Title
Utilizing Problem Structure in Optimization Algorithms for Model Predictive Control

Author
Kelman, Anthony David

Publication Date
2015

Peer reviewed|Thesis/dissertation

Utilizing Problem Structure in Optimization Algorithms for Model Predictive Control

by

Anthony David Kelman

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Francesco Borrelli, Chair
Professor Andrew Packard
Professor James Demmel

Fall 2015

Utilizing Problem Structure in Optimization Algorithms for Model Predictive Control

Abstract

Utilizing Problem Structure in Optimization Algorithms for Model Predictive Control

by

Anthony David Kelman

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Associate Professor Francesco Borrelli, Chair

In this work we perform control design and demonstrate the effectiveness of model predictive control (MPC), an optimization based control approach that is capable of satisfying state and input constraints and using forecasts of disturbance inputs, for the application of energy efficient control of heating, ventilation, and air conditioning systems in buildings. We derive simplified control oriented models and express the relevant constraints and disturbance predictions, and show that online solution of the resulting optimization problems is able to reduce energy consumption while satisfying occupant thermal comfort constraints and limits on control actuator inputs.

We investigate the implementation challenges of applying model predictive control to large systems, focusing on online solution of an optimization problem at every time step. We show that it is critical to take advantage of problem structure in system modeling and the formulation of the constraints and objective function. Applying an interior point algorithm making use of parallel sparse linear algebra solvers performs well, solving nonlinear MPC problems with tens of thousands of variables and constraints in less than a minute on modern multicore processors. We design an optimization modeling tool that allows simple expression of a MPC problem and efficient interfaces to compiled optimization solvers, calculating sparse derivatives and constraint Jacobians automatically. Finally we examine specialized optimization algorithms for linear systems with polyhedral constraints, reusing repeated model data for time invariant systems to solve block banded linear systems of equations.

Contents

List of Figures

List of Tables

Acknowledgments

This work has been partially supported by United Technologies Research Center (UTRC) under grant W912-09-C-0056 and by U.S. Air Force Office of Scientific Research (AFOSR) under grant FA9550-09-1-0106. This material is based upon work supported by the National Science Foundation under Grant No. 1239552. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to thank my committee, labmates, collaborators, family and friends.

Chapter 1

Introduction to Model Predictive Control

Many problems in control design are naturally expressed in terms of dynamic optimization. Starting from a mathematical model of how a system evolves over time as a function of its current states x, controlled inputs u, and uncontrolled disturbances w, we would like a strategy for selecting u in such a way that the system minimizes (or maximizes) some objective function while satisfying a set of constraints.

The dynamic model of a system may be specified implicitly as a set of differential algebraic equations. For continuous time systems we can write the dynamic optimization problem as follows.

$$\min_{\mathbf{U},\mathbf{X}} \int_0^T J(x(t), u(t), w(t), t)dt \tag{1.1a}$$

$$\text{subj. to, } \forall\, t \in [0, T],$$

$$f(\dot{x}(t), x(t), u(t), w(t), t) = 0 \tag{1.1b}$$

$$g(\dot{x}(t), x(t), u(t), w(t), t) \leq 0 \tag{1.1c}$$

$$x(0) = x_0 \tag{1.1d}$$

The initial state x_0 is treated as a known constant value, either measured or estimated from the present state of the system. The time horizon over which the optimization is performed is denoted T. The cost function is $J(x, u, w, t)$ and the dynamic evolution of the system is captured in the equality constraint $f(\dot{x}, x, u, w, t) = 0$. An explicit ordinary differential equation is a special case where $f(\dot{x}, x, u, w, t) = \dot{x} - f_{ex}(x, u, w, t)$. The minimization is performed over the space of input trajectories $\mathbf{U} = \{u(t) \mid t \in [0, T], u(t) \in \mathbb{R}^m\}$ and state trajectories $\mathbf{X} = \{x(t) \mid t \in [0, T], x(t) \in \mathbb{R}^n\}$. In a nominal predictive control problem, we assume the disturbance trajectories $w(t)$ follow known forecast values.

For discrete time systems we write

$$\min_{\mathbf{U},\mathbf{X}} \sum_{k=0}^{N-1} J(x(k+1), x(k), u(k), w(k), k) \tag{1.2a}$$

$$\text{subj. to, } \forall\ k \in \{0, \ldots, N-1\},$$

$$f(x(k+1), x(k), u(k), w(k), k) = 0 \tag{1.2b}$$

$$g(x(k+1), x(k), u(k), w(k), k) \leq 0, \tag{1.2c}$$

$$x(0) = x_0 \tag{1.2d}$$

where the optimization time horizon is N steps. In discrete time the minimization is performed over the space of input trajectories $\mathbf{U} = \{u(k) \mid k \in \{0, \ldots, N-1\}, u(k) \in \mathbb{R}^m\}$ and state trajectories $\mathbf{X} = \{x(k) \mid k \in \{1, \ldots, N\}, x(k) \in \mathbb{R}^n\}$. The disturbance trajectories $w(k)$ are assumed to follow known forecast values as in the continuous time setting.

In the general case there may not be a closed form solution to problems of the form (1.1) or (1.2), so a numerical solution approach will be necessary. The space of input and state trajectories $\mathbf{U}, \mathbf{X}$ in the continuous time problem (1.1) is infinite dimensional, so some form of finite dimensional parameterization is necessary for numerical solution. Either the value domain or the time domain can be discretized to achieve this.

Discretizing in the value domain can be achieved by representing the continuous time trajectory spaces in terms of a series of basis functions, then expanding the differential equation in terms of those basis functions. The optimization can then be performed over the parameters of the basis decomposition, where a finite truncation of the basis series results in a parameterization that approximates the true optimum trajectories. This is commonly performed using orthogonal polynomials as basis functions, however orthogonal polynomials are not necessarily closed for nonlinear differential equations.

Discretizing (1.1) in the time domain transforms it into a finite dimensional problem of the form (1.2). Explicit continuous time ordinary differential equations can be approximated as (either explicit or implicit) discrete time difference equations via multistep or Runge Kutta methods.

The above dynamic optimization problems (1.1) and (1.2) are open loop constrained finite time optimal control (CFTOC) problems. Disturbance predictions are made ahead of time for the entire horizon, and state measurements are available only for the initial point. If the optimal input trajectory were applied for the entire prediction horizon length, model mismatch and inaccuracy in disturbance predictions would accumulate and could cause divergence from the predicted optimal state trajectory. Since newer state measurements and disturbance prediction data will become available at later times in the horizon, it is beneficial to re-plan a new CFTOC problem with a shifted horizon.

In control applications we are often interested in the limit as the horizon length goes to infinity, for steady state tracking behavior or systems that remain operational much longer than the characteristic time constant of their dynamics. Model predictive control (MPC) is the closed loop receding horizon application of repeated solutions of CFTOC problems.

Using newly measured state data as the initial point in each optimization problem makes MPC a feedback control scheme. The initial values of the optimal input trajectory are applied to the system then a new optimization problem is solved at the next time step.

The natural statement of an MPC problem in terms of dynamic optimization means hard constraints on inputs and states can be rigorously enforced while optimizing the desired control objective, large numbers of input and state variables can be handled in the dynamic system model, and predictive forecast knowledge of future uncontrolled disturbance inputs can be utilized to improve system performance. The disadvantage of MPC is that it requires solving an optimization problem over an entire prediction horizon at each time step. That optimization problem must be solved in real time within the length of a time step so the first optimal input value can be applied to the system.

In this thesis we will discuss the application of model predictive control to energy efficient control of buildings, and the performance advantages and implementation challenges in applying MPC to this type of system. We will examine the details of optimization formulation, algorithmic solution, and linear algebra scalability to the general problem of applying MPC to large systems, and taking advantage of modern computational platforms.

In chapter 2 we derive the system model for an optimization formulation of the problem of maximizing efficiency in heating, ventilation, and air conditioning (HVAC) systems in buildings. Simulation and experimental results demonstrate sophisticated closed loop behavior that resembles existing state of the art digital control strategies, but does not require manually writing out the implementation logic for when to apply or how to tune these heuristics. Applying a numerical optimization solver to the problem identifies energy efficient inputs based entirely on the system model, constraints, and objective function. We investigate the nonlinearity of the system and its consequences in terms of local optima of the MPC optimization problem.

In chapter 3 we discuss the details of optimization algorithms for large scale MPC, and computational challenges for real time optimization. Optimization algorithms using parallel sparse direct linear algebra are shown to perform well on MPC problems with tens of thousands of variables and constraints, solving a nonlinear constrained optimization problem in under a minute on modern multicore processors.

In chapter 4 we design a tool that allows a user to specify a dynamic system model, objective function, and constraints in a direct manner. The tool, called the Berkeley Library for Optimization Modeling, is responsible for extracting the problem structure, transforming the user model input into a sparse multivariate polynomial parameterization. This parameterization allows for derivatives to be automatically calculated in closed form by efficient compiled optimization solver interfaces. The user does not have to carry out the tedious and error prone task of symbolic differentation of their objective and constraint functions. Sparse Jacobians and Hessians can be calculated efficiently from the polynomial representation, resulting in good performance even for large nonlinear systems.

We conclude by looking at the characteristic block banded problem structure arising from propagation of system dynamics over the time horizon in MPC optimization problems. This structure can be exploited in the linear algebra operations that are the computational

bottleneck at each iteration of an optimization algorithm. For time invariant systems in particular, we show how the constant repeated problem data of the linear system model matrices can be exploiting when using iterative linear algebra techniques to solve the linear systems of equations to determine a descent direction.

Chapter 2

Application to Energy Efficient Buildings

We study the problem of heating, ventilation, and air conditioning (HVAC) control in a typical commercial building. We propose a model predictive control (MPC) approach which minimizes energy cost while satisfying occupant comfort and control actuator constraints, using a simplified system model and incorporating predictions of future weather and occupancy inputs.

Extensive numerical simulations show the effectiveness of the proposed approach. In particular, the MPC is able to systematically reproduce a variety of well-known commercial solutions for energy savings, which include demand response, "economizer mode" and precooling/preheating.

In simplified physics-based models of HVAC systems, the product between air temperatures and flow rates arising from energy balance equations leads to a non-convex MPC problem. Fast computational techniques for solving non-convex optimization can only provide certificates of local optimality. Local optima can potentially cause MPC to have worse performance than existing control implementations, so deserve careful consideration. The objective of this chapter is to investigate the phenomenon of local optima in the MPC optimization problem for a simple HVAC system model.

In the first part of this chapter, simplified physics-based models and MPC design for two common HVAC configurations are introduced. In the second part, simulation results exhibiting local optima for both configurations are presented. We perform a detailed analysis on the different types of local optima and their physical interpretation. We then use this analysis to derive physics-based rules to rule out classes of locally optimal control sequences under specific conditions.

This chapter is based on work that has been previously published in [35] and [37].

2.1 Introduction

The building sector consumes about 40% of the energy used in the United States and is responsible for nearly 40% of greenhouse gas emissions, see [50]. It is therefore economically, socially and environmentally significant to reduce the energy consumption of buildings. Previous work by [29, 47, 53] has evaluated the energy saving potential of model predictive control (MPC) for heating ventilation and air conditioning (HVAC) in buildings.

This chapter focuses on model predictive control (MPC) of HVAC systems over networks of thermal zones. The main idea of model predictive control is to use a model of the plant to predict the future evolution of the system [11, 49]. At each sampling time, an open-loop optimal control problem is solved over a finite horizon. The optimal command signal is applied to the process only during the first sampling interval. At the next time step a new optimal control problem based on new measurements of the state is solved over a shifted horizon. Model predictive control has become the accepted standard in the process industry for solving complicated constrained multivariable control problems, see [56]. The success of MPC is largely due to its ability to simply and effectively handle hard constraints on states and control inputs.

We consider two common configurations of HVAC systems. The first configuration is known as dual-duct, single fan [26]. The second configuration is known as single-duct variable air volume (VAV) with reheat. For each HVAC configuration we derive simplified low-order thermal models for the temperature dynamics and energy costs. Our intent is to use the simplest possible physics-based model that can capture the main contributions to dynamics and energy consumption.

Simulation results presented in Section 2.5 show good performance and computational tractability of the resulting scheme. Additionally, the model predictive controller exhibits desirable control behaviors that resemble modern advanced heuristic strategies for HVAC control, reproducing those strategies in a systematic manner.

The resulting model predictive controllers are generally non-convex optimization problems. Nonlinear programming (NLP) solvers based on sequential quadratic programming [52] or interior point methods [64] cannot guarantee global optimality. In this chapter, we study the local optima of the MPC optimization problem for the two aforementioned HVAC configurations. We present physical explanations for each local optimum observed in our results and two detailed investigations into the optimization behavior.

Examples of nonlinear MPC problems exhibiting local optima have been presented in previous literature [62]. The main contribution of this chapter is the detailed investigation into several factors influencing the local optima behavior for simplified models of HVAC systems, and derivation of physics-based rules to classify and rule out a subset of local optima a priori.

For the dual-duct HVAC configuration we observed local optima corresponding to different system operating modes. We investigate analytical behavior of the different operating modes in terms of a subset of the gradient and constraint optimality conditions to determine worst case a-priori conditions under which certain modes can be ruled out as suboptimal.

For the single-duct HVAC configuration we observed local optima corresponding to different transient sequences of control actions. In this case we investigate the effect of system model discretization method on the local optima results of the MPC problem.

2.2 Configuration Descriptions

This section describes the mechanical components and control inputs of the two HVAC system configurations we focus on.

Configuration A

The HVAC system configuration known as dual-duct, single fan is shown in Figure 2.1. Supply air is heated and cooled to desired temperatures in two separate duct systems by a pair of coils (water-to-air heat exchangers). The water flow rate across each coil is controlled by a modulating valve in order to maintain a desired air side outlet temperature setpoint. The two separate duct systems route hot and cold air to mixing boxes at each thermal zone (typically one or several rooms). A mixing box contains a pair of linked dampers (position-controlled louvers), designed so that when the hot side damper closes, the cold side damper opens and vice-versa. We refer to these mixing boxes as the *zone dampers*. The zone dampers serve as a control actuator to provide the desired mixed supply air temperature to a zone. The supply air can be set anywhere in the range between the cooling coil outlet temperature and the heating coil outlet temperature.

In existing simple HVAC control schemes there is no communication between zones, and the central cooling and heating coils are set to constant conservative setpoints. This ensures heating and cooling capacity is always at the system design value, but in the majority of operating conditions below that design capacity, this basic control design wastes a great deal of energy.

Mixed zone air returns to the central air handling unit (AHU) through a return duct. A set of AHU dampers can either exhaust the return air to ambient and use fresh outside air as input to the supply fan, or recirculate the return air, or some combination of the two. Usually the return air will be cooler than outside air temperature on a hot day when cooling is required, or warmer than outside air on a cold day when heating is required. So conventional practice is to recirculate as much air as possible, while maintaining a minimum fraction of fresh air for acceptable indoor air quality. However, the opposite scenario of cooling when return air is warmer than outside air (or heating when return air is cooler than outside air) can occur, and in that case using 100% outside air consumes the least total coil energy. This is known as *economizer operation*.

The described dual-duct single fan configuration is an outdated HVAC design. It was prevalent before the advent of inexpensive and reliable variable frequency drives (VFD) for HVAC fans. Without a VFD, the supply fan runs at a constant design speed at all times, often wasting energy due to excess unused capacity. Because the mixing box dampers are

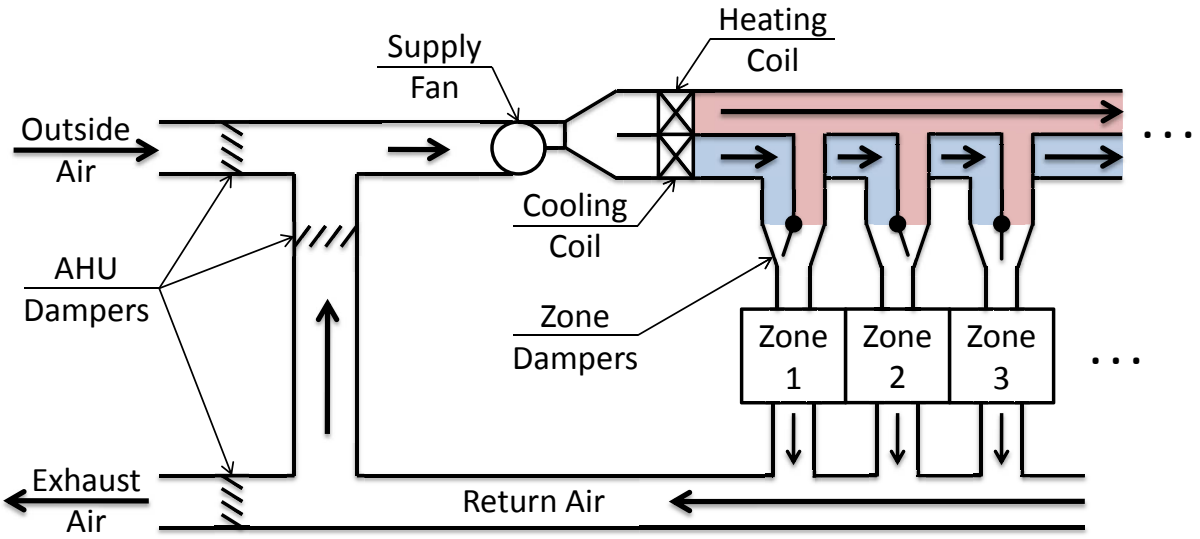


Figure 2.1: Dual-duct, single fan HVAC system schematic, denoted configuration A

linked at each zone, the pressure drop of the entire distribution system to the zones is not highly sensitive to the positions of the zone dampers. Therefore the total flow rate through the supply fan is effectively constant for a given supply fan speed, and the proportions of that flow rate delivered to each zone are also constant. For this reason, this type of system is known as a constant air volume (CAV) system.

We consider systems where the supply fan is equipped with a variable frequency drive (VFD), so the fan speed and therefore the total flow rate can be controlled. We will assume there is a sensor measuring total flow rate, and a lower-level controller for the VFD fan speed to track a desired total flow rate setpoint. Since the mixing box dampers for each zone are linked, the individual zone flow rates are not independently controllable. As a first-order approximation, we treat the flow splits to each zone as constant. This system will be denoted configuration A in the remaining sections.

