air in the space is not well-mixed, and significant temperature gradients exist both horizontally and vertically throughout the space. All of the major macro-scale transient energy simulation software currently available (e.g. DOE2, TRNSYS, EnergyPlus, ESP-r) work with the assumption of a uniform temperature distribution in the space, and calculate their heat balances accordingly. On the other hand, CFD simulation does not make this assumption, but because of computational capacity restrictions we cannot run transient CFD simulations for any substantial simulation period without supercomputers. So this is one

example, among many, where it seems quite fruitful to somehow combine the benefits of CFD and macro-scale energy simulation by coupling the two programs.

This desire to couple CFD and ES (energy simulation) has been discussed in a few cases (as noted below), but has received very little research attention thus far. It should, however, be receiving much more attention in the near future – partly due to increases in computation power, partly due to the increased interest in ventilation systems that produce significant temperature gradients (such as displacement systems, or the ceiling jet diffusers of 8B1), and partly due to the new NIST project to investigate the embedding of micro-scale models of airflow and temperature distributions into macro-scale models.

With the exception of J Axley's recent work with NIST, the previous research in this field has focused on the use of CFD software and ES software running side-by-side and passing information back and forth (rather than a true mathematical integration of the micro into the macro, which is what Axley is attempting to do with his studies). The major coupling approaches considered in these previous studies were summarized by Zhai et al (2002), and they are outlined in Figure A2.1. They all use full CFD simulations alongside the ES simulation, differing only in how often the CFD simulations are run. Another approach was noted by Zhai et al (2002), but not discussed in detail nor given examples, called "virtual dynamic coupling". This involves the pre-creation of a interpolative-lookup database with CFD inputs and outputs, and then using this throughout the energy simulation.

Figure A2.1: Summary of previous CFD-ES coupling approaches



Another question, and likely the more significant question, is 'what information is being passed between the CFD and the ES?' In most of the previous studies, this information passing focused on the boundary conditions of the CFD model and the heat transfer coefficient. The main role of the CFD within the energy simulation was to give a better estimate of the h_c value than could be done by the default approaches of the energy simulation software. The CFD also produced valuable information about comfort parameters within the space, which are considered as outputs to the combined model, but which were not passed to the ES program for use in its calculations (as it was not required within the ES models).

Since the coupled CFD-ES model for the PEC system is being used for optimization studies and for studies of simulation-based control, it must be able to run much faster than would be possible with an approach that used CFD runs during the ES simulation.

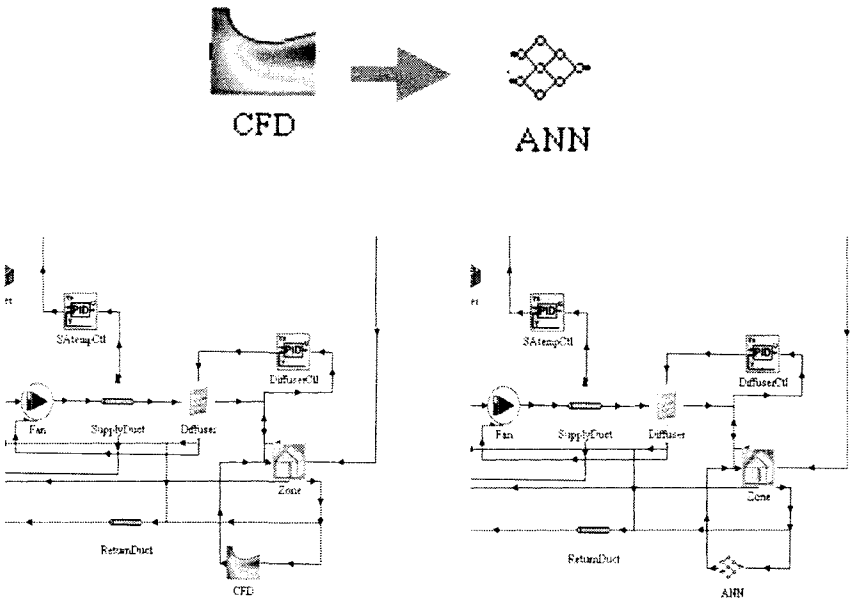
So the approach that we have developed for this model is akin to the "virtual dynamic coupling" noted by Zhai et al. (with a 'fully dynamic' coupling between the ES simulation and the lookup table). But instead of using a pre-filled interpolative lookup table in place of the concurrent CFD simulations, we are using a pre-trained ANN, because it allows for greater accuracy and is more easily generalizable for use in other modeling cases. And in this case, we are also interested in more CFD outputs than just the h_c value, since we need to know some point temperatures in the space to properly model the diffuser control (we need the point temperature at the thermostat) and the cooling load (we need the temperature of the return air).