In summary, the control inputs in this system are: the total supply fan flow rate, the outside air flow rate into the AHU, the cooling coil outlet temperature setpoint, the heating coil outlet temperature setpoint, and the zone mixing box supply temperature setpoints. The states in this system are the zone temperatures.

Configuration B

In this section we introduce an alternative HVAC system configuration, known as single-duct variable air volume (VAV) with reheat. We consider an air handling unit serving multiple

zones, as before. The AHU in this configuration is capable of using either recirculated zone exit air, fresh outside air, or a mix of the two. As shown in Figure 2.2, all of the supply air flows through a cooling coil. The cool air is distributed by a fan to the *VAV boxes* at each zone. A VAV box consists of a damper and a heating coil. We assume there is a supply flow rate sensor in each VAV box, and a lower-level controller for damper position to track a desired zone supply flow rate. The heating coil is used to warm the supply air if that zone requires heating.

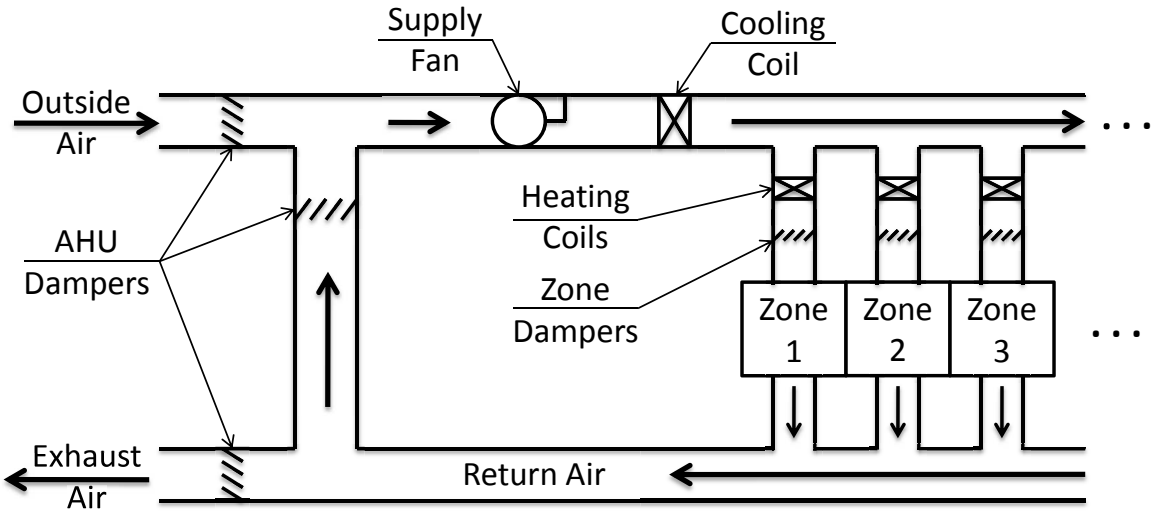


Figure 2.2: Single-duct variable air volume with reheat HVAC system schematic, denoted configuration B

Compared to configuration A, this single-duct system has local control over heating rather than one central heating coil, and the VAV dampers provide individual control over the supply flow splits delivered to each zone. This capability allows outlier zones with unusually high or unusually low thermal loads to be controlled without requiring more flow to be delivered to all other zones. The single-duct configuration also avoids the inefficiency of mixing, and one duct system requires less material and space to install than two so the single-duct configuration is often less expensive. Due to these factors, HVAC systems of the single-duct configuration are more common in modern construction.

In summary, the control inputs in this system are: the flow rates of air supplied to each zone, the outside air flow rate into the AHU, the cooling coil outlet temperature setpoint, and the heating coil outlet temperature setpoints at each zone. The states in this system are the zone temperatures.

2.3 System Modeling

In order to develop control-oriented thermal models of limited order and reduced complexity, we make the following assumptions:

A1 The average lumped temperature dynamics of the thermal zones can be reasonably approximated as first-order. We therefore combine the thermal capacitance of the air, walls, furnishings, and other contents of zone i into a single lumped parameter denoted $(mc)_i$.

A2 Humidity is not explicitly included in our model.

A3 All dynamics except those of the thermal zones are neglected. Actuators are assumed to instantly meet their control setpoints.

A4 A prediction of the thermal loads $\dot{Q}_i$ in each zone due to occupants, equipment, and all heat transfer to or from ambient and other zones is known in advance. Predicted outside air temperature T_{oa} is also known.

Assumption A2 limits the applicability of our model to sensible heat loads rather than latent loads. Latent heat and humidity considerations are important for cooling applications in many climates. Our model is not intended to be high enough fidelity to capture these effects, rather it aims to be reasonably physically representative while remaining simple enough to analyze in closed form.

Assumption A3 neglects all lags in system actuators and lower-level controllers. Those lags could be significant relative to the time constants of the thermal zones in some systems. Our model formulation and MPC methodology can easily be extended to include these lags when necessary, but they are neglected here for simplicity. Similarly, weakening assumption A1 by using a model with multiple states and capacitances per zone would be straightforward.

Assumption A4 is optimistic, because these predictions will never be perfectly accurate. Extensions of this work could account for sensitivity to inaccuracies in predictions by formulating robust or stochastic MPC problems. We consider only the nominal MPC problem in this chapter.

Thermal Zone Model

A first order energy balance gives the following continuous time system dynamics for the temperature T_{zi} of zone i

$$(mc)_i \frac{d}{dt} T_{zi} = \dot{Q}_i + \dot{m}_{zi} c_p (T_{si} - T_{zi}), \tag{2.1}$$

where c_p is the specific heat capacity of air and T_{si} is the temperature of the supply air delivered to zone i. The flow rate $\dot{m}_{zi}$ and supply temperature T_{si} are inputs to this model from the HVAC system.

If we neglect heat transfer by radiation, the thermal loads $\dot{Q}_i$ can be represented as an affine function of the zone temperatures

$$\dot{Q} = \begin{bmatrix} \dot{Q}_1 \\ \vdots \\ \dot{Q}_n \end{bmatrix} = R \begin{bmatrix} T_{z1} \\ \vdots \\ T_{zn} \end{bmatrix} + \dot{Q}_{\text{offset}}, \tag{2.2}$$

where n is the number of thermal zones served by the same AHU, R is a symmetric $n \times n$ matrix of inter-zone heat transfer coefficients and $\dot{Q}_{\text{offset}}$ is a $n \times 1$ vector of constant terms from the predicted thermal load.

A higher-fidelity model of zone thermal dynamics would include multiple states per zone in a higher order RC network [48].

The compact form of (2.1) for all n zones together is

$$M \frac{d}{dt} T_z = R\, T_z + \dot{Q}_{\text{offset}} + c_p \,\text{diag}(\dot{m}_z)(T_s - T_z), \tag{2.3}$$

where $M = \text{diag}((mc)_1, \ldots, (mc)_n)$, $T_z = [T_{z1}, \ldots, T_{zn}]^T$, $\dot{m}_z = [\dot{m}_{z1}, \ldots, \dot{m}_{zn}]^T$, and $T_s = [T_{s1}, \ldots, T_{sn}]^T$.

Let $u_z = [\dot{m}_z^T, T_s^T]^T$, $A(u_z) = M^{-1}(R - c_p \,\text{diag}(\dot{m}_z))$, $B(u_z) = c_p M^{-1} \,\text{diag}(\dot{m}_z) T_s$, and $w = M^{-1}\dot{Q}_{\text{offset}}$, then (2.3) has the following state-affine form

$$\frac{d}{dt} T_z = A(u_z) T_z + B(u_z) + w. \tag{2.4}$$

Assuming $\dot{m}_z$, T_s, and $\dot{Q}_{\text{offset}}$ are piecewise constant (zero-order held) with sample rate Δt, we discretize (2.4) using the trapezoidal method

$$\frac{T_z^+ - T_z}{\Delta t} = A(u_z) \frac{T_z^+ + T_z}{2} + B(u_z) + w, \tag{2.5}$$

where T_z^+ denotes the value of T_z at the next discrete time step $t + \Delta t$. We choose the trapezoidal discretization as a compromise between simplicity and numerical stability. We examine the effect of using different discretization methods in Section 2.6.

This zone thermal model is used for both HVAC configurations, however the mass flow rates $\dot{m}_z$ and temperatures T_s of supply air differ depending on the HVAC configuration.

HVAC System Model

Configuration A

The HVAC control inputs in this configuration are: the total mass flow rate at the supply fan $\dot{m}_s$, the outside air flow rate into the AHU $\dot{m}_{oa}$, the cooling coil setpoint T_c, the heating coil setpoint T_h, and the zone mixing box supply temperature setpoints $T_s = [T_{s1}, \ldots, T_{sn}]^T$. The number of thermal zones served by the same AHU is denoted n.

The portion of the total supply flow delivered to each zone, denoted φ_i for zone i, is constant so that

$$\dot{m}_{zi} = \varphi_i \dot{m}_s, \text{ where } \varphi_i \geq 0 \; \forall \; i \in \{1, \ldots, n\} \text{ and } \sum_{i=1}^{n} \varphi_i = 1. \tag{2.6}$$

In this configuration the individual zone supply flows are derived variables which depend on the supply fan flow rate.

In configuration A, only part of the supply flow passes through the heating and cooling coils. The partial flow rates depend on the zone supply temperatures and flow splits.

$$\dot{m}_h = \sum_{i=1}^{n} \frac{\dot{m}_{zi}(T_{si} - T_c)}{T_h - T_c}, \text{ and } \dot{m}_c = \dot{m}_s - \dot{m}_h, \tag{2.7}$$

where $\dot{m}_h$ and $\dot{m}_c$ are respectively the mass flow rates through the heating and cooling coils. The coil flows do not directly influence the zone temperature dynamics, but will be important for the coil energy consumption in Section 2.3.

Configuration B

The HVAC control inputs in this configuration are: the mass flow rates of air supplied to each zone $\dot{m}_z = [\dot{m}_{z1}, \ldots, \dot{m}_{zn}]^T$, the outside air flow rate into the AHU $\dot{m}_{oa}$, the cooling coil setpoint T_c, and the heating coil setpoints at each zone $T_h = [T_{h1}, \ldots, T_{hn}]^T$. The supply air temperature T_{si} delivered to zone i is equal to the VAV box heating coil setpoint, $T_{si} = T_{hi}$.

All of the supply flow passes through the cooling coil in configuration B, but only the flow to a single zone passes through each heating coil. The coil flow rates are then

$$\dot{m}_{hi} = \dot{m}_{zi}, \text{ and } \dot{m}_c = \dot{m}_s = \sum_{i=1}^{n} \dot{m}_{zi}, \tag{2.8}$$

where $\dot{m}_{hi}$ is the flow rate through the heating coil at zone i, and $\dot{m}_c$ is the flow rate through the cooling coil. The total mass flow rate at the supply fan is again denoted $\dot{m}_s$. For configuration B the total flow rate is a derived variable which depends on the individual zone flows.

Common Components

In both configurations, we assume the ratio of supply flow rate to return flow rate is the same for all zones. Neglecting heat transfer to or from the return duct, the return air temperature T_r is therefore given by a flow-rate-weighted average

$$T_r = \frac{\sum_{i=1}^{n}(\dot{m}_{zi} T_{zi})}{\sum_{i=1}^{n} \dot{m}_{zi}}. \tag{2.9}$$

The AHU mixed air temperature T_m is similarly a flow-weighted average of outside air temperature T_{oa} and return temperature T_r

$$T_m = \frac{\dot{m}_{oa} T_{oa} + (\dot{m}_s - \dot{m}_{oa}) T_r}{\dot{m}_s}. \tag{2.10}$$

In this model the return and mixed air temperatures influence the coil energy consumption, but not the zone dynamics.

Cost Function: Energy Consumption and Price

Energy use of the cooling and heating coils is calculated by integrating over time the air-side thermal power $\dot{m}\, c_p \Delta T$, based on the models in the previous section. We represent the operating characteristics of the cold and hot water circuits with two parameters: efficiency η_h for the hot side, and coefficient of performance η_c for the cold side. The two HVAC configurations A and B have different flow rates and delta temperatures across the coils, so the expressions for the power used by the coils depend on the configuration.

For both configurations cooling coil power P_c has the form

$$P_c = \frac{c_p}{\eta_c} \dot{m}_c (T_m - T_c), \tag{2.11}$$

where $\dot{m}_c$ is given by (2.7) for configuration A and by (2.8) for configuration B.

For configuration A the heating coil power P_h is

$$P_h = \frac{c_p}{\eta_h} \dot{m}_h (T_h - T_m), \tag{2.12}$$

where $\dot{m}_h$ is given by (2.7).

For configuration B the total power of all heating coils is

$$P_h = \sum_{i=1}^{n} \left(\frac{c_p}{\eta_h} \dot{m}_{zi} (T_{si} - T_c) \right). \tag{2.13}$$

A higher-fidelity model would include ancillary equipment such as water pumps and cooling towers, as well as a detailed water-side energy balance. These would be represented by non-constant η_h and η_c as functions of state and input using performance maps, effectiveness curves, etc. In this simplified model we take these parameters to be constants.

In both configurations the electrical power P_f used by the supply fan is

$$P_f = \frac{\dot{m}_s \, \Delta p}{\rho \, \eta_f}, \tag{2.14}$$

where $\dot{m}_s$ is the mass flow rate through the fan, Δp is the pressure difference across the fan, ρ is the air density, and η_f is the efficiency of the fan. Assuming incompressible flow gives

Δp proportional to $\dot{m}_s^2$, where the ratio of proportionality depends on the flow resistance of all the downstream zone dampers.

In configuration A we assume the flow resistance of the mixing boxes is constant, so we can take $\Delta p = \rho\,\eta_f \kappa_A \dot{m}_s^2$ where the constant parameter κ_A captures the fan efficiency and duct pressure losses. Equation (2.14) then becomes

$$P_f = \kappa_A \dot{m}_s^3. \tag{2.15}$$

In configuration B however, the flow resistance of the VAV dampers depends on their positions. At higher flow rates with the dampers more open, the overall flow resistance is lower. So the increase of pressure drop with flow rate will be slower than quadratic, and fan power increases slower than cubic. For simplicity we restrict our model to polynomial form, so we take $\Delta p = \rho\,\eta_f \kappa_B \dot{m}_s$. With this form of simplification for configuration B, we have

$$P_f = \kappa_B \dot{m}_s^2. \tag{2.16}$$

A higher-fidelity model would include detailed pressure drop characteristics of the ducts and dampers as functions of flow rate, and a representation of fan speed control incorporating fan performance curves. We have abstracted all notions of the supply pressure control loop into our simplified model (2.16), with an implicit assumption that the supply fan is operated at the minimum power level necessary to deliver the desired total flow rate to all the zones. Fig. 2.3 shows that a quadratic fit of recorded fan power data from a real VAV system under supply pressure control is reasonably accurate.

We introduce several parameters to reflect utility pricing. The cost in dollars per unit energy content is denoted r_e for electricity, r_h for heating fuel (typically gas, or steam from a central plant). These costs may vary in time, especially for electricity, to reflect time-of-use or dynamic utility pricing. We assume time variation of utility rates occurs in a zero-order hold manner at sample rate Δt.

We also incorporate a feature of some utility structures wherein peak electric power use is penalized. Some utilities only implement this peak-use charge during certain hours of the day, so we express this feature by defining a windowing function $\psi(t)$. The value of $\psi(t)$ equals the given cost per unit peak power during restricted time intervals, and zero elsewhere.

The total utility cost from time t to time $t + N\Delta t$, where N is the prediction horizon length in number of steps, is

$$J = \int_t^{t+N\Delta t} (r_e P_f + r_e P_c + r_h P_h) d\tau + \max_{\tau \in [t, t+N\Delta t]} (\psi(\tau)(P_f + P_c)). \tag{2.17}$$

Constraints

The system states and control inputs are subject to constraints due to control requirements and actuator limits. Since the control inputs differ for the two HVAC configurations, some of these constraints only apply to one configuration.

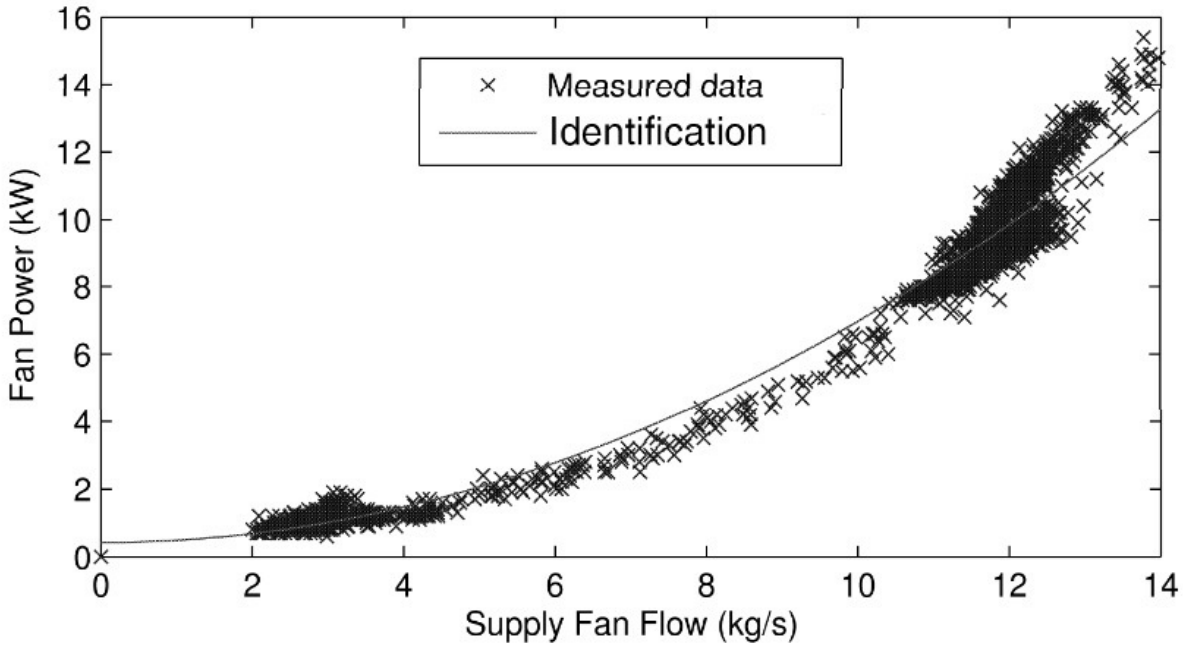


Figure 2.3: Recorded data from an AHU supply fan in the Bancroft Library at University of California, Berkeley

In our models for both configurations, the coil capacity constraints are particularly simplified. A higher-fidelity model would constrain the coil capacities more accurately using detailed energy balances and performance maps.

Configuration A

- $T_h \geq T_m$, heating coil can only increase temperature. (2.18a)
- $T_h \leq \overline{T}_h$, heating coil setpoint must be less than hot water temperature. (2.18b)
- $T_{si} \leq T_h \; \forall \; i \in \{1, \ldots, n\}$, supply temperature must be less than heating coil setpoint. (2.18c)

Configuration B

- $\underline{\dot{m}}_{zi} \leq \dot{m}_{zi} \leq \overline{\dot{m}}_{zi} \; \forall \; i \in \{1, \ldots, n\}$, zone supply flow must be between minimum ventilation requirement and maximum VAV box capacity. (2.19a)
- $T_{si} \leq \overline{T}_h \; \forall \; i \in \{1, \ldots, n\}$, heating coil setpoint must be less than hot water temperature. (2.19b)

Common Constraints

- $\underline{\dot{m}}_s \leq \dot{m}_s \leq \overline{\dot{m}}_s$, total supply flow must be between min overall ventilation requirement and max fan capacity. (2.20a)
- $\underline{\dot{m}}_{oa} \leq \dot{m}_{oa} \leq \dot{m}_s$, outside air flow must be above a minimum value for indoor air quality, and cannot be greater than total supply flow. (2.20b)
- $T_c \leq T_m$, cooling coil can only decrease temperature. (2.20c)
- $T_c \geq \underline{T}_c$, cooling coil setpoint must be greater than cold water temperature. (2.20d)
- $T_{si} \geq T_c \ \forall \ i \in \{1, \ldots, n\}$, supply temperature must be greater than cooling coil setpoint. (2.20e)
- $\underline{T}_{zi} \leq T_{zi} \leq \overline{T}_{zi} \ \forall \ i \in \{1, \ldots, n\}$, zone temperature must be within comfort range. (2.20f)

In the above constraints, limit values denoted by $\underline{\star}$ and $\overline{\star}$ are treated as system parameters, assumed to be known in advance.

Model Summary

Combining the HVAC system model from Section 2.3 and the discretized thermal zone model from Section 2.3, the consolidated model can be expressed as

$$f(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) = 0 \ \forall \ k \in \{0, \ldots, N-1\}. \tag{2.21a}$$

where $x_{k|t}$ is the value of the state vector (zone temperatures T_z) at time $t + k\Delta t$ predicted at time t, $u_{k|t}$ is the value of all control inputs at time $t + k\Delta t$ predicted at time t, and $w_{k|t}$ is the value of the disturbance inputs $\dot{Q}_{\text{offset}}$ and T_{oa} at time $t + k\Delta t$ predicted at time t.

The constraints from Section 2.3 can be expressed as

$$g(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) \leq 0 \ \forall \ k \in \{0, \ldots, N-1\}. \tag{2.21b}$$

Define the continuous-time one-step cost as

$$J^c_{k|t} = \int_{t+k\Delta t}^{t+(k+1)\Delta t} (r_e P_f + r_e P_c + r_h P_h) d\tau. \tag{2.21c}$$

The integral (2.21c) is approximated according to the trapezoidal discretization for consistency with the discretization of the state dynamics (2.5). Let $J_{k|t}$ be the discretization of (2.21c). The peak power charge is discretized by taking the maximum over the discretely sampled time instants rather than a continuous time interval.

The system model f, control inputs u, cost J, and constraints g are different for the two HVAC configurations.

For configuration A, the model f applies the zone supply temperatures and flows, with flow splits from (2.6), to the thermal zone model (2.5). The control inputs are $u = [\dot{m}_s, \dot{m}_{oa}, T_c, T_h, T_s^T]^T$. The cost function J combines (2.7), (2.9)-(2.12), and (2.15). The constraint function g combines (2.18a)-(2.18c), (2.20a)-(2.20f), (2.9), and (2.10).

For configuration B, the model f applies the zone supply temperatures (with $T_s = T_h$) and flows to the thermal zone model (2.5). The control inputs are $u = [\dot{m}_z^T, \dot{m}_{oa}, T_c, T_h^T]^T$. The cost function J combines (2.8)-(2.11), (2.13), and (2.16). The constraint function g combines (2.19a)-(2.19b), (2.20a)-(2.20f), (2.9), and (2.10).

2.4 Optimal Control Design

Model predictive control solves at each time step t the following optimization problem

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{X}} \sum_{k=0}^{N-1} J_{k|t} + \max_{k\in\{0,\dots,N\}} (\psi(t+k\Delta t)(P_f + P_c)) \\ &\text{subj. to, } \forall\, k \in \{0,\dots,N-1\}, \\ &f(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) = 0 \\ &g(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) \le 0 \\ &x_{0|t} = T_z(t) \end{aligned} \tag{2.22}$$

where $\mathbf{U} = \{u_{0|t}, \dots, u_{N-1|t}\}$ is the set of predicted control inputs at time t, $\mathbf{X} = \{x_{1|t}, \dots, x_{N|t}\}$ is the set of predicted system states at time t, starting from initial state $x_{0|t} = T_z(t)$ and applying the input sequence $\mathbf{U}$ to the system model (2.21a).

The peak power charge $\max_{k\in\{0,\dots,N\}}(\psi(t+k\Delta t)(P_f + P_c))$ can be reformulated by introducing an epigraph variable ξ and expressing the maximization as additional inequality constraints.

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{X},\xi} \sum_{k=0}^{N-1} J_{k|t} + \xi \\ &\text{subj. to, } \forall\, k \in \{0,\dots,N-1\}, \\ &f(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) = 0 \\ &g(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) \le 0 \\ &\psi(t+k\Delta t)(P_f + P_c) \le \xi \\ &\psi(t+N\Delta t)(P_f + P_c) \le \xi \\ &x_{0|t} = T_z(t) \end{aligned} \tag{2.23}$$

Let the optimal control input solution of problem (2.23) at time t be denoted by $\mathbf{U}^\star = \{u^\star_{0|t}, \dots, u^\star_{N-1|t}\}$. Then, the first step of $\mathbf{U}^\star$ is input to the system, $u(t) = u^\star_{0|t}$. The optimization (2.23) is repeated at time $t+\Delta t$, with the updated new state $x_{0|t+\Delta t} = T_z(t+\Delta t)$ yielding a *moving or receding horizon control* strategy.

The optimization problem (2.23) has nonlinear cost and nonlinear constraints. In order to solve this optimization problem we use the interior-point NLP solver Ipopt [64] via the Yalmip toolbox [43].

2.5 Simulation Results

We present simulation results for configuration B in the following cases:

1. Nominal case, r_e uses "low" value from Table 2.1 and $\psi(t) = 0$ at all times.
2. Modified electric rate schedule: r_e uses "high" value between 12 noon and 4:30 PM and the "low" value at all other times. No peak power charge, $\psi(t) = 0$.
3. Peak power penalty: $\psi(t) = 1$ \$/kW and r_e uses "low" rate at all times.

Table 2.1: Parameter values used

Parameter	Value	Units
n	5	zones
N	48	steps
Δt	1800	s
c_p	1	kJ/(kg·K)
$(mc)_i$	1000	kJ/K
R	0	kW/K
η_h	0.9	dimensionless
η_c	4	dimensionless
κ_f	0.065	kW·s^2/kg^2
r_e	$\begin{cases} \text{"high"} = 1.5 \cdot 10^{-4} \\ \text{"low"} = 3 \cdot 10^{-5} \end{cases}$	\$/kJ
r_h	$5 \cdot 10^{-6}$	\$/kJ
T_0	18	°C
$\underline{T}_{zi}$	$\begin{cases} \text{6:30 AM to 6:30 PM} = 21 \\ \text{7 PM to 6 AM} = 12 \end{cases}$	°C
$\overline{T}_{zi}$	$\begin{cases} \text{6:30 AM to 6:30 PM} = 24 \\ \text{7 PM to 6 AM} = 32 \end{cases}$	°C
$\underline{\dot{m}}_{zi}$	$\begin{cases} \text{6:30 AM to 6:30 PM} = 0.025 \\ \text{7 PM to 6 AM} = 0 \end{cases}$	kg/s
$\overline{\dot{m}}_{zi}$	1.5	kg/s
$\underline{T}_c$	5	°C
$\overline{T}_h$	40	°C
$\overline{d_r}$	0.9	dimensionless

For ambient temperature T_{oa}, we use a sinusoid with period 1 day, minimum value 10 °C at time 1:30 AM, and maximum value 30 °C at 1:30 PM. Zone thermal loads $\dot{Q}_i$ are set to the time-varying profiles shown in Fig. 2.4. The results of case 1 are shown in Fig. 2.5 and 2.6, case 2 is shown in Fig. 2.7, and case 3 is shown in Fig. 2.8.

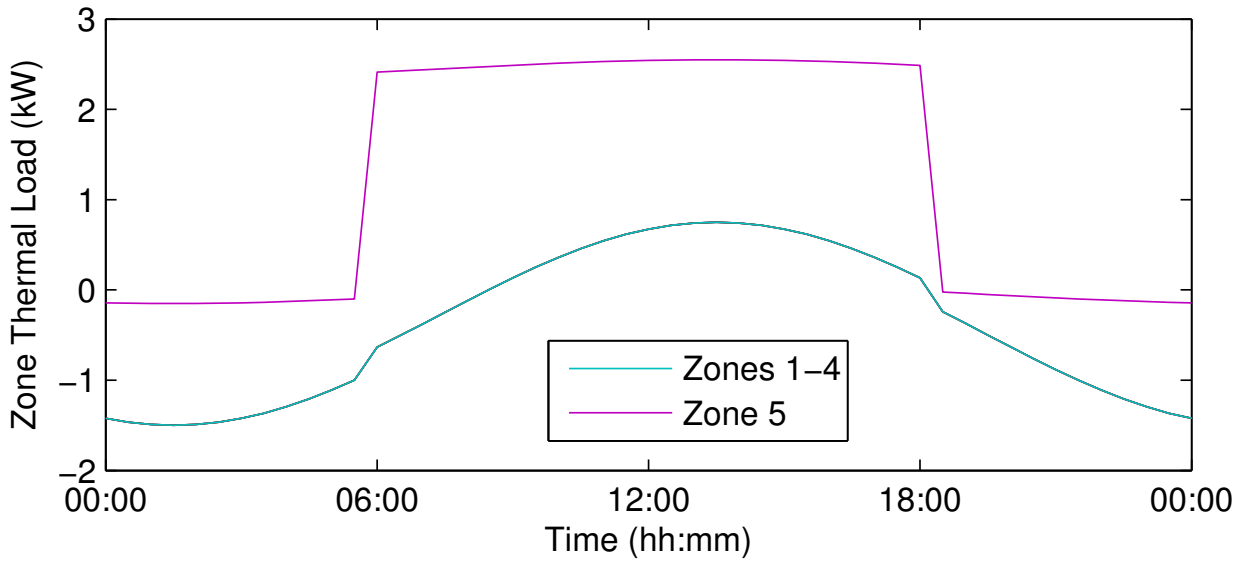


Figure 2.4: Zone thermal loads $\dot{Q}_i$

We observe several interesting behaviors for the nominal case 1 in Fig. 2.5 and 2.6. Before the occupied period begins at 6:30 AM, zone 5 is set to zero flow and zones 1-4 have a small flow rate at the maximum heating temperature to counteract the cooling loads (negative $\dot{Q}_i$). All zones preheat to satisfy the tighter occupied temperature constraints before the occupied hours begin. From 6:30 AM until 9 AM, zone 5 is in cooling mode but the other zones are in heating. This period is an economizer mode condition: a mix of outside air maintains the mixed temperature close to $\underline{T}_{zi}$, while zone 5 satisfies its cooling demand with a very large flow rate. After 9 AM the ambient temperature is warmer than the return temperature so the AHU dampers return to maximum recirculation. The cooling coil activates at this time, reaching its lowest setpoint by 10AM. Zones 1-4 begin transitioning to cooling mode here. Immediately before the end of the occupied period at 6:30 PM, we see a cooling coil supply temperature reset behavior. The load prediction is much lower after 6:30 PM, so the cooling coil begins increasing its setpoint early, trading lower cooling power for higher fan power (the flow to zone 5 must increase to keep it cooled using warmer supply air). The erratic one-at-a-time heating of zones 1-4 after 10:30 PM appears to be a consequence of the return temperature dependence on mass flows. When only one zone is heated with a large mass flow (others at low flow), the return temperature is influenced most by the high-flow zone. Increased return temperature reduces the required heating coil energy for the next zones to be heated.

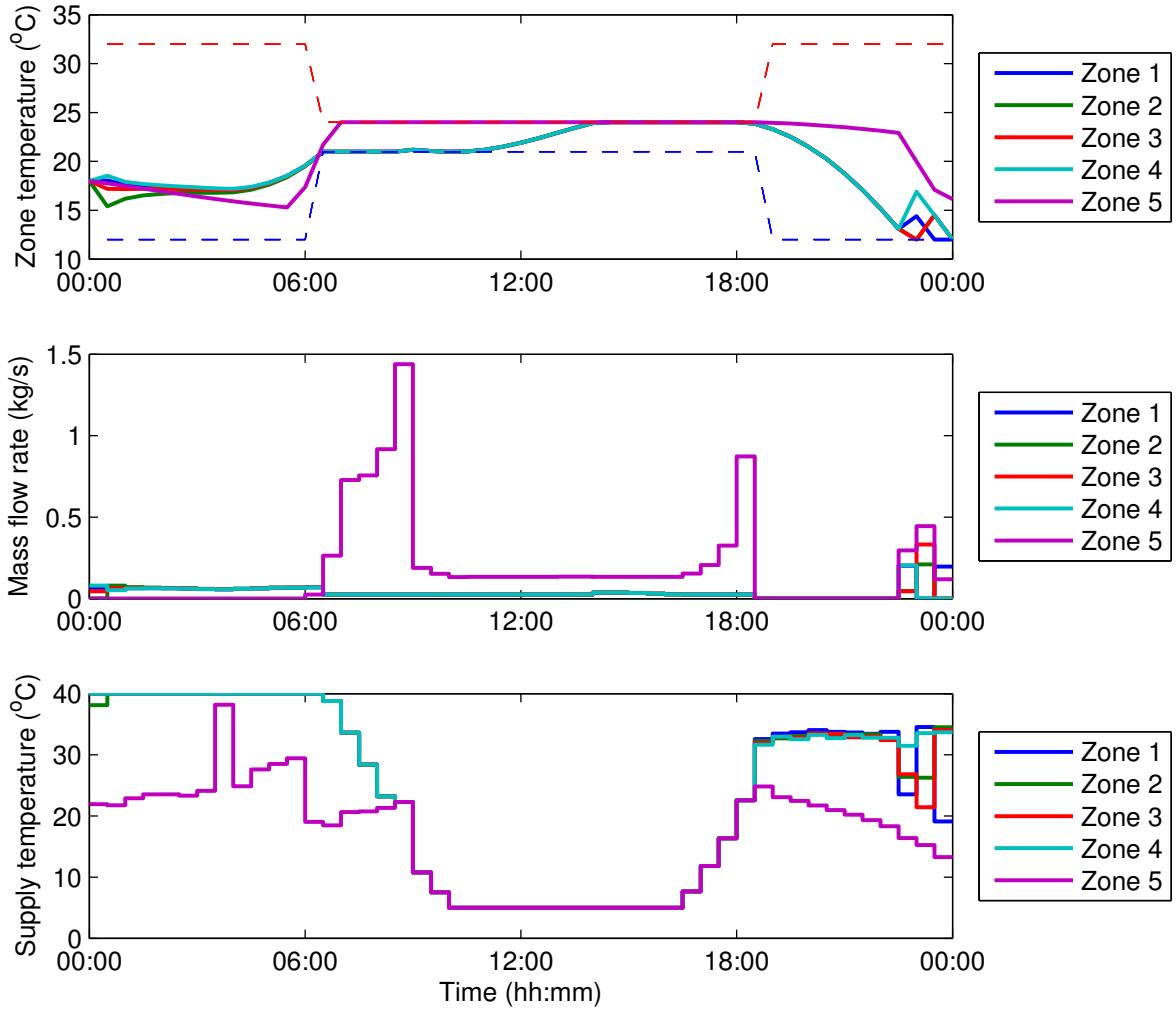


Figure 2.5: Case 1 zone results. Shown dashed in the first plot are $\underline{T}_{zi}$ and $\overline{T}_{zi}$

Both case 2 in Fig. 2.7 and case 3 in Fig. 2.8 demonstrate precooling of zone 5 and lengthened cooling of zones 1-4, but with different timing and intent. In case 2, $\psi(t) = 0$ but the electric rate r_e has a higher value between 12 noon and 4:30 PM. So precooling is only performed immediately before noon, with a corresponding spike in cooling power, so that less cooling energy is used between 12 noon and 4:30 PM. In case 3, $\psi(t) = 1$ so load-shifting is used to minimize peak power. Zone 5 is precooled beginning earlier in the morning, increasing cooling power at a time when it would otherwise be low and shifting electric power use away from the times it would normally be highest.