A2.2 General description of approach

The overall goal of the approach may be described by the illustration in Figure A2.2. The snapshot of a TRNSYS model in the bottom left corner of the figure shows how CFD and TRNSYS would normally be coupled – Fluent (CFD software) would be sent inputs and provide outputs as if it were a TRNSYS module. (This is facilitated by a new TRNSYS module created by Diego Arias of the Solar Energy Lab at UWisc, and released in spring 2006, that allows TRNSYS to call Fluent as an external program, similar to the modules that allow for external calls to Matlab or Excel.) The goal is to train an ANN to behave the same way that the CFD module would behave, in terms of its inputs and outputs, and then use this ANN as a module within TRNSYS. The loss: a bit of accuracy (how much is at this point unclear, but it depends on the quality of the ANN training, and we expect it to be very small for most of the aspects under consideration). The gain: for the office space under consideration, each call to the CFD module would take about 4-5

hours to compute, while each call to the ANN module would take far less than a second – this difference is the key to making transient coupled simulations computationally feasible.

Figure A2.2: Overall goal of the CFD-to-ANN approach



Two challenges then arise: (1) effectively training the ANN to mimic the CFD outputs; and (2) appropriately linking the ANN module with the rest of the TRNSYS model. The first challenge has precedents in the training of ANNs with CFD outputs for use in optimization studies, and is discussed in the next section. The second challenge is shared by other CFD-ES coupling strategies, and is discussed in section four.

A2.3 PEC system model: Creating the CFD database and training the ANN

The CFD model of the PEC system had been previously created by L Zhou for use in an optimization study (Zhou et al, 2005 and Zhou et al, 2006). She also performed the Fluent runs and the ANN training in Matlab for this study.

The first step in training the ANN involves the determination of appropriate input and output variables. This process involves some give-and-take with the creation of the complete TRNSYS model, and a number of test cases were run until the variables listed in Table A2.1 were decided upon:

Table A2.1: Input and output variables for the ANN

Inputs
Internal window surface temperature
Airflow rates for the diffusers
Supply air temperature

Outputs
Sensor temperature at each workstation
Return air temperature
Average room temperature
Inside window convective heat transfer coefficient
PMV for each workstation

Appropriate ranges of values were determined for each input variable, and latin hypercube sampling was used to select 30 input value configurations. These sets of inputs where then run through 30 Fluent simulations, and the results used to create a database of input and output values. An ANN structure of one hidden row with 20 nodes was used (again chosen through various trials), with transig as the first transfer function and a linear function as the second. The ANN was then trained using the database and Levenberg-Marquardt back-propagation with Bayesian regularization.

The resulting set of weights and biases were then coded into a Fortan module for use in TRNSYS. This ANN module is thus specific to this PEC model, but it can be easily modified to use different weights and biases to simulate different systems in different models.

A2.4 PEC system model: Using the ANN within TRNSYS

One of the main difficulties with any type of CFD coupling with energy simulation is that the CFD simulations are steady-state, and the energy simulations are transient. As such, the CFD simulations do not take into account the thermal capacitance of the air in the space, and so the T_{bulk} calculated by the CFD (the average of all the point temperatures, excluding the air within a small distance from the walls, ceiling and floor) tends to shift to changes in boundary conditions much more drastically than does the T_{bulk} calculated by the energy simulation. And the CFD cannot take into account fluctuating bulk temperatures that may occur on purpose (e.g. night setback and demand trimming). However, it is felt that the general shapes of the temperature gradients predicted by the CFD would be fairly accurate, as long as the T_{bulk} of the CFD is not too far off what the actual T_{bulk} would be. (Further study is required on this point, as noted in the discussion in section five.) With this reasoning, it was determined that the best way to estimate the point temperature at the sensor was to use the bulk temperature determined by the energy simulation (we trust this more because it does consider the thermal mass of the air), and modify it according to what the CFD simulation predicts for the relative temperatures in the space. The point temperature at the sensor is calculated as shown in Equation A2.1.