Our computational results show that in response to either time-varying electric rates (r_e, case 2) or peak power penalties ($\psi(t)$, case 3), this optimization-based controller does not use appreciably more total energy. We are not showing the combination of r_e high and $\psi(t) = 1$

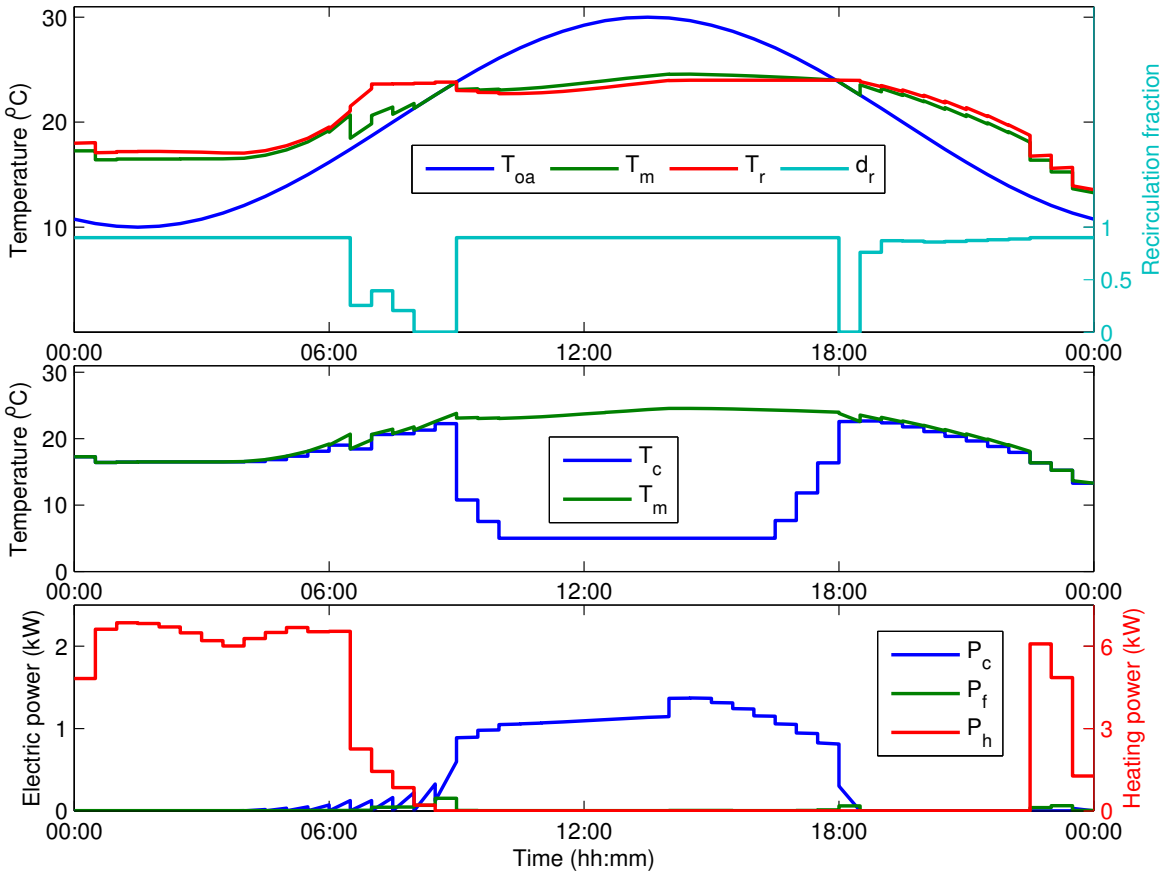


Figure 2.6: Case 1 AHU results

here, but the results are very similar to case 3. Imposing a penalty on peak power rules out the type of short-duration precooling seen in case 2.

This MPC algorithm has shown encouraging results in a few interesting cases for a sample problem. The control performance, entirely from an optimization origin, exhibits aspects of heuristic HVAC control such as economizer control, supply temperature reset, demand response, precooling, and load-shifting in a coordinated manner. The computational time for each of the above cases was less than one minute, faster than the time scales of a HVAC system. Additional work is necessary in the areas of system identification, model validation, and thermal load prediction. Extensions could include robust or stochastic MPC for this system, wherein we account for uncertainty in future thermal load values and the effects of model mismatch.

Experimental deployments of these algorithms were performed in [7], [6], and [1]. Closed loop performance on physical buildings confirmed the advanced control behavior and constraint satisfaction observed in simulations, and demonstrated 20 to 60 percent energy savings relative to standard practice digital HVAC control.

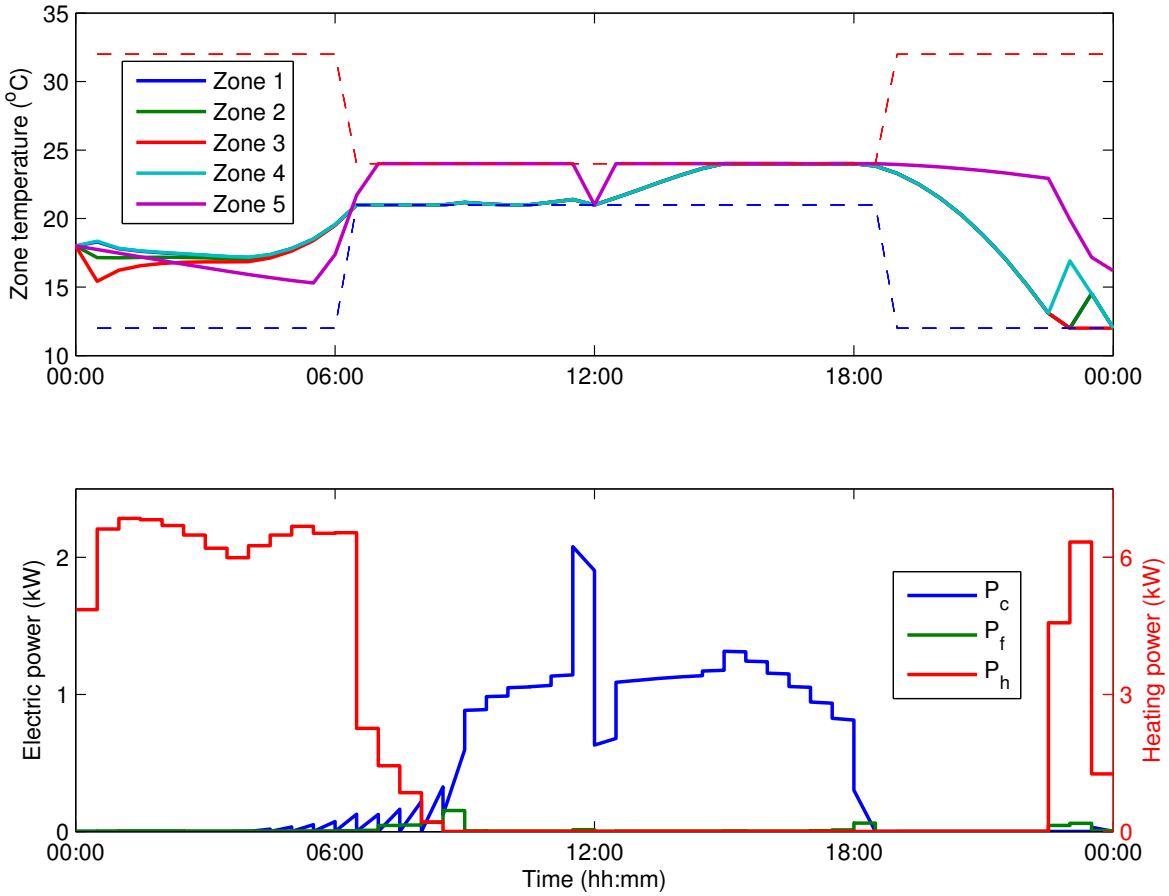


Figure 2.7: Case 2 overview. Note the precooling and spike in cooling power immediately before noon

2.6 Local Optima Analysis

Ipopt and other fast nonlinear programming codes are not global solvers for non-convex problems, so these algorithms can converge to a local optimizer. We must provide guess values of the optimization variables to initialize the first iteration of a NLP solver. Denote these guess values by $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{X}}$ for the input and state trajectories respectively. If the guess trajectories $(\widehat{\mathbf{U}}, \widehat{\mathbf{X}})$ are feasible according to the constraints (2.21a) and (2.21b), then the cost value at the corresponding local optimizer $(\mathbf{U}^\star, \mathbf{X}^\star)$ must be less than or equal to the cost value at the starting guess. In this work the term *region of attraction* of a local optimizer $(\mathbf{U}^\star, \mathbf{X}^\star)$ refers to the set of initial guesses $(\widehat{\mathbf{U}}, \widehat{\mathbf{X}})$ that converge to the same local optimizer $(\mathbf{U}^\star, \mathbf{X}^\star)$ when using a given NLP algorithm.

There are several methods commonly used in practice to select an initial guess $(\widehat{\mathbf{U}}, \widehat{\mathbf{X}})$ for nonlinear MPC. Using a shifted version of the MPC solution from the previous time step is known as warm start and often results in faster NLP convergence. Or if a representation of a

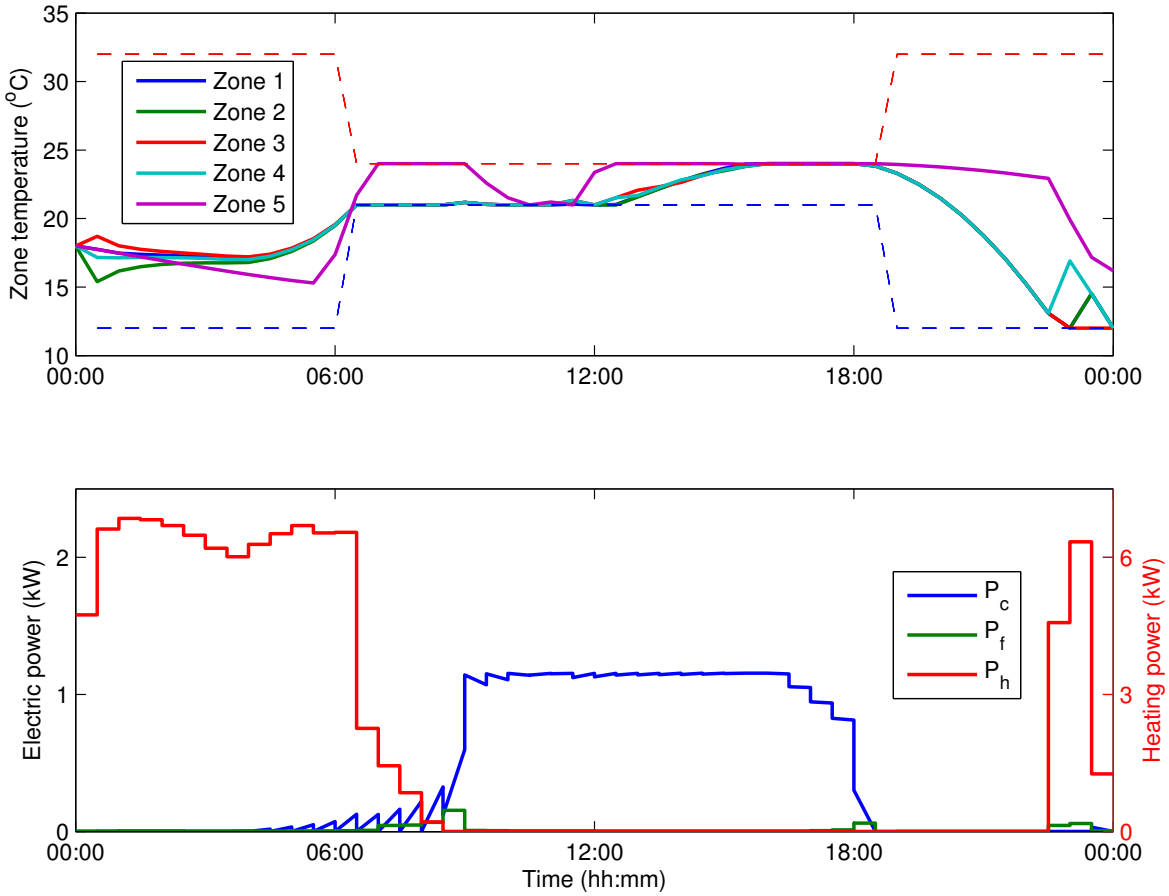


Figure 2.8: Case 3 overview. Note the timing of the precooling and the intentional plateau in cooling power

baseline current-practice controller for the same system is available, then we can generate a guess $(\widehat{\mathbf{U}}, \widehat{\mathbf{X}})$ by simulating the baseline controller in closed-loop with the simplified system model (2.21a). This strategy has the desirable property that any local optimum MPC solution will outperform a baseline control strategy according to open-loop prediction, assuming the baseline controller does not violate any constraints.

For a non-convex optimization problem, if there are multiple local optima then the local optimizer $(\mathbf{U}^\star, \mathbf{X}^\star)$ and the local optimum cost value obtained by a NLP algorithm depend on the initial guess $(\widehat{\mathbf{U}}, \widehat{\mathbf{X}})$. Regardless of the method used to select it, there is no guarantee that the initial guess is in the region of attraction of the global optimum solution.

In this section we first show simulation results exhibiting local optima for HVAC configuration A and present physical interpretations of the observed local optima. Informed by these example results, we then derive analytical physics-based rules for configuration A that can be used to rule out local optima such as those observed in our example results. Then

we show simulation results exhibiting local optima for HVAC configuration B and present physical interpretations. Lastly we perform more detailed simulations for configuration B comparing different system model discretization methods.

Configuration A

Our simulations revealed the presence of local optima for configuration A in a small instance of the optimization problem (2.22) with three zones and a prediction horizon of 2 steps. System parameter values for this example are given in Table 2.2 and Table 2.3. The initial state condition and all other system model parameters are held constant for all simulations.

The local optima are found by repeated execution of a NLP algorithm. Each execution of the NLP algorithm is started from a randomly selected initial guess, uniformly distributed within the allowable range of states and control inputs. This method is not guaranteed to find the global optimum or identify every possible local optimum, however over many samples the probability of identifying local optima having large regions of attraction should increase.

Table 2.2: Parameters used for both configurations

Description	Symbol	Value	Units
Number of zones	n	3	-
Prediction horizon length	N	2	steps
Discrete sample time (step length)	Δt	1800	s
Air specific heat capacity	c_p	1	kJ / (kg·K)
Heat transfer matrix	R	0	kW / K
Outside air temperature	T_{oa}	16	C
Cold water coefficient of performance	η_c	4	$\mathrm{kW_{thermal}/kW_{electric}}$
Hot water efficiency	η_h	0.9	$\mathrm{kW_{thermal}/kW_{fuel}}$
Electric utility rate	r_e	0.108	$\mathrm{\$/kWh_{electric}}$
Heating fuel utility rate	r_h	0.018	$\mathrm{\$/kWh_{fuel}}$
Peak power penalty	$\phi(t)$	0	$\mathrm{\$/kW_{electric}}$
Max fan capacity	$\overline{\dot{m}}_s$	6	kg / s
Min total ventilation	$\underline{\dot{m}}_s$	0.015	kg / s
Min fresh air	$\underline{\dot{m}}_{oa}$	0.015	kg / s
Heating coil max temperature	$\overline{T}_h$	40	C
Cooling coil min temperature	$\underline{T}_c$	5	C

We found 6 distinct families of local optima in this example. At the first time step the set of locally optimal solutions exhibit two different control modes: a *heating mode* and a *cooling mode*. At the second time step, the local optima exhibit three different control modes: a heating mode, a cooling mode, and a third *intermediate mode*. Every combination of modes

Table 2.3: Configuration A parameters

Description	Symbol	Value	Units
Zone thermal capacitance	$(mc)_i$	100	kJ / K
Fixed thermal loads	$\dot{Q}_{\text{offset}}$	$[3, 4.5, 6]^T$	kW
Zone flow splits	φ_i	0.333	-
Fan power coefficient	κ_A	0.08	kW/(kg/s)^3
Upper comfort bound	$\overline{T}_{zi}$	24	C
Lower comfort bound	$\underline{T}_{zi}$	21	C
Initial state conditions	$T_{zi}(0)$	22	C

for the first and second time step was feasible, so over the horizon of 2 steps we have six families of local optima. Different local optima belonging to the same family have different heating or cooling coil setpoints. The cost value, state trajectories, and all other control inputs are *equal* within a family. The coil setpoints and zone supply temperatures of each family are illustrated in Figure 2.9 and full results are given in Table 2.4. Coil temperature setpoints denoted by "Fig.2.9" in Table 2.4 correspond to the x-axis ranges highlighted in Figure 2.9.

Table 2.4: Result details, configuration A

Family #	t	$\dot{m}_s$ (kg/s)	$\frac{\dot{m}_{oa}}{\dot{m}_s}$	T_c (C)	T_h (C)	$[T_{z1}^+, T_{z2}^+, T_{z3}^+]$ (C)	$J^\star$ ($)
1	0	1.667	1	Fig.2.9	16	[21, 24, 24]	0.095
	Δt	1.688	1	Fig.2.9	16	[21.56, 24, 24]	
2	0	1.667	1	Fig.2.9	16	[21, 24, 24]	0.113
	Δt	2.25	1	16	Fig.2.9	[21, 21, 24]	
3	0	2.524	1	16	Fig.2.9	[21, 21, 24]	0.133
	Δt	1.8	1	Fig.2.9	16	[21, 24, 24]	
4	0	2.524	1	16	Fig.2.9	[21, 21, 24]	0.143
	Δt	2.25	1	16	Fig.2.9	[21, 22.74, 24]	
5	0	1.667	1	Fig.2.9	16	[21, 22.5, 24]	0.177
	Δt	3	0.69	18	18	[21, 22.5, 24]	
6	0	2.524	1	16	Fig.2.9	[21, 22.5, 24]	0.21
	Δt	3	0.69	18	18	[21, 22.5, 24]	

In this example the zones have positive thermal loads $\dot{Q}_i$ so the supply temperatures must be lower than the zone temperatures in order to counteract the loads and remain within the comfort range. Because the zones have different thermal load values but the flow rates to different zones have fixed ratios in configuration A, the required supply temperatures will in

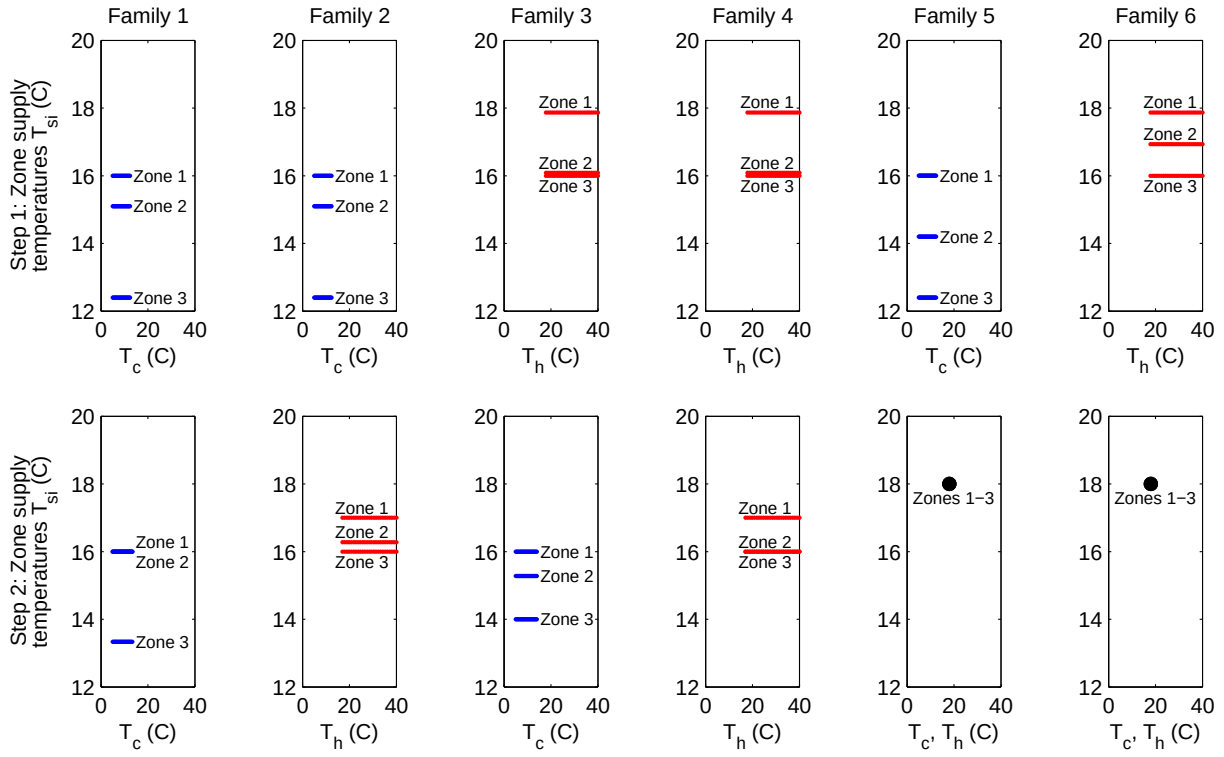


Figure 2.9: Families of local optima for HVAC configuration A

general be different for each zone. In order for the zone dampers to be capable of meeting the required supply temperatures, as a consequence of constraints (2.18b) and (2.20e) the heating coil setpoint must be warmer than the highest required supply temperature and the cooling coil setpoint must be cooler than the lowest required supply temperature.

The outside air temperature here is $T_{oa} = 16$ C, which is cooler than the zone temperatures so this is an economizer condition. Free cooling is available by using outside air instead of recirculated return air at the AHU, reducing coil energy. The difference between the cooling mode and the heating mode is whether the outside air is cooled or heated before being supplied to the zones. The zones must be cooled in all feasible modes for this example, so the names of the modes here refer only to which coil is consuming energy in the AHU.

In the cooling mode, the cooling coil setpoint T_c is strictly less than 16 C and the heating coil setpoint T_h is equal to 16 C (no heating of the outside air). In the heating mode, the heating coil setpoint T_h is strictly greater than 16 C and the cooling coil setpoint T_c is equal to 16 C (no cooling of the outside air). The intermediate mode is a special case where a single supply temperature can meet all of the different thermal loads. The single supply temperature for these parameters is 18 C which is warmer than outside air temperature, so this mode can be realized by mixing outside air with recirculated air. When AHU mixed

temperature T_m equals 18 C, the cooling and heating coil setpoints can both be set to 18 C requiring no coil energy at all. This mode requires zones with different loads to be at different temperatures, so it is not possible at the first time step when all of the zones have the same initial condition temperature.

Note that in cooling mode, the cooling coil setpoint T_c can have any value between 5 and 16 C as long as it is less than or equal to the coolest zone supply temperature. Likewise in heating mode, the heating coil setpoint T_h can have any value between 16 and 40 C as long as it is greater than or equal to the warmest zone supply temperature. When T_c is cooler or T_h warmer than necessary, the zone dampers compensate to maintain the same mixed supply temperatures T_s. A lower cooling coil setpoint or a higher heating coil setpoint requires less flow through the respective coil to produce the same zone supply temperatures.

Cooling mode is feasible for low supply flow rates, where all required zone supply temperatures are below 16 C (otherwise the heating coil setpoint would need to be warmer than outside air temperature). Heating mode is feasible at higher flow rates, where all required supply temperatures are above 16 C but cooler than the zone temperatures. The heating mode will require a higher flow rate to counteract the same thermal load, since the difference between supply and zone temperatures will be smaller than for the cooling mode.

The intermediate mode is feasible at the second time step. A supply temperature of 18 C and a high supply fan flow rate of 3 kg/s are capable of maintaining steady-state conditions within the comfort bounds in all zones. At that steady state zone temperature condition, the return temperature is 22.5 C so a mix of 31% recirculated flow and 69% outside air produces the necessary supply temperature without using either the cooling or heating coils at all. At the first time step, either cooling mode or heating mode can reach the required state values for this third mode from the initial conditions.

Table 2.4 confirms that the modes requiring lower flow rates have lower overall cost (accounting for coil energy as well). Cooling mode requires less flow than heating mode, and both require less flow than the intermediate mode. Therefore, the lowest-cost family of local optima is in cooling mode at both time steps. Due to the initial conditions, heating mode at the second time step requires less flow than heating mode at the first time step to remain inside the comfort bounds. So the second-lowest-cost family of local optima is in cooling mode at the first step, then heating mode at the second step. The third-lowest-cost family is the reverse: heating mode then cooling mode. The fourth-lowest cost family is in heating mode at both steps. The fifth-lowest cost family is in cooling mode then the intermediate mode. The highest-cost family of local optima is in heating mode then the intermediate mode.

The apparent global optimum family 1 corresponds to the intuitive control strategy of cooling the supply air to achieve cooling of the zones. The heating and intermediate modes are physically feasible according to the system model, but would not be used by a conventional controller in this operating condition. We would like a way of preventing MPC from selecting an unintuitive control strategy when such a strategy is clearly suboptimal. In the next section we use the simplified model to derive rules and conditions to determine when this is possible.

Physics-based Rules

In this section, we use the simplified model for configuration A to derive rules that can exclude local optima under specific conditions. These rules will apply to all instances of this system model, at every time step, regardless of the number of zones or the length of the prediction horizon.

First we examine the unconstrained partial derivatives of the coil powers (2.11) and (2.12) with respect to coil temperature setpoints. Substituting (2.7) into the coil powers (2.11) and (2.12) gives

$$r_e P_c + r_h P_h = \frac{r_e c_p}{\eta_c}(T_m - T_c)\left(\sum_{i=1}^{n} \frac{\dot{m}_{zi}(T_h - T_{si})}{T_h - T_c}\right) + \frac{r_h c_p}{\eta_h}(T_h - T_m)\left(\sum_{i=1}^{n} \frac{\dot{m}_{zi}(T_{si} - T_c)}{T_h - T_c}\right). \tag{2.24}$$

The partial derivatives of (2.24) with respect to coil temperature setpoints T_c and T_h are

$$\frac{\partial}{\partial T_c}(r_e P_c + r_h P_h) = -c_p\left(\frac{r_e}{\eta_c} + \frac{r_h}{\eta_h}\right)\frac{T_h - T_m}{(T_h - T_c)^2}\left(\sum_{i=1}^{n} \dot{m}_{zi}(T_h - T_{si})\right)$$

$$\text{and} \quad \frac{\partial}{\partial T_h}(r_e P_c + r_h P_h) = c_p\left(\frac{r_e}{\eta_c} + \frac{r_h}{\eta_h}\right)\frac{T_m - T_c}{(T_h - T_c)^2}\left(\sum_{i=1}^{n} \dot{m}_{zi}(T_{si} - T_c)\right). \tag{2.25}$$

The partial derivative with respect to T_c is nonpositive so if $T_h \neq T_m$ then T_c should be as large as possible to minimize coil power. The partial derivative with respect to T_h is nonnegative so if $T_m \neq T_c$ then T_h should be as small as possible to minimize coil power.

Minimizing T_h subject to (2.18a), (2.18c) and maximizing T_c subject to (2.20c), (2.20e), the control variables at each time step for any local optimum must fall under one of 4 scenarios. The scenarios correspond to different possible orderings of mixed temperature T_m, minimum supply temperature $T_{s,min} = \min_i T_{si}$, and maximum zone supply temperature $T_{s,max} = \max_i T_{si}$, and are defined by the following inequality functions.

(S1) $T_c = T_{s,min} \leq T_m$ and $T_h = T_{s,max} \geq T_m$

(S2) $T_c \leq T_{s,min} \leq T_m$ and $T_h = T_m \geq T_{s,max}$

(S3) $T_c = T_m \leq T_{s,min}$ and $T_h \geq T_{s,max} \geq T_m$

(S4) $T_c = T_m \leq T_{s,min}$ and $T_h = T_m \geq T_{s,max}$, so $T_{si} = T_m$ for all zones i

Note that S2 has an additional degree of freedom on T_c and S3 has an additional degree of freedom on T_h, due to cancellations in (2.24). In S2 when $T_h = T_m$, the coil cost is independent of T_c. In S3 when $T_c = T_m$, the coil cost is independent of T_h.

These scenarios are not mutually exclusive. In fact, S4 = S1 $\cap$ S2 $\cap$ S3. The boundaries and intersection sets of these 4 scenarios are shown with a Venn diagram in Figure 2.10.

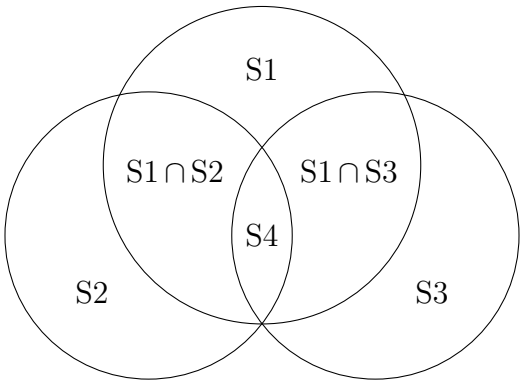


Figure 2.10: Venn diagram of potentially optimal scenarios

In scenario S1, either the cooling coil or the heating coil can be actively using energy, or both, or neither. In scenario S2, the cooling coil can be actively using energy but the heating coil does not. In scenario S3, the heating coil can be actively using energy but the cooling coil does not. In scenario S4, neither coil is actively using energy. By the terminology of the previous section, S2 corresponds to the cooling mode, S3 corresponds to the heating mode, and S4 corresponds to the intermediate mode. When a coil is inactive, its setpoint is equal to the mixed air temperature so it uses no energy. There may still be air flow through an inactive coil, according to equation (2.7).

The partial derivative and constraint conditions used to derive these scenarios are a subset of the optimality conditions of problem (2.22). At each time step, the control variables of any locally optimum solution to problem (2.22) must be contained in one of the scenarios. Hence we can restrict the search space to the union $\mathrm{S1} \cup \mathrm{S2} \cup \mathrm{S3} \cup \mathrm{S4}$. Our goal is to further restrict the search space for the global optimum by identifying conditions when one or more of the scenarios can be ruled out.

We propose to do so by calculating a-priori lower and upper bounds on the possible cost values of local optima within each scenario. We denote these cost bounds by $S_{i,lb}(v)$ and $S_{i,ub}(v)$ for each scenario $i \in \{1, 2, 3, 4\}$. These bounds are functions only of the constant system model parameters collected in v, where

$$v = (\, M,\, c_p,\, R,\, \dot{Q}_{\text{offset}},\, \Delta t,\, \varphi,\, T_{oa},\, \eta_c,\, \eta_h,\, \kappa_A,\, r_e,\, r_h,\, \underline{\dot{m}}_s,\, \overline{\dot{m}}_s,\, \underline{T}_c,\, \overline{T}_h,\, \underline{T}_z,\, \overline{T}_z\,).$$

The derivation of the functions $S_{i,lb}(v)$ and $S_{i,ub}(v)$ is lengthy and reported in Appendix 2.8.

We determine which scenarios can be ruled out using the following steps.

Algorithm 1 *Elimination of scenarios*

1. *If $S_{1,lb}(v) \geq S_{2,ub}(v)$ or $S_{1,lb}(v) \geq S_{3,ub}(v)$ or $S_{1,lb}(v) \geq S_{4,ub}(v)$, then S1 can be ruled out*

2. *If* $S_{2,lb}(v) \geq S_{1,ub}(v)$ *or* $S_{2,lb}(v) \geq S_{3,ub}(v)$ *or* $S_{2,lb}(v) \geq S_{4,ub}(v)$*, then S2 can be ruled out*

3. *If* $S_{3,lb}(v) \geq S_{1,ub}(v)$ *or* $S_{3,lb}(v) \geq S_{2,ub}(v)$ *or* $S_{3,lb}(v) \geq S_{4,ub}(v)$*, then S3 can be ruled out*

4. *If* $S_{4,lb}(v) \geq S_{1,ub}(v)$ *or* $S_{4,lb}(v) \geq S_{2,ub}(v)$ *or* $S_{4,lb}(v) \geq S_{3,ub}(v)$*, then S4 can be ruled out*

We rule out scenarios by including additional constraints in the optimization problem (2.22). The additional constraints restrict the search space to a subset of the scenarios, and eliminate any local optima that would otherwise be found in the removed scenarios. Note that the resulting optimization problem is still non-convex in the remaining scenarios. The additional constraints do however reduce the size of the feasible region of the optimization problem, so may improve convergence speed and simplify the task of identifying multiple local optima. Further iterative refinement to calculate bounds within smaller subregions cannot be performed a-priori because the number of regions required would be prohibitive. Performing this refinement online requires a branch and bound algorithm.

Configuration B

Our simulations revealed the presence of local optima for configuration B in a small instance of the optimization problem (2.22) with three zones and a prediction horizon of 2 steps. System parameter values for this example are given in Table 2.2 and Table 2.5. Compared to the example previously presented for configuration A in Section 2.6 the zone thermal capacitances are higher, the thermal loads are smaller, and the comfort bound temperatures are time-varying. As in Section 2.6, the initial state condition and all other system model parameters are held constant for all simulations. The local optima are generated by repeated execution of the optimization algorithm starting from randomly selected initial guesses.

We found 6 distinct local optima for configuration B in this example. Each of the local optima here was a single point, rather than a connected family of solutions as in configuration A. For all six local optima, only outside air is used so $\dot{m}_s = \dot{m}_{oa}$, and the heating and cooling coils are inactive with $T_c = T_h = 16$ C at all time steps. In this example outside air is sufficient to counteract the small zone thermal loads and cool the zones to remain within the time-varying comfort bounds.

The six optima are qualitatively very similar to one another: in the first time step, one zone is cooled to slightly below 24 C, another zone is cooled to approximately 25 C, and a third zone remains at the initial upper bound of 26 C (see the upper row of Figure 2.11). In the second time step, all 3 zones are controlled to reach the reduced upper bound at 24 C. Mass flow rates are large for time steps when a zone requires a 2 C temperature change, intermediate for steps when a zone requires a 1 C temperature change, and small for steps when the zone temperature change is small (see the lower row of Figure 2.11). The difference

Table 2.5: Configuration B parameters

Description	Symbol	Value	Units
Zone thermal capacitance	$(mc)_i$	8000	kJ / K
Fixed thermal loads	$\dot{Q}_{\text{offset}}$	$[\frac{1}{3}, \frac{1}{2}, \frac{2}{3}]^T$	kW
Fan power coefficient	κ_B	0.4	kW/(kg/s)^2
Max VAV capacity	$\overline{\dot{m}}_{zi}$	6	kg / s
Min zone ventilation	$\underline{\dot{m}}_{zi}$	0.005	kg / s
Upper comfort bound: step 1	$\overline{T}_{zi}(1)$	26	C
Upper comfort bound: step 2	$\overline{T}_{zi}(2)$	24	C
Lower comfort bound: step 1	$\underline{T}_{zi}(1)$	19	C
Lower comfort bound: step 2	$\underline{T}_{zi}(2)$	21	C
Initial state conditions	$T_{zi}(0)$	26	C

between the local optima here is a matter of sequencing: the local optima correspond to the 6 different ordered permutations of 3 zones.