$$T_{sensor} = T_{bulk\_Type19} + (T_{sensor\_ANN} - T_{bulk\_ANN}) \qquad \text{(Eq A2.1)}$$

The same idea is also used to calculate the return air temperature. Thus, the ANN is added to the TRNSYS simulation as a module that aids in the estimation of T_{sensor} and T_{return}, and also provides information about comfort parameters that would not be available through TRNSYS alone. Since it is treated as an add-on in this way, its outputs can be over-ridden if they are deemed inappropriate. This over-ride feature has been used in the model to account for conditions that are outside of the range of conditions used in creating the CFD database to train the ANN – in such cases, average values are used for the temperature outputs, which result in the return air temperature being approximately 0.5°C higher than the bulk air temperature, and the sensor temperature being approximately 0.5°C below the bulk air temperature.

Using CFD to provide an estimation of h_c is an interesting dilemma. CFD should be able to do this much better than the default approach that is used in energy simulations (usually just using a constant h_c value throughout the simulation), but most CFD software has significant difficulties in dealing with the near-wall regions in mixed-flow regimes, which is the case in most buildings applications. (The wall functions used in CFD software were generally developed for forced flow applications, which is where most CFD is used – pipes, ducts, aeronautics, engines, etc). So the h_c values that were used to train the ANN in this study were derived from the temperature and airflow outputs of the CFD simulation using a correlation for mixed-flow heat transfer developed by Ian Beausoleil-Morrison, and which he recommends for use when performing CFD-ES coupling (Beausoleil-Morrison, 2000). This correlation is shown in Figure A2.3.

Figure A2.3: Mixed-flow correlation used for the convective heat transfer coefficient

$$h_c = \left(\left\{ \left[1.5 \cdot \left(\frac{T_{surf} - T_{air}}{H} \right)^{1/4} \right]^6 + \left[1.23 \cdot \Delta T^{1/3} \right]^6 \right\}^{(3.1/6)} + \left\{ \left[\frac{T_{surf} - T_{diffuser}}{T_{surf} - T_{air}} \right] \cdot \left[-0.199 + 0.19 \cdot (ac/h)^{0.8} \right] \right\}^3 \right)$$

The ANN has been set up within the TRNSYS model to allow the h_c value to be passed to the Type19 zone bulk air module. And the Type19 module sends the $T_{surface}$ value to the ANN, which it turn affects the h_c output, which affects the $T_{surface}$, etc.. As such, TRNSYS must iterate over these values at each timestep until they converge. This usually works fairly well, but it slows down the simulation and occasionally causes problems when the solution fails to converge. As such, the h_c output from the ANN has been over-ridden by a constant default h_c value for the study of simulation-based control for demand response discussed in this dissertation.

A2.5 Discussion

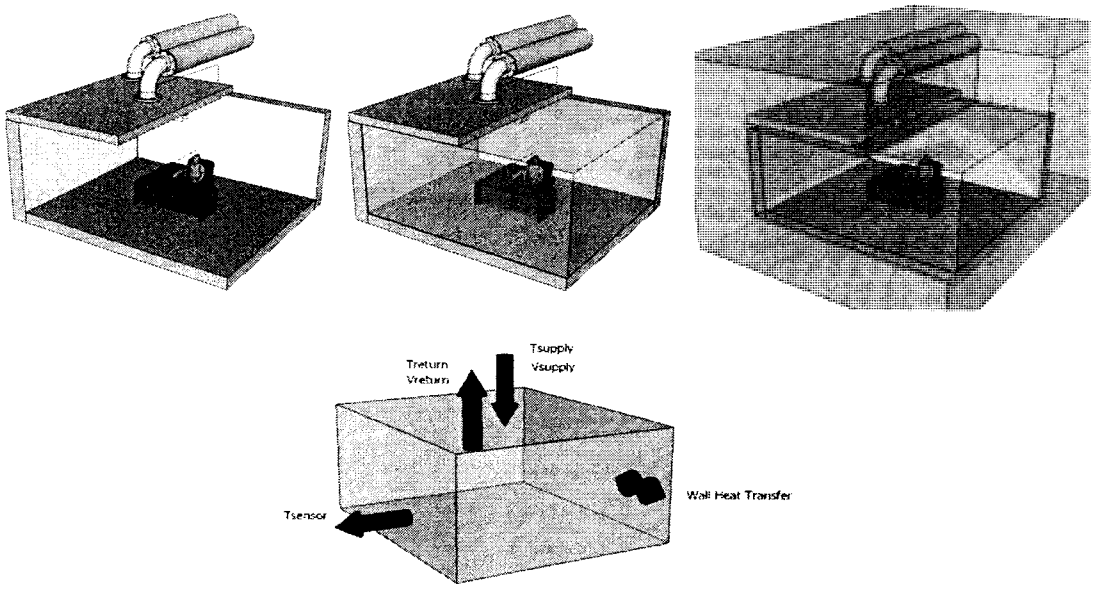
Other possible uses for this combined TRNSYS-ANN-CFD model of the 8B1 PEC system include comparisons with other types of diffuser systems, and design optimization studies. One aspect of this approach, however, makes design optimization quite difficult – whenever you want to consider a different physical configuration of diffuser or occupant location, a whole new database of CFD outputs would have to be created to train a new ANN. Control optimization, on the other hand, is working with an existing system, so just one database and one ANN is all that is required.