Table 2.6: Result details, configuration B

Point #	t	$\dot{m}_{z1}$ (kg/s)	$\dot{m}_{z2}$ (kg/s)	$\dot{m}_{z3}$ (kg/s)	$\dot{m}_s$ (kg/s)	$J^\star$ ($)
1	0	1.0612	0.5092	0.0667	1.6371	0.11738†
	Δt	0.005	0.5928	1.0617	1.6595	
2	0	1.0612	0.05	0.5232	1.6344	0.11751†
	Δt	0.005	1.0432	0.6158	1.664	
3	0	0.4739	1.1007	0.0667	1.6413	0.11753†
	Δt	0.5908	0.005	1.0617	1.6575	
4	0	0.0333	1.1007	0.5019	1.6359	0.11779†
	Δt	1.0247	0.005	0.6368	1.6664	
5	0	0.4526	0.05	1.1403	1.643	0.11783†
	Δt	0.6119	1.0432	0.005	1.6601	
6	0	0.0333	0.4666	1.1403	1.6402	0.11796†
	Δt	1.0247	0.6348	0.005	1.6645	

Table 2.6 gives the cost values and flow rates for each of the 6 locally optimal solutions. The cost values (†) show very little variation between the local optima, only 0.5% difference between the lowest cost and highest cost points. The differences in cost are due to the different thermal load values for each zone. While we would expect the differences in cost to grow if we increased the differences in zone thermal loads, the optimal solutions for

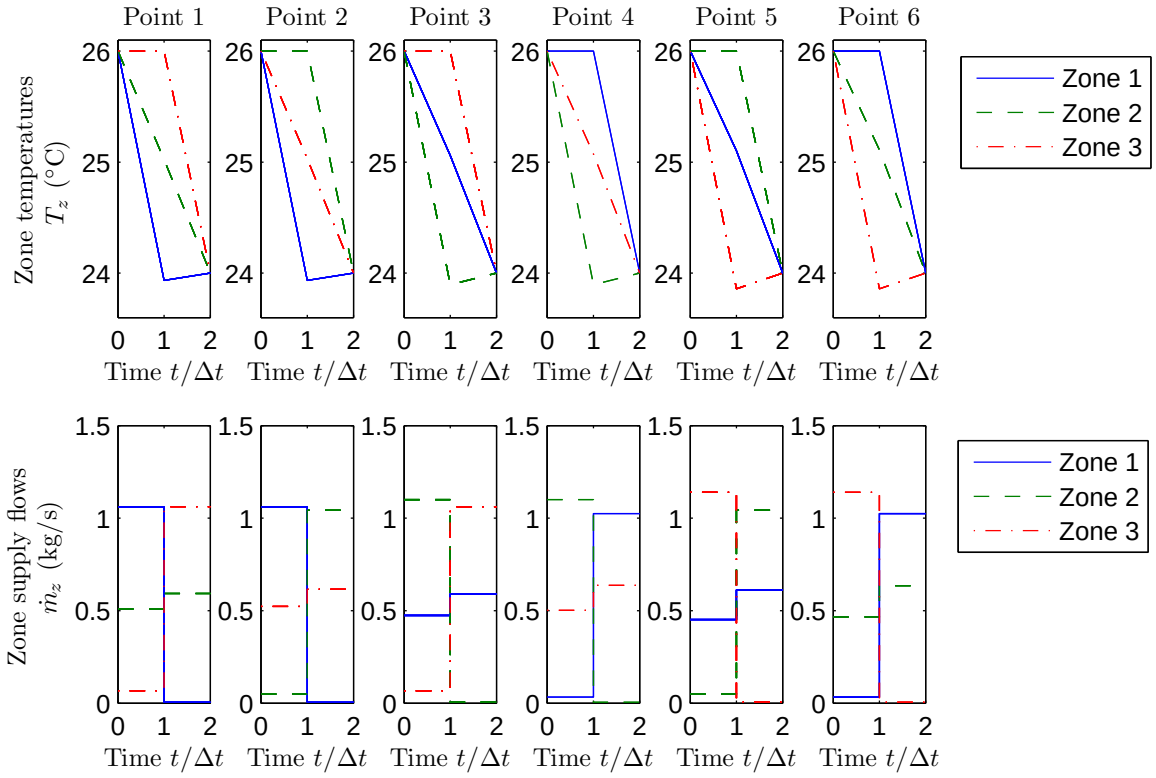


Figure 2.11: Local optima for HVAC configuration B

other control variables such as coil setpoints and percentage of outside air flow may change significantly for different system parameters such as thermal loads. The solution to the optimization problem may vary discontinuously as a function of its parameter data.

For a constant supply temperature, less flow is required to counteract a positive thermal load when zone temperature is higher due to (2.1). So the control strategy that cools the zones with the lowest thermal loads first is the most efficient here, keeping the high-load zones at higher temperatures for longer. This strategy is point 1, so the results confirm this statement. Point 2 partially cools the highest-load zone before the intermediate-load zone, and point 3 fully cools the intermediate-load zone before the lowest-load zone. Point 4 fully cools the lowest-load zone last instead of first but is otherwise in order, and point 5 fully cools the highest-load zone first instead of last but is otherwise in order. Point 6 cools the highest-load zone first and the lowest-load zone last.

The small difference in cost between points 2 and 3, and between points 4 and 5, is a consequence of which zone is fully cooled at the first time step. The fastest-cooled zone (zone 1 for points 1 and 2, zone 2 for points 3 and 4, zone 3 for points 5 and 6) is then supplied with exactly the lower bound minimum flow rate at the second time step. At the minimum

flow rate, the change in zone temperature during the second time step is larger for higher thermal loads. Therefore the higher the thermal load, the more a zone must be overcooled at the first time step in order to remain below 24 C at the second time step. This additional required energy causes points 5 and 6 to have higher cost than points 3 and 4, and points 3 and 4 to have higher cost than points 1 and 2.

Although the local optima in this example have practically equivalent costs, the individual local optima attain that cost by means of different control strategies. Depending on the method used for initializing the optimization algorithm at sequential time steps, the presence of local optima could lead to undesirable oscillatory switching behavior. In larger problem instances with more zones and longer prediction horizons, examples similar to this one could possess a combinatorial number of local optima, corresponding to sequencing permutations. Thorough enumeration of local optima by random sampling quickly becomes intractable for larger problem sizes because both the number of local optima and the computational effort to solve a single instance tend to increase with problem size.

Discretization Comparison

The local optima we observed for configuration B in the previous section were related to transient dynamic behavior. To investigate these local optima in more detail, we revisit the issue of system model discretization. The physics-based model (2.1) is a continuous-time differential equation, and including this continuous-time model directly in a MPC formulation would lead to an infinite dimensional differential-algebraic optimization problem. To solve the underlying continuous control problem computationally, we must numerically discretize time. We choose to perform a fixed-step discretization so the size of the resulting discrete optimization problem is known in advance.

In (2.5) and all results presented above, we used the trapezoidal discretization method. In this section we compare the results of forward Euler (abbreviated FE), trapezoidal (TR), and exact (EX) discretization methods on a modified version of the example from Section 2.6.

(FE) Forward Euler: the simplest method to implement, it is explicit but only first-order accurate in the timestep. Forward Euler is numerically unstable if the discrete time step is larger than the time constant of the continuous system dynamics. The resulting discrete system from the forward Euler discretization method is

$$\frac{T_z^+ - T_z}{\Delta t} = A(u_z)T_z + B(u_z) + w. \tag{2.26}$$

(TR) Trapezoidal method: equivalent to approximating the time variation of zone temperatures as piecewise linear. This method is second-order accurate in the timestep and has the desirable property of A-stability [18], meaning the trapezoidal discretization of any stable continuous-time system gives a stable discrete-time system. The trapezoidal method is implicit so requires solution of a nonlinear system of equations at

each timestep. These implicit systems of equations are included in our optimization formulation in constraint (2.21a) and solved by the nonlinear programming algorithm. The resulting discrete system from the trapezoidal discretization method is

$$\frac{T_z^+ - T_z}{\Delta t} = A(u_z)\frac{T_z^+ + T_z}{2} + B(u_z) + w. \tag{2.27}$$

(EX) Exact discretization: reproduces the exact solution values at the discrete sampling times. A closed-form discrete solution is not always possible for a general nonlinear differential equation. Since the continuous dynamics (2.4) have state-affine form and we are assuming the control inputs and disturbances are zero-order held, there is a solution given by the matrix exponential for this system. The resulting discrete system from the exact discretization is

$$A(u_z)T_z^+ + B(u_z) + w = e^{A(u_z)\Delta t}(A(u_z)T_z + B(u_z) + w), \tag{2.28}$$

where $e^{A(u_z)\Delta t}$ is the matrix exponential.

Each of the above discretization methods is an approximation to the same continuous-time problem, but different discretizations result in different discrete-time optimization problems (2.22). Changing the discretization method modifies the structure and form of nonlinearities in the equality constraints (2.21a). Different discretization methods can therefore have different optimal cost and solution values.

Next we present 4 aspects of the effect of discretization method: (1) existence of local optima, (2) shape of cost function around local optima, (3) verification of local optima, and (4) extension to longer prediction horizon.

Existence of Local Optima

The solution method and parameters used here are identical to those in Section 2.6, with two exceptions. Here we consider only two zones ($n = 2$) with fixed thermal loads $\dot{Q}_{\text{offset}} = [1/3,\ 2/3]^T$. Note that with 3 zones we observed 6 local minima corresponding to the 6 different ordered permutations of 3 zones, but now with 2 zones there are only 2 possible ordered permutations.

Table 2.7 summarizes the (locally) optimal solutions for each of the discretization methods. The local optimum numbered as 1 is the observed solution with lowest cost (presumed but not proven to be the global optimum) for each method. It is observed that forward Euler and the trapezoidal method have two locally optimal solutions, while the exact discretization results in a single optimal solution in this example. Furthermore, the local solutions with the trapezoidal discretization have much closer cost values to the exact discretization than do the solutions with forward Euler. This is as expected, due to the higher order of accuracy of the trapezoidal method and the transient nature of the solutions in this example.

The substantially lower cost values of the locally optimal solutions for the forward Euler method are also due to discretization inaccuracy. The energy delivered to a zone is

Table 2.7: Locally optimal solutions for different discretization methods, configuration B

Discretization method	Average runtime (s)	Solution number	$[T_{z1}, T_{z2}]$ (C) at time $t = \Delta t$	$J^\star$ (\$)	Cost variation relative to EX
FE	0.244	1	[23.93, 26.00]	0.042325	–19.32%
		2	[26.00, 23.86]	0.042458	–19.06%
TR	0.327	1	[23.98, 26.00]	0.052022	–0.83%
		2	[26.00, 23.99]	0.052419	–0.07%
EX	0.787	1	[24.00, 26.00]	0.052458	0

proportional to flow rate times the difference between supply and zone temperatures. The forward Euler method treats that temperature difference as constant throughout a time step, whereas in reality the temperature difference decreases over time as a zone is cooled. Hence the forward Euler method predicts that a lower flow rate is required to deliver the same total energy to a zone.

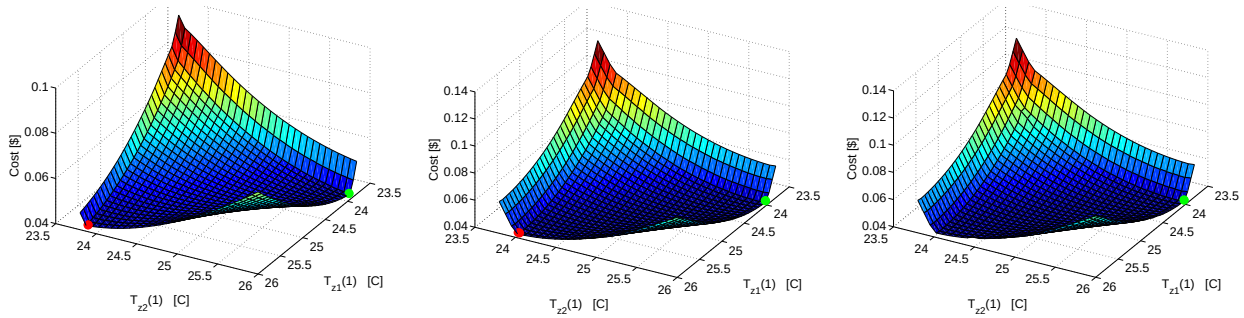
Table 2.7 also lists the average solver time for the different discretization methods. The exact discretization is more time consuming than forward Euler or the trapezoidal method due to exponential function computations. This implementation used unoptimized Matlab code for the objective and constraint functions via the Yalmip toolbox, but we expect that an implementation with optimized code for real-time execution on a much larger-scale system would show similar trends.

Shape of Cost Function Around Local Optima

In Figure 2.12 we explore the optimal cost as a function of zone temperatures at the intermediate time step $t = \Delta t$ for each discretization method. This is achieved by including additional constraints fixing the zone temperature values T_z at time $t = \Delta t$, and executing a modified version of problem (2.22) for each entry on a sampled grid of vectors T_z. The locally optimal solutions from Table 2.7 are marked by a green dot at each lower-cost solution 1, and a red dot at each higher-cost solution 2 (for FE and TR, but not for EX since it has only one solution).

We make several observations from Figure 2.12. For all 3 discretization methods the optimal cost is not very sensitive to the sum of the zone temperatures at time $t = \Delta t$, but is highly sensitive to the difference between the zone temperatures at time $t = \Delta t$. In other words, as long as the overall cooling delivered to both zones sums to 2 C total temperature change over the first time step, the energy cost over the whole horizon is close to optimal. This corresponds to the intuition that it is better to operate the system at a lower rate for a longer period of time. If both zones are cooled by too much or too little at the first time step, the total energy cost will be significantly higher.

Additionally, the cost values with the trapezoidal method are very similar to the cost values with the exact discretization. The cost values for forward Euler did not match as



(a) Forward Euler (b) Trapezoidal method (c) Exact discretization

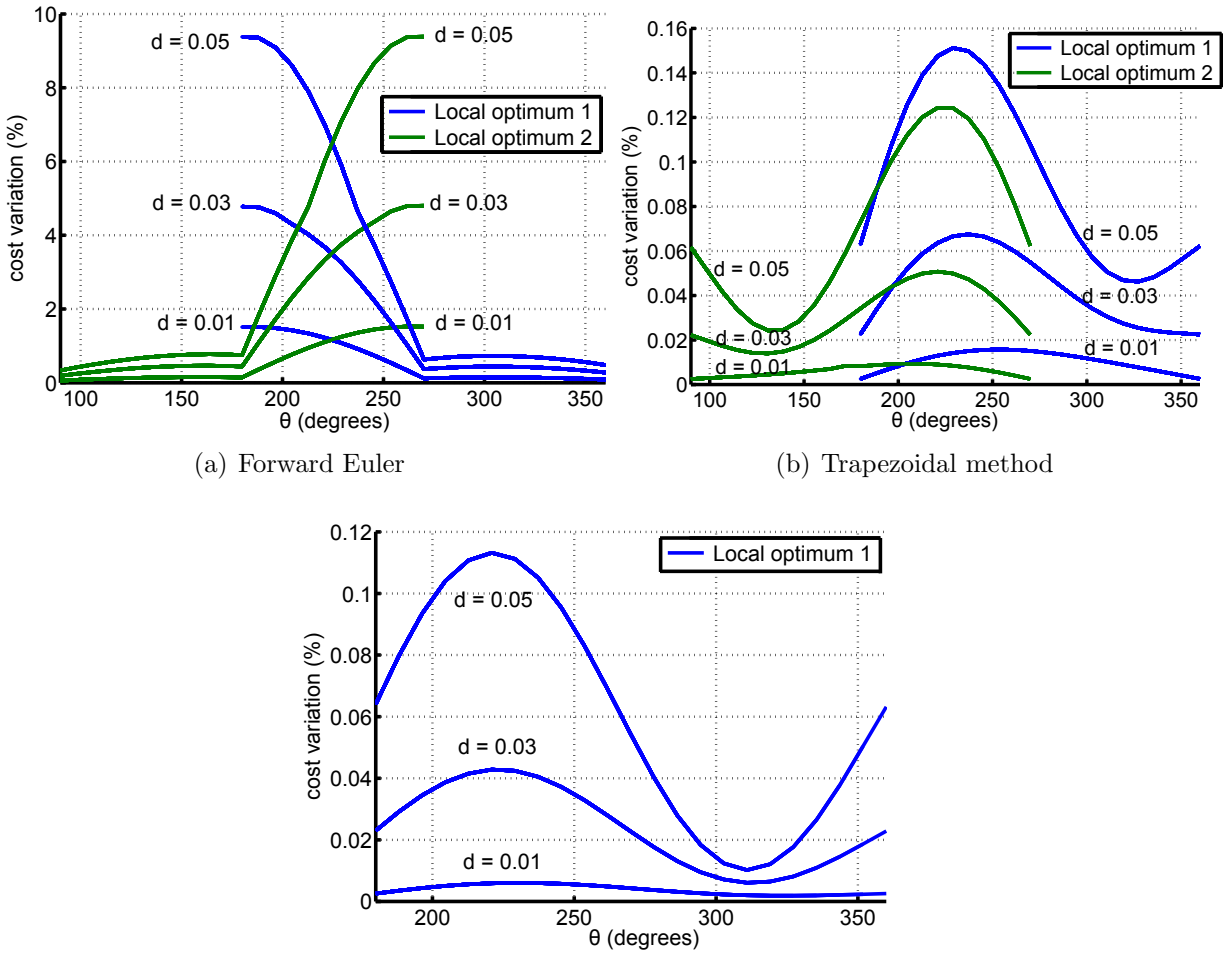
Figure 2.12: Visualization of optimal cost values as a function of zone temperatures at the intermediate time step $t = \Delta t$

well due to the inaccuracy of the FE discretization. In the exact discretization, the cost varies monotonically along the floor of the valley between the points $T_z = [24, 26]^T$ and $T_z = [26, 24]^T$ and there is only one local optimum. The trapezoidal and forward Euler discretizations, on the other hand, show a saddle-shaped cost behavior leading to a second local optimum close to $T_z \approx [26, 24]^T$. The non-convex direction of the saddle shape is much more pronounced for the forward Euler discretization.

Verification of Local Optima

To verify that the solutions found in Table 2.7 are all local optima, we examine the cost variation in the neighborhood around each identified solution $T_z^\star$ by fixing $T_z = T_z^\star + [d\cos\theta,\ d\sin\theta]^T$ at time $t = \Delta t$. Note that for local optima with $T_z^\star \approx [24, 26]^T$ at time $t = \Delta t$ (numbered 1 and marked by green dots in Figure 2.12), only values of $\theta \in [180, 360]^\circ$ will result in feasible $T_z = T_z^\star + [d\cos\theta,\ d\sin\theta]^T$ inside the comfort region. For local optima with $T_z^\star \approx [26, 24]^T$ at time $t = \Delta t$ (numbered 2 and marked by red dots in Figure 2.12), only values of $\theta \in [90, 270]^\circ$ will result in feasible $T_z = T_z^\star + [d\cos\theta,\ d\sin\theta]^T$ inside the comfort region.