One of the main dilemmas with the chosen approach is that it involves two different values for the bulk air temperature of the space – one calculated by the energy simulation

software, and one calculated by the CFD software. We think that it is a reasonable choice to use the T_{bulk} from the energy simulation and then adjust it according to the relative values in the CFD simulation. However, this may not be appropriate if the T_{bulk} from the two different programs is substantially different. It is unclear how big of a variation is required to create a significant error in the results. But this concern should be noted in the context of the demand response study, since it involves temperature setpoint fluctuation, and thus could result in cases where the temperature distribution pattern predicted by the ANN is not appropriate.

One possible way of avoiding this dilemma in future studies is just to avoid having two separate values for the bulk air temperature of the space. What if the ANN is used to replace the room air node in the energy simulation? This idea is shown in Figure A4, where the blue box represents the CFD (ANN) domain, and the remainder is covered by the ES domain. The bottom of the figure shows the information exchange between the ANN and the rest of the TRNSYS modules. The key aspect is that the two domains do not overlap, as they do in the approach that we have chosen to use above.

Figure A2.4: Illustration of the "ANN as room node replacement" approach



This approach is conceptually cleaner, but it has does have its own problems. It requires a rewriting of some TRNSYS modules, since they can no longer contain the room air node within their domain. But the bigger problem is that the ANN must somehow incorporate the consideration of the zone air's capacitance.

In general, the coupling of CFD and energy simulation remains an important and interesting area of research, with many open questions. The ANN approach discussed here seems pragmatic, and has worked fairly well in the current case study. Further research should be undertaken to investigate the full implications of this approach in terms of accuracy and computation time.

Appendix 3: Automated Model Calibration

Modeling an existing building is inherently different than modeling a building in the design stage. While the validity of models for paper buildings is based primarily on the computational method, the experience of the modeler, and the confidence in the modeling tools, the measure of model performance in the case of existing buildings is simpler – how well do the model's outputs compare with the actual building's behaviour? The modeler knows this criterion from the outset, and in most cases has the actual energy data on hand for quick comparisons as the building is modeled.

The creation of such models is often thought of in two stages: the first involves laying out the essential geometries and component descriptions using what seem like reasonable parameter values; and the second step involves comparing the model outputs with the actual data and tweaking some of the model's parameters until the error between the two is within some reasonable range. This process of calibrating the model is usually done manually, and is usually more of an art than a science, with different modelers using different approaches (and favouring different parameters) to tweak the model.

This two-step approach is compared with the case of inverse ('grey-box') modeling or 'black-box' modeling, where the first step is essentially eliminated, and the second step is fully automated. These methods focus completely on the model inputs and outputs, and can often produce more accurate models than the more detailed approach, while also taking less time to create and less computation time per simulation. But these models do not offer any help in explaining unusual phenomena, nor are they of much use in analyzing possible changes in building configurations.