In Figure 2.13 we plot the cost variation of these neighboring points $T_z = T_z^\star + [d\cos\theta,\ d\sin\theta]^T$ with respect to each local optimum as a function of θ, for 3 small values of d. The positive cost variation across all feasible values of θ indicates that the cost value increases in every feasible direction away from the local optima. If we were to plot similar curves for cost variation of the exact discretization around the point $T_z \approx [26, 24]^T$, the cost variation would be negative for θ near 135°. This point was not a local optimum for the exact discretization. We also note that the vertical scale for the cost variation is two orders of magnitude larger for the forward Euler discretization than the other two. Possibly due to the more pronounced saddle non-convexity observed in Figure 2.12, the forward Euler method has larger local slopes in the vicinity of the local optima.



(a) Forward Euler

(b) Trapezoidal method

(c) Exact discretization

Figure 2.13: Cost variation in a neighborhood around each local optimum solution

Extension to Longer Prediction Horizon

In Table 2.8 we show the number of local optima we observed with an increasing length of prediction horizon from $N = 2$ to $N = 6$. The forward Euler discretization shows an increasing number of locally optimal solutions with longer prediction horizon N, and the exact discretization results in a single optimal solution for all horizon lengths considered. The trapezoidal discretization exhibited multiple optima for prediction horizon length N up to four, but only one optimal solution for $N \geq 5$. This result for the trapezoidal method is unexpected and requires further analysis.

The results of this section present some interesting challenges. The discretized system dynamics (2.26), (2.27), and (2.28) all contain bilinear product terms between temperature states and flow inputs, as well as between temperature inputs and flow inputs. So the exact

Table 2.8: Number of local optima with various prediction horizons and discretization methods

	N=2	N=3	N=4	N=5	N=6
FE	2	6	9	17	42
TR	2	6	3	1	1
EX	1	1	1	1	1

discretization (2.28) is also a non-convex problem, but for our example it demonstrated cost variation (with respect to a small subset of intermediate state variables) that appeared to be convex or very nearly convex. It may be possible to transform or reformulate this problem in such a way that it decomposes into a convex part and a non-convex part, which can then be taken advantage of by an optimization algorithm.

The exact discretization may also be difficult to compute if the matrix $A(u_z)$ is not diagonal, for example when heat transfer between neighboring zones is included. The trapezoidal discretization approximates the cost values of the exact discretization with a practically acceptable accuracy, and one of its local optima is a close approximation of the exact discretization's optimal solution. However the trapezoidal discretization introduced additional local optima (for horizon lengths $N = 2, 3$, or 4) at points where the exact discretization would have identified a descent direction, and the control strategies at these additional local optima are quite different. In our particular example for this configuration all identified local optima had similar cost values to one another, but this might not be the case for different system parameters.

2.7 Conclusions

We have studied the problem of using model predictive control (MPC) for heating, ventilation, and air conditioning (HVAC) systems in a typical commercial building for two common HVAC configurations. We derived simplified physics-based nonlinear system models for each configuration, then investigated the phenomenon of local optima by solving the nonlinear MPC problem using the local nonlinear programming solver Ipopt. We have shown by simulation that the resulting optimization problem exhibits local optima for both configurations. We performed a detailed analysis of the different types of local optima and their physical interpretation.

For the dual-duct HVAC configuration, the local optima in our example results had notably different energy costs, so finding a globally optimal control strategy has obvious significance. We have divided the optimization search space into 4 regions and proposed an algorithm that can exclude one or more of those regions from consideration a priori. Physics-based rules like the ones we have identified are particularly useful for a branch and bound scheme, as they reduce the size of the search space when they can be applied.

For the single-duct configuration, the local optima in our example results had very similar cost values but differed in transient sequencing. We have shown that the choice of discretization method has an important influence on the existence, number, and characteristics of local optima. The issue of discretization method with physics-based continuous-time models for nonlinear MPC deserves further study. A partially convex decomposition, if possible, may make up for some of the computational drawbacks of the exact discretization.

2.8 Detailed Derivation for Physics-based Rules

Here we present derivation steps for the physics-based rules from Section 2.6. We begin by integrating the continuous-time dynamics (2.1) over a time step of Δt assuming the control inputs of supply flow rate and supply temperature are held piecewise constant, giving

$$(mc)_i(T_{zi}^+ - T_{zi}) = \Delta t(\dot{Q}_{i,avg} + \dot{m}_{zi}c_p(T_{si} - T_{zi,avg})), \tag{2.29}$$

where $\dot{Q}_{i,avg} = \frac{1}{\Delta t}\int_t^{t+\Delta t} \dot{Q}_i d\tau$ and $T_{zi,avg} = \frac{1}{\Delta t}\int_t^{t+\Delta t} T_{zi} d\tau$. Solving for T_{si}, we obtain

$$T_{si} = T_{zi,avg} + \frac{1}{\dot{m}_{zi}c_p}\left(\frac{(mc)_i}{\Delta t}(T_{zi}^+ - T_{zi}) - \dot{Q}_{i,avg}\right). \tag{2.30}$$

$$\text{Let } \beta_i = \frac{1}{c_p}\left(\frac{(mc)_i}{\Delta t}(T_{zi}^+ - T_{zi}) - \dot{Q}_{i,avg}\right), \text{ then } T_{si} = T_{zi,avg} + \frac{1}{\dot{m}_{zi}}\beta_i. \tag{2.31}$$

If $\beta_i > 0$, then decreasing $\dot{m}_{zi}$ (leaving all other variables unchanged) increases supply temperature T_{si}. If $\beta_i < 0$, then decreasing $\dot{m}_{zi}$ decreases supply temperature T_{si}.

Zone and return temperatures can be bounded by applying the comfort constraints (2.20f), $\underline{T}_{zi} \le T_{zi,avg} \le \overline{T}_{zi}$ which implies $\min_i \underline{T}_{zi} \le T_{r,avg} \le \max_i \overline{T}_{zi}$. The mixed temperature can be bounded using (2.10) and (2.20b), giving $\min(T_{r,avg}, T_{oa}) \le T_m \le \max(T_{r,avg}, T_{oa})$ therefore $\min(\min_i \underline{T}_{zi}, T_{oa}) \le T_m \le \max(\max_i \overline{T}_{zi}, T_{oa})$.

We can bound β_i by applying the comfort constraints as follows. Let $R_+ = \max(R, 0)$ and $R_- = \min(R, 0)$ taken elementwise. Then $\underline{\beta}_i \le \beta_i \le \overline{\beta}_i$, where

$$\begin{aligned} \underline{\beta}_i &= \frac{1}{c_p}\left(\frac{(mc)_i}{\Delta t}(\underline{T}_{zi} - \overline{T}_{zi}) - R_+\overline{T}_{zi} - R_-\underline{T}_{zi} - \dot{Q}_{\text{offset}}\right) \\ \text{and } \overline{\beta}_i &= \frac{1}{c_p}\left(\frac{(mc)_i}{\Delta t}(\overline{T}_{zi} - \underline{T}_{zi}) - R_+\underline{T}_{zi} - R_-\overline{T}_{zi} - \dot{Q}_{\text{offset}}\right). \end{aligned} \tag{2.32}$$

Applying (2.31), the sum of supply temperatures times flow rates is

$$\sum_{i=1}^{n} \dot{m}_{zi}T_{si} = \sum_{i=1}^{n}(\dot{m}_{zi}T_{zi,avg} + \beta_i) = \dot{m}_s T_{r,avg} + \sum_{i=1}^{n}\beta_i, \tag{2.33}$$

where $T_{r,avg} = \frac{1}{\Delta t}\int_t^{t+\Delta t} T_r d\tau$. The second equality in (2.33) is due to (2.9).

Scenario S1

In scenario S1, $T_c = T_{s,min}$ and $T_h = T_{s,max}$. Substituting (2.33) into the cost (2.24) gives

$$\begin{aligned} r_e P_c + r_h P_h = &\frac{r_e c_p \dot{m}_s (T_m - T_{s,min})}{\eta_c \dot{m}_s (T_{s,max} - T_{s,min})} \left(\dot{m}_s (T_{s,max} - T_{r,avg}) - \sum_{i=1}^{n} \beta_i \right) \\ &+ \frac{r_h c_p \dot{m}_s (T_{s,max} - T_m)}{\eta_h \dot{m}_s (T_{s,max} - T_{s,min})} \left(\dot{m}_s (T_{r,avg} - T_{s,min}) + \sum_{i=1}^{n} \beta_i \right). \end{aligned} \tag{2.34}$$

Let $j = \arg\min_i T_{si}$ and $k = \arg\max_i T_{si}$. Substituting $T_{s,min} = T_{zj,avg} + \frac{1}{\varphi_j \dot{m}_s} \beta_j$ and $T_{s,max} = T_{zk,avg} + \frac{1}{\varphi_k \dot{m}_s} \beta_k$ into (2.34) and adding the cubic fan cost (2.15) gives

$$\begin{aligned} &r_e P_f + r_e P_c + r_h P_h = \\ &\quad \kappa_A \dot{m}_s^3 + \frac{r_e c_p \left(\dot{m}_s T_m - \dot{m}_s T_{zj,avg} - \frac{\beta_j}{\varphi_j} \right) \left(\dot{m}_s T_{zk,avg} + \frac{\beta_k}{\varphi_k} - \dot{m}_s T_{r,avg} - \sum_{i=1}^{n} \beta_i \right)}{\eta_c \left(\dot{m}_s T_{zk,avg} + \frac{\beta_k}{\varphi_k} - \dot{m}_s T_{zj,avg} - \frac{\beta_j}{\varphi_j} \right)} \\ &\quad + \frac{r_h c_p \left(\dot{m}_s T_{zk,avg} + \frac{\beta_k}{\varphi_k} - \dot{m}_s T_m \right) \left(\dot{m}_s T_{r,avg} - \dot{m}_s T_{zj,avg} - \frac{\beta_j}{\varphi_j} + \sum_{i=1}^{n} \beta_i \right)}{\eta_h \left(\dot{m}_s T_{zk,avg} + \frac{\beta_k}{\varphi_k} - \dot{m}_s T_{zj,avg} - \frac{\beta_j}{\varphi_j} \right)}. \end{aligned} \tag{2.35}$$

We can combine (2.35) into a single rational polynomial function of $\dot{m}_s$ with quartic numerator and linear denominator. We note that (2.34) and (2.35) cannot have singularities in scenario S1 because the denominator term $(T_{s,max} - T_{s,min}) \geq 0$ by construction, with equality only possible for S4 (where there are additional cancellations, see Section 2.8). The partial derivative $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h)$ can be combined into a rational polynomial function of $\dot{m}_s$ with quartic numerator and quadratic denominator, and also has no singularities in S1. Local minima can therefore only occur at real roots of the numerator of $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h)$, or at the minimum or maximum feasible flow rates for scenario S1.

The range of flow rates $\dot{m}_s$ for which S1 is feasible is determined by (2.31), the constraints (2.18b) and (2.18c) which require $T_{s,max} \leq \overline{T}_h$, the constraints (2.20d) and (2.20e) which require $T_{s,min} \geq \underline{T}_c$, and the requirement for scenario S1 that $T_{s,min} \leq T_m \leq T_{s,max}$.

If $\beta_i > 0$ for all zones i, then

$$\max\left(\underline{\dot{m}}_s, \ \frac{\beta_k}{\varphi_k (\overline{T}_h - T_{zk,avg})}, \ \frac{\beta_j}{\varphi_j (T_m - T_{zj,avg})} \right) \leq \dot{m}_s \leq \overline{\dot{m}}_s. \tag{2.36}$$

If $\beta_i < 0$ for all zones i, then

$$\max\left(\underline{\dot{m}}_s, \ \frac{\beta_j}{\varphi_j (\underline{T}_c - T_{zj,avg})}, \ \frac{\beta_k}{\varphi_k (T_m - T_{zk,avg})} \right) \leq \dot{m}_s \leq \overline{\dot{m}}_s. \tag{2.37}$$

We calculate the upper and lower cost bounds for S1 as follows.

Algorithm 2 *Cost bounds for scenario S1*

1. *Initialize* $S_{1,lb}(v) = \infty$ *and* $S_{1,ub}(v) = \infty$
2. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) > 0$ *at minimum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at minimum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{1,lb}(v) = (r_e P_f + r_e P_c + r_h P_h)_{lb}$
   d) *Set* $S_{1,ub}(v) = (r_e P_f + r_e P_c + r_h P_h)_{ub}$
3. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) < 0$ *at maximum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at maximum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{1,lb}(v) = \min(S_{1,lb}, (r_e P_f + r_e P_c + r_h P_h)_{lb})$
   d) *Set* $S_{1,ub}(v) = \min(S_{1,ub}, (r_e P_f + r_e P_c + r_h P_h)_{ub})$
4. *For each real root of* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 0$ *within feasible range of flow rates* $\dot{m}_s$*,*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at root of* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 0$
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{1,lb}(v) = \min(S_{1,lb}, (r_e P_f + r_e P_c + r_h P_h)_{lb})$
   d) *Set* $S_{1,ub}(v) = \min(S_{1,ub}, (r_e P_f + r_e P_c + r_h P_h)_{ub})$

Scenario S2

In scenario S2, $T_h = T_m$ so substituting (2.33) into the cost (2.24) gives

$$r_e P_c + r_h P_h = \frac{r_e c_p}{\eta_c} \sum_{i=1}^{n} \dot{m}_{zi}(T_m - T_{si}) = \frac{r_e c_p}{\eta_c} \left(\dot{m}_s (T_m - T_{r,avg}) - \sum_{i=1}^{n} \beta_i \right). \tag{2.38}$$

Mixed temperature T_m must be between return temperature $T_{r,avg}$ and outside air temperature T_{oa}, so if outside air is warmer than return temperature then T_m should be set equal to $T_{r,avg}$ and supply flow should be minimized (due to fan cost).

If outside air is cooler than return temperature and $\beta_i > 0$ for all zones i, then scenario S2 is infeasible because

- $T_m \leq \max(T_{r,avg}, T_{oa})$ from (2.10) and (2.20b),
- $T_{si} = T_{zi,avg} + \frac{1}{\dot{m}_{zi}}\beta_i > T_{zi,avg}$ by (2.31) when $\beta_i > 0$, so
- $\dot{m}_s T_{s,max} \geq \sum_{i=1}^{n} \dot{m}_{zi} T_{si} > \sum_{i=1}^{n} \dot{m}_{zi} T_{zi,avg} = \dot{m}_s T_{r,avg}$, last equality due to (2.9).

When $T_{r,avg} \geq T_{oa}$ this contradicts the requirement for scenario S2 that $T_m \geq T_{s,max}$.

If $\beta_i < 0$ for all zones i, then scenario S2 may be feasible when $T_{r,avg} \geq T_{oa}$, and the partial derivative $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h)$ will have a real root where

$$\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 3\kappa_A \dot{m}_s^2 + \frac{r_e c_p}{\eta_c}(T_m - T_{r,avg}) = 0. \tag{2.39}$$

The range of flow rates $\dot{m}_s$ for which S2 is feasible is determined by (2.31), the constraints (2.20d) and (2.20e) which require $T_{s,min} \geq \underline{T}_c$, and the requirement for scenario S2 that $T_{s,max} \leq T_m$.

Again letting $j = \arg\min_i T_{si}$ and $k = \arg\max_i T_{si}$, the flow rate $\dot{m}_s$ must be in the range

$$\max\left(\underline{\dot{m}}_s,\ \frac{\beta_j}{\varphi_j(\underline{T}_c - T_{zj,avg})},\ \frac{\beta_k}{\varphi_k(T_m - T_{zk,avg})}\right) \leq \dot{m}_s \leq \overline{\dot{m}}_s. \tag{2.40}$$

We calculate the upper and lower cost bounds for S2 as follows.

Algorithm 3 *Cost bounds for scenario S2*

1. *Initialize* $S_{2,lb}(v) = \infty$ *and* $S_{2,ub}(v) = \infty$
2. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) > 0$ *at minimum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at minimum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{2,lb}(v) = (r_e P_f + r_e P_c + r_h P_h)_{lb}$
   d) *Set* $S_{2,ub}(v) = (r_e P_f + r_e P_c + r_h P_h)_{ub}$
3. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) < 0$ *at maximum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at maximum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{2,lb}(v) = \min(S_{2,lb}, (r_e P_f + r_e P_c + r_h P_h)_{lb})$
   d) *Set* $S_{2,ub}(v) = \min(S_{2,ub}, (r_e P_f + r_e P_c + r_h P_h)_{ub})$
4. *For each real root of* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 0$ *within feasible range of flow rates* $\dot{m}_s$*,*

a) Evaluate $(r_eP_f + r_eP_c + r_hP_h)$ *at root of* $\frac{\partial}{\partial \dot{m}_s}(r_eP_f + r_eP_c + r_hP_h) = 0$

b) Calculate bounds $(r_eP_f + r_eP_c + r_hP_h)_{lb}$ *and* $(r_eP_f + r_eP_c + r_hP_h)_{ub}$ *using comfort constraints and mixed temperature range*

c) Set $S_{2,lb}(v) = \min(S_{2,lb}, (r_eP_f + r_eP_c + r_hP_h)_{lb})$

d) Set $S_{2,ub}(v) = \min(S_{2,ub}, (r_eP_f + r_eP_c + r_hP_h)_{ub})$

Scenario S3

In scenario S3, $T_c = T_m$ so substituting (2.33) into the cost (2.24) gives

$$r_eP_c + r_hP_h = \frac{r_hc_p}{\eta_h}\sum_{i=1}^{n}\dot{m}_{zi}(T_{si} - T_m) = \frac{r_hc_p}{\eta_h}\left(\dot{m}_s(T_{r,avg} - T_m) + \sum_{i=1}^{n}\beta_i\right). \tag{2.41}$$

Mixed temperature T_m must be between return temperature $T_{r,avg}$ and outside air temperature T_{oa}, so if outside air is cooler than return temperature then T_m should be set equal to $T_{r,avg}$ and supply flow should be minimized (due to fan cost).

If outside air is warmer than return temperature and $\beta_i < 0$ for all zones i, then scenario S3 is infeasible because

- $T_m \geq \min(T_{r,avg}, T_{oa})$ from (2.10) and (2.20b),
- $T_{si} = T_{zi,avg} + \frac{1}{\dot{m}_{zi}}\beta_i < T_{zi,avg}$ by (2.31) when $\beta_i < 0$, so
- $\dot{m}_sT_{s,min} \leq \sum_{i=1}^{n}\dot{m}_{zi}T_{si} < \sum_{i=1}^{n}\dot{m}_{zi}T_{zi,avg} = \dot{m}_sT_{r,avg}$, last equality due to (2.9).

When $T_{r,avg} \leq T_{oa}$ this contradicts the requirement for scenario S3 that $T_m \leq T_{s,min}$.

If $\beta_i > 0$ for all zones i, then scenario S3 may be feasible when $T_{r,avg} \leq T_{oa}$, and the partial derivative $\frac{\partial}{\partial \dot{m}_s}(r_eP_f + r_eP_c + r_hP_h)$ will have a real root where

$$\frac{\partial}{\partial \dot{m}_s}(r_eP_f + r_eP_c + r_hP_h) = 3\kappa_A\dot{m}_s^2 + \frac{r_hc_p}{\eta_h}(T_{r,avg} - T_m) = 0. \tag{2.42}$$

The range of flow rates $\dot{m}_s$ for which S3 is feasible is determined by (2.31), the constraints (2.18b) and (2.18c) which require $T_{s,max} \leq \overline{T}_h$, and the requirement for scenario S3 that $T_{s,min} \geq T_m$.

Again letting $j = \arg\min_i T_{si}$ and $k = \arg\max_i T_{si}$, the flow rate $\dot{m}_s$ must be in the range

$$\max\left(\underline{\dot{m}}_s,\ \frac{\beta_k}{\varphi_k(\overline{T}_h - T_{zk,avg})},\ \frac{\beta_j}{\varphi_j(T_m - T_{zj,avg})}\right) \leq \dot{m}_s \leq \overline{\dot{m}}_s. \tag{2.43}$$

We calculate the upper and lower cost bounds for S3 as follows.

Algorithm 4 *Cost bounds for scenario S3*

1. *Initialize* $S_{3,lb}(v) = \infty$ *and* $S_{3,ub}(v) = \infty$
2. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) > 0$ *at minimum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at minimum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{3,lb}(v) = (r_e P_f + r_e P_c + r_h P_h)_{lb}$
   d) *Set* $S_{3,ub}(v) = (r_e P_f + r_e P_c + r_h P_h)_{ub}$
3. *If* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) < 0$ *at maximum feasible flow rate* $\dot{m}_s$*, then*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at maximum feasible flow rate*
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{3,lb}(v) = \min(S_{3,lb}, (r_e P_f + r_e P_c + r_h P_h)_{lb})$
   d) *Set* $S_{3,ub}(v) = \min(S_{3,ub}, (r_e P_f + r_e P_c + r_h P_h)_{ub})$
4. *For each real root of* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 0$ *within feasible range of flow rates* $\dot{m}_s$*,*
   a) *Evaluate* $(r_e P_f + r_e P_c + r_h P_h)$ *at root of* $\frac{\partial}{\partial \dot{m}_s}(r_e P_f + r_e P_c + r_h P_h) = 0$
   b) *Calculate bounds* $(r_e P_f + r_e P_c + r_h P_h)_{lb}$ *and* $(r_e P_f + r_e P_c + r_h P_h)_{ub}$ *using comfort constraints and mixed temperature range*
   c) *Set* $S_{3,lb}(v) = \min(S_{3,lb}, (r_e P_f + r_e P_c + r_h P_h)_{lb})$
   d) *Set* $S_{3,ub}(v) = \min(S_{3,ub}, (r_e P_f + r_e P_c + r_h P_h)_{ub})$

Scenario S4

In scenario S4, $T_c = T_m$ and $T_h = T_m$ so the coil cost $r_e P_c + r_h P_h = 0$. However, scenario S4 is only feasible if the flow rate $\dot{m}_s$ is such that every zone supply temperature has the same value $T_{si} = T_{zi,avg} + \frac{1}{\varphi_i \dot{m}_s}\beta_i = T_m$, so

$$\dot{m}_s = \frac{\beta_i}{\varphi_i (T_m - T_{zi,avg})}, \ \forall \ i. \tag{2.44}$$

We calculate the upper and lower cost bounds for S4 as follows.

Algorithm 5 *Cost bounds for scenario S4*

1. *Calculate bounds* $\dot{m}_{s,lb}$ *and* $\dot{m}_{s,ub}$ *of (2.44) using comfort constraints and mixed temperature range*

2. *If* $\dot{m}_{s,lb} > \overline{\dot{m}}_s$ *or* $\dot{m}_{s,ub} < \underline{\dot{m}}_s$*, then S4 is infeasible so set* $S_{4,lb}(v) = \infty$ *and* $S_{4,ub}(v) = \infty$
3. *Otherwise set* $S_{4,lb}(v) = r_e \kappa_A \dot{m}^3_{s,lb}$ *and* $S_{4,ub}(v) = r_e \kappa_A \dot{m}^3_{s,ub}$

Chapter 3

Computational Challenges for Large Scale Real Time Optimization

In this work we discuss methods of implementing nonlinear predictive control on a parallel computational platform. We focus on the ability of these methods to handle, in a systematic way, large-scale systems with nonlinearities and constraints. We propose a method based on parallel linear algebra algorithms and apply this method to energy efficient control of commercial buildings. The proposed model predictive controller (MPC) minimizes energy consumption while satisfying occupant thermal comfort by using predictive knowledge of weather and occupancy profiles.

We present a numerical study examining the capabilities of state-of-the-art interior point optimization solvers to utilize parallel linear algebra. We are particularly concerned with the sparsity of large scale MPC problems and how well the linear algebra algorithms are able to take advantage of that sparsity.

This chapter is based on work that has been previously published in [36].

3.1 Control Design With Predictions and Constraints

Our goal in this work is to design a controller which:

1. satisfies constraints on states and control inputs,
2. incorporates predictions of future disturbances,
3. accounts for and can handle system nonlinearities, and
4. can be implemented in real-time for a large scale system (hundreds to thousands of states and inputs).

We categorize established control design approaches for addressing these issues into 3 general families.

Ad-hoc decentralized design

The first approach corresponds to current practice in a number of industries. Simple PID controllers are designed for small local control loops, with manually tuned or scheduled gains, setpoints, and modes of operation. Controllers of different subsystems communicate in a limited fashion using application-specific heuristic coordination schemes. These schemes are based on the accumulated experience of application engineers over many years.

This design method is labor-intensive for large scale systems, but the resulting controllers are not computationally demanding; they can readily be implemented using inexpensive embedded hardware. This typically leads to a distributed layout of local sensing, computation, and actuation hardware, with communication between separate controllers over a wired or wireless network.

The disadvantage of this method is verifying performance and constraint satisfaction requires extensive tuning and simulation. Prediction data can be incorporated in certain setpoint or mode schedules, but this amplifies the need for tuning and simulation, and can rarely be done in a systematic manner. As observed in practice, performance is highly variable and constraints are often violated.

A specific example of coordination heuristics from the heating, ventilation, and air conditioning (HVAC) industry is known as trim and respond [61]. If a thermal zone is close to violating a comfort constraint and the local zone controller is saturated at its input limit, that zone controller sends a request to a higher-level air handling unit (AHU) controller. Based on the combined requests from all of the zones it serves, that AHU controller makes slow adjustments to the supply fan speed or cooling coil supply temperature. These trim and respond (also known as supply pressure reset or supply temperature reset) heuristics are observed to reduce energy consumption, but can be difficult to tune and deploy so are currently considered advanced methods within the HVAC industry.

Centralized design of distributed policy

In this approach a centralized control problem is posed to determine the optimal inputs (subject to constraints and forecasts), but rather than solving that problem directly, a distributed policy of local computation and pairwise communication is designed to obtain the same solution iteratively. This is a well-studied approach, building on a great deal of distributed control and optimization literature [8].

Examples of this approach include dual decomposition [40, 42, 46, 57] and the alternating direction method of multipliers [12, 39]. These methods perform well and have been applied successfully on linear or unconstrained systems, but for nonlinear constrained systems these methods are not necessarily guaranteed to converge to a valid solution of the original centralized problem.

Distributed computation of centralized controller

Centralized solution of predictive control for a large scale, constrained, nonlinear system is computationally demanding. Serial performance of computer hardware has largely plateaued for the past 5-10 years due to power and thermal limitations. Improvements in technology following Moore's law have continued, but are now being applied to increase parallelism rather than serial performance [5].

Several authors have investigated parallel approaches to improve the performance of predictive control [16, 32, 44]. Much of this literature has focused on control of linear systems, and using field-programmable gate array (FPGA) architectures. In contrast with this literature, we propose leveraging the research progress and available algorithms from the high performance computing community, specifically parallel direct solvers for sparse systems of linear equations.

In the following sections we present the mathematical statement of nonlinear predictive control and the interior point method for solution of the associated optimization problem.

3.2 Nonlinear Predictive Control Formulation

The basic idea of model predictive control (MPC) is to solve at each time step t the following optimization problem

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{X}} \sum_{k=0}^{N-1} J(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) \\ &\text{subj. to, } \forall\ k \in \{0,\dots,N-1\}, \\ &f(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) = 0 \\ &g(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t}) \leq 0 \\ &x_{0|t} = x(t) \end{aligned} \tag{3.1}$$

where $x_{k|t}$ is the vector of states at time $t + k\Delta t$ predicted at time t, $u_{k|t}$ is the vector of control inputs at time $t + k\Delta t$ predicted at time t, $w_{k|t}$ is the vector of disturbances at time $t + k\Delta t$ predicted at time t, $\mathbf{U} = \{u_{0|t}, \dots, u_{N-1|t}\}$, and $\mathbf{X} = \{x_{1|t}, \dots, x_{N|t}\}$. We denote the number of time steps in the prediction horizon by N. The initial state vector measured at time t is $x(t)$. We denote the contribution to the cost function at each time step by J, the discrete-time system dynamic model (with sample rate Δt) by f, and the constraints by g.

Let the optimal solution of problem (3.1) at time t be denoted by $\mathbf{U}^\star = \{u^\star_{0|t}, \dots, u^\star_{N-1|t}\}$. Then, the first step of $\mathbf{U}^\star$ is input to the system, $u(t) = u^\star_{0|t}$. The optimization (3.1) is repeated at time $t + \Delta t$, with the updated new state $x_{0|t+\Delta t} = x(t + \Delta t)$ yielding a *moving or receding horizon control* strategy.

In this chapter we focus on nominal MPC formulation, i.e. the disturbance forecast $\{w_{0|t}, \dots, w_{N-1|t}\}$ is assumed to be known perfectly in advance. Stochastic MPC formulation accounts for uncertainty in predicted disturbances and model mismatch as part of the

control design. We would also like stochastic MPC to satisfy properties 1-4 from section 3.1. However this requires several additional steps to translate a stochastic control problem into a tractable deterministic computation. There are important issues to consider such as managing the tradeoff between conservatism and complexity, optimizing over nominal inputs versus feedback policies, performing nonlinear system identification with measured data versus using linear models and accounting for larger model errors, using explicit chance constraints versus sample-based methods, propagating uncertainty and risk allocation for nonlinear systems, etc. Our numerical examples will demonstrate improved performance on nominal MPC problems, and we hypothesize that the same computational methods used here will be capable of improving performance of stochastic MPC formulations as well.

3.3 MPC Solution by Interior Point Method

We solve problem (3.1) using a primal-dual interior point method [51]. Our computational experience implementing large scale nonlinear MPC in multiple experiments has shown that interior point methods can handle nonlinearities and large sparse optimization problems with better performance and reliability than active set methods. We avoid sequential quadratic programming (SQP) since the individual iterations of an interior point method are less expensive than a full QP subproblem, and nonconvexity is more difficult to handle in a SQP framework.

First, we consolidate the control and state variables over the prediction horizon into a single optimization variable $z = [u_{0|t}^T, x_{1|t}^T, u_{1|t}^T, \ldots, x_{N-1|t}^T, u_{N-1|t}^T, x_{N|t}^T]^T$. Let $\mathbf{J}_t(z) = \sum_{k=0}^{N-1} J(x_{k+1|t}, x_{k|t}, u_{k|t}, w_{k|t})$,

$$\mathbf{g}_t(z) = \begin{bmatrix} g(x_{1|t}, x(t), u_{0|t}, w_{0|t}) \\ \vdots \\ g(x_{N|t}, x_{N-1|t}, u_{N-1|t}, w_{N-1|t}) \end{bmatrix},$$

$$\text{and } \mathbf{f}_t(z) = \begin{bmatrix} f(x_{1|t}, x(t), u_{0|t}, w_{0|t}) \\ \vdots \\ f(x_{N|t}, x_{N-1|t}, u_{N-1|t}, w_{N-1|t}) \end{bmatrix}.$$

Introducing slack variables s, problem (3.1) can be equivalently restated as

$$\begin{aligned} \min_{z,s} \quad & \mathbf{J}_t(z) \\ \text{subj. to} \quad & \mathbf{g}_t(z) + s = 0 \\ & \mathbf{f}_t(z) = 0 \\ & s \geq 0. \end{aligned} \tag{3.2}$$

Let λ be the dual variables (Lagrange multipliers) associated with the inequality constraints, and let ν be the dual variables associated with the dynamics. The Karush-Kuhn-Tucker (KKT) optimality conditions for problem (3.2) are

$$\nabla \mathbf{J}_t(z) + \nabla \mathbf{g}_t(z)\lambda + \nabla \mathbf{f}_t(z)\nu = 0 \tag{3.3a}$$
$$\mathbf{g}_t(z) + s = 0 \tag{3.3b}$$
$$\mathbf{f}_t(z) = 0 \tag{3.3c}$$
$$s \geq 0 \tag{3.3d}$$
$$\lambda \geq 0 \tag{3.3e}$$
$$\Lambda S \mathbf{1} = 0, \tag{3.3f}$$

where $\Lambda = \text{diag}(\lambda)$, $S = \text{diag}(s)$, and $\mathbf{1} = [1, \ldots, 1]^T$.

In a primal-dual interior point method, we restrict the slack and inequality dual variables to be strictly positive, and solve a sequence of approximate problems,

$$\nabla \mathbf{J}_t(z) + \nabla \mathbf{g}_t(z)\lambda + \nabla \mathbf{f}_t(z)\nu = 0 \tag{3.4a}$$
$$\mathbf{g}_t(z) + s = 0 \tag{3.4b}$$
$$\mathbf{f}_t(z) = 0 \tag{3.4c}$$
$$s > 0 \tag{3.4d}$$
$$\lambda > 0 \tag{3.4e}$$
$$\Lambda S \mathbf{1} = \mu \mathbf{1}. \tag{3.4f}$$

This approximate problem can be interpreted as adding a penalty term of the form $-\mu \mathbf{1}^T \log(s)$ to the cost function. By decreasing the barrier parameter $\mu \to 0$, solutions to (3.4) approach solutions to (3.3).

A solution to (3.4) is iteratively obtained using Newton's method. Denote the primal-dual guessed solution at iteration j by $z^{(j)}, s^{(j)}, \lambda^{(j)}, \nu^{(j)}$. Each iteration of Newton's method solves for a step direction $\Delta z^{(j)}, \Delta s^{(j)}, \Delta \lambda^{(j)}, \Delta \nu^{(j)}$ such that

$$\begin{bmatrix} \nabla^2 \mathcal{L}_t^{(j)} & 0 & \nabla \mathbf{g}_t(z^{(j)}) & \nabla \mathbf{f}_t(z^{(j)}) \\ \nabla \mathbf{g}_t(z^{(j)})^T & I & 0 & 0 \\ \nabla \mathbf{f}_t(z^{(j)})^T & 0 & 0 & 0 \\ 0 & \Lambda^{(j)} & S^{(j)} & 0 \end{bmatrix} \begin{bmatrix} \Delta z^{(j)} \\ \Delta s^{(j)} \\ \Delta \lambda^{(j)} \\ \Delta \nu^{(j)} \end{bmatrix} = r^{(j)}, \tag{3.5}$$

where $\nabla^2 \mathcal{L}_t^{(j)}$ is the Hessian of the Lagrangian function $\mathbf{J}_t(z) + \lambda^T \mathbf{g}_t(z) + \nu^T \mathbf{f}_t(z)$ evaluated at $z^{(j)}, \lambda^{(j)}, \nu^{(j)}$, and $r^{(j)}$ is a residual vector. We eliminate $\Delta s^{(j)}$ to obtain the equivalent system

$$\begin{bmatrix} \nabla^2 \mathcal{L}_t^{(j)} & \nabla \mathbf{g}_t(z^{(j)}) & \nabla \mathbf{f}_t(z^{(j)}) \\ \nabla \mathbf{g}_t(z^{(j)})^T & -(\Lambda^{(j)})^{-1} S^{(j)} & 0 \\ \nabla \mathbf{f}_t(z^{(j)})^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z^{(j)} \\ \Delta \lambda^{(j)} \\ \Delta \nu^{(j)} \end{bmatrix} = \tilde{r}^{(j)}. \tag{3.6}$$

We refer to system (3.6) as the KKT system for one interior point iteration, and the matrix on the left hand side as the KKT matrix. This KKT matrix is symmetric and indefinite, reflecting the saddle-point nature of the optimality conditions at a solution (minimum in the primal variables, maximum in the dual variables). For large scale MPC problems the Hessian of the Lagrangian and the Jacobians of the constraints and dynamics are sparse, the consequences of which we will discuss in the next section. Solving the sparse symmetric indefinite system of linear equations (3.6) at each iteration is the computational bottleneck of an interior point method.

3.4 Sparsity Considerations

There are two sources of sparsity in the KKT system for a large scale MPC problem. The first source of sparsity is block-bandedness of the Hessian and Jacobians, since the cost, dynamics, and constraint functions only depend on the control inputs at a single time step and the states at two consecutive time steps. This block-banded structure has been noted and taken advantage of in the context of MPC by several authors [20, 65, 66].

The second source of sparsity has been largely neglected in the MPC literature, but it is of critical importance to achieving acceptable optimization performance with large scale problems. This is the sparsity within the band: the problem-dependent structure of the Hessian and Jacobian within each time step. The sub-blocks of these matrices at every time step are also sparse since the cost nonlinearities, constraint functions, and system dynamic models only depend on a subset of the state and control variables. Large scale systems do not have all-to-all dense dependencies, and algorithms for control of large scale systems