In an effort to keep the benefits of a detailed building-physics modeling approach, while also increasing model accuracy, some researchers have begun to develop automated calibration techniques. Once the first stage of modeling is complete, and the basic structure of the model is felt to be accurate, the subsequent process of calibration is essentially just an optimization problem: the parameters are the optimization variables and the minimization of an error metric is the objective. Questions remain, however, about how best to set up the optimization problem and how to solve it. In most building models, there are a large number of parameters that may be considered for calibration, resulting in a very large search space for the optimization algorithm. And the more imposing dilemma is that, in most calibration problems, there are far more parameters than there are output data points upon which the error metric is based – this results in there being many different (potentially infinite) solutions to the minimization problem.

This appendix examines an approach to automated calibration that helps to avoid (or at least mitigate) the underdetermined-problem dilemma, while also allowing the model to be used in building diagnostics. Automated systems for 'continuous commissioning' or 'building diagnostics' are also currently receiving some research interest (see, for example, DABO, 2008 and Xu et al, 2005), and some of these systems use building (and/or system and/or component) models to compare the actual building behaviour to what it should be. The overlap between this endeavor and that of automated calibration has yet to receive much research attention.

The basic approach is to have the modeler note uncertainty metrics along with their default values for each parameter, and then incorporate these into the optimization objective function. This reflects part of the intuitive approach used in manual calibration,

and produces greater differentiation between points in the optimization search, limiting the possibilities for multiple solutions, and helping to ensure that the solution reached is a reasonable representation of what is actually happening in the building. If the model's structure is thought to be correct, then if after calibration with this approach a parameter value is showing a large deviation from the default value (relative to the uncertainty metrics), then this suggests that it might be worth investigating the behaviour of the physical counterpart of this model parameter to ensure that everything is operating as it should be. (If so, then the model can be fixed. If not, well, I guess that is the point of diagnostics.)

This approach has been carried out with the model used in this dissertation. The first section describes how the calibration problem is incorporated within the TRNSYS model, and how the basic error-minimization optimization problem is set up and solved using GenOpt. The second section provides a more detailed description of the calibration approach being used. The new objective function is incorporated into the model and the new optimization problem is solved. The results are presented in section three. This leads into a broader discussion of various aspects of this approach, including how it can be incorporated within the simulation-based supervisory control software framework, and possible implications of this approach to building simulation.

A3.1 Calibration as an Error-Minimization Optimization Problem

To facilitate the process, the numerical values originally used for these parameters in the model input file are replaced in the text by a variable name, and this variable name is defined and given its value near the top of the file. (This is not strictly necessary, but it

helps to organize the process somewhat.) One could, at this point, simply run a collection of simulations by manually modifying these parameter values, recording the hourly electrical consumptions outputted by the model and then comparing these with the measured data to find the sum of errors or the root-mean-squared-error (RMSE), or whatever else is being used to measure the accuracy of the model. But this calculation process can be simplified by doing the error calculation within the TRNSYS simulation itself. Another text-file reader is added to the model file so that it can read in the measured data. The error calculation is done within the model, and this value is output to another text file. So now one need only modify the parameter values at the top of the model file, run the simulation, and then read the error metric from the output file.

Since one is simply tweaking these parameter values in search of the configuration that produces the least error, it is easy to see this as an optimization problem with the parameter values as optimization variables and the error metric as the objective function to be minimized. And the way that it is set up within the TRNSYS model makes it easy to use GenOpt to run an optimization algorithm to find the best configuration. The parameter values near the top of the TRNSYS model file are replaced with variable demarcations (%x%) recognizable by GenOpt. The input files for GenOpt can then be configured appropriately, and the optimization problem is fully prepared.

A3.2 Incorporating Uncertainty Metrics into Automated Calibration

Some research in automated calibration has begun to address the problem of having a very large search space and multiple possible solutions by finding reasonable ways of limiting the number of parameters under consideration. The work of Mara et al (2001) considers sensitivity analysis as a way of identifying which parameters are most influential. Sun and Reddy (2006) combine this sensitivity analysis with a correlation analysis to further limit the list of parameters. And the various methodologies that have been developed for manual calibration (as summarized by Reddy (2006), including ways of splitting the parameters into weather-dependent and weather-independent and then using different periods of data to calibrate the two sets separately or one after the other) can also been seen as ways of limiting the number of parameters under consideration in order to make the calibration problem tractable. But the concerns about setting ranges for parameter values, or tradeoffs between error values and the reasonability of parameter values, have thus far been neglected.

These concerns can be addressed by modifying the objective function slightly, so that the optimization algorithm considers not only the value of the error metric, but also considers how far the parameter values are deviating from their expected values. Instead of just considering a range of possible values for each parameter, the possible parameter values can be thought of as a bell curve with an expected value μ and a standard deviation σ. Parameter values closer to μ are preferred over values further away (relative to σ), so a penalty is added to the objective function that is related to the deviation from μ.

The objective function in the original minimization problem studied in the previous section is as shown in Equation A3.1. It is simply equal to the error metric that is being used.

$$Z = \text{Error} \qquad \text{(Eq A3.1)}$$

Equation A3.2 shows the incorporation of parameter deviations into the objective function.

$$Z = w_0 \cdot f_0(\text{Error}) + w_1 \cdot f_1\left|\frac{p_1 - \mu_1}{\sigma_1}\right| + w_2 \cdot f_2\left|\frac{p_2 - \mu_2}{\sigma_2}\right| + \ldots \qquad \text{(Eq A3.2)}$$

where p_i is the value under consideration for parameter i, μ_i is the expected value for that parameter, and σ_i is its standard deviation. The weighting factors w_i and functions f_i are included to allow the user to provide relative emphasis to either the error minimization or to minimizing the deviation of one or all of the parameters.

A3.3 Results with PEC Model

This method was applied to the PEC model described in Chapter 4. Table A3.1 shows the expected values, deviations and weights that were used in configuring the optimization process. The weighting values were chosen fairly arbitrarily, but were intended to help offset the fact that the error metric value is significantly larger than the deviation of any of the parameters. And a squared function was used to emphasize the penalty for the error metric or any deviation being particularly large.

Table A3.1: Expected values, deviations and weightings for each parameter

parameter	μ	σ	weight
Induction supply (m^3/s)	0.1	0.02	0.1
Lights, general (kJ/hr)	216	50	0.1
Lights, occupant (kJ/hr.occ)	36	7.5	0.1
Computer gain (kJ/hr.occ)	36	7.5	0.1
K1 (infiltration coefficient)	0.1	0.02	0.1
Room capacitance (kJ/°C)	300	50	0.1
Solar transmittance (ratio)	0.7	0.1	0.1
Error	-	-	1.0

The optimization algorithm used was a modified genetic algorithm that uses an ANN-optimization as an evolutionary operator. (The algorithm trains an ANN with previous points in the search so that the ANN gradually becomes a good approximation of the objective function, producing a gradient-like operator. The algorithm was developed for use within the simulation-based supervisory control framework, but has not yet been tested in that regard.) The algorithm configuration is shown in Table A3.2.

Table A3.2: Configuration of GA-ANN-entwined algorithm

Population size	20
Mating pool selection	None
Crossover	One-point
Mutation	Uniform
Next generation selection	Fitness proportional with elitism
Ratio crossover	0.8
Ratio mutation	0.1
Ratio elite	0.1
Learning rate for ANN	0.1
Number epochs for ANN	500
Max simulations	1000

After one thousand simulations, the algorithm identified the values shown in Table A3.3 as the most appropriate values that it could find in its search.

Table A3.3: Minimum point found by the algorithm

Induction supply (m^3/s)	0.11
Lights, general (kJ/hr)	232.0
Lights, occupant (kJ/hr.occ)	37.2
Computer gain (kJ/hr.occ)	34.8
K1 (infiltration coefficient)	0.106
Room capacitance (kJ/°C)	400
Solar transmittance (ratio)	0.64

A3.4 Discussion

Although this approach does not eliminate the problem of having to solve very large optimization problems because of the many parameters under consideration, it does help to alleviate concerns over there being many mathematically-valid solutions to the problem. For any two parameter value configurations that produce the same error metric, the objective function will favour the configuration that is closer to the μ values provided for each parameter. So while multiple solutions to the optimization problem are still possible, this approach should limit these multiple solutions, and should automatically weed out the most unreasonable ones.

To deal with the size of the optimization problem, and to try to limit the number of simulations that must be run to find a solution, the approaches of sensitivity analysis and correlation analysis originally suggested to limit the parameters under consideration could instead be used within the optimization approach itself. Identifying the most sensitive parameters could allow for a two-stage optimization approach: in the first stage the less sensitive parameters could be frozen at their default values while the sensitive parameters are modified to find near-optimal values for them; and then the algorithm could un-freeze the less sensitive parameters to optimize for the entire set, but would now only have to consider a smaller range around where the sensitive parameter values ended

up during the first stage. The optimization challenge can also be addressed just through more efficient algorithms that require fewer simulation calls, like the GA-ANN-entwined algorithm used in this case.

As is the case in most optimization problems involving weighted objective functions, providing appropriate values for the weights is key to the success of the optimization approach. A tradeoff would have to be struck between the emphasis on the minimization of deviation and the minimization of error, and to do so requires some manipulation of the weighting values. Further research is required to find rules or guidelines to help in the selection of appropriate weighting values.

And there is much more that can be done with the functions instead of just squaring each of the terms. Of particular interest is the function f_0, associated with the error metric. We may wish to use a function for this that is quite flat for values that are close to zero, and which increases quite quickly beyond some error band that we deem acceptable. How big to make this band is fairly arbitrary, but we could use some advice from ASHRAE as a guide: Reddy (2006) notes that the ASHRAE Guideline 14 provides the somewhat arbitrary accuracy measure of "models are declared to be calibrated if they produce NMBEs [normalized mean bias error] within 10% and CV-RMSEs [root mean squared error] within 30% when using hourly data, or 5% to 15% with monthly data."

Another way of approaching the automated calibration problem through the use of uncertainty metrics might be to consider it as a two-stage process. In the first stage, a multi-modal optimization algorithm could search out as many solutions as possible that fall within the desired error band. In the second stage, the best solution among these is chosen based on the minimization of the deviations from the expected values. Although

using these two separate optimization procedures is perhaps somewhat less elegant than using just the one, and does not allow for more differentiation between solutions that have close to zero error and those whose errors are close to the edge of the acceptable range, this two-stage approach is nonetheless more transparent and does not run into the same difficulties in trying to set appropriate weighting values. So it may be worth further investigation.

The relationship between the calibration approach discussed here and model-based approaches to diagnostics deserves further investigation. Including deviations within the objective function forces the optimization towards solutions that are closer to the expected values of the parameters. So if the optimization still finds that the measured data is suggesting that a parameter value is other than what is expected, then something might be wrong. This something could be the model structure or the expected value for the parameter, or it could be that something is wrong within the actual system. So a flag could be thrown by the a post-calibration processor if it finds a calibrated parameter value that is beyond some threshold deviation from its expected value.

Finally, it is worth noting how this approach to model calibration can be used within the simulation-based supervisory control framework. Simulation-based control is only effective if the model being used is accurate, so calibrating it well and keeping it up to date are both very important. This automated calibration approach can use the same data that is being collected and stored by the organizational layer in the software framework, and the calibration can be run on the model periodically (e.g. once per month, with the calibration running overnight when the supervisory control system is not needed as urgently). The building model in the software framework is in template form, with %x%

being used to demarcate the control set-points inputs and x marking the external conditions. SimCon inserts the x variables before sending it to GenOpt, which inputs the %x% variables. The calibration process could be thought of as working one layer above SimCon, with the model parameters (perhaps demarcated by #x# in the super-template) inserted by the calibration process before sending it to SimCon for use.

With all of these variable demarcations within the model files, the model begins to look less like a definite model and more like a general structure or archetype into which particular values are placed as required, depending on the circumstances. This more flexible view of models corresponds well with a more flexible view of buildings themselves, and it may be interesting to see to what extent this flexibility can be carried.

The other possible implication of this calibration approach, in terms of its relationship with building simulation in general, is that it further pushes the envelope for the development of modeling techniques that can account for uncertainty in model parameters and model inputs. It would be ideal if our building models could take into consideration the uncertainty in weather predictions (e.g. for control optimization, or for economic analysis of investments and their payback periods). One may also wish to note error bands in their parameter values (e.g. in trying to guesstimate the R-value without knowing the construction details), and have these carried through the simulation. The computational challenges associated with such modeling with uncertainty are substantial, so these techniques are unlikely to be developed soon. But further developing this relatively simple use of uncertainty metrics in model calibration could be a good start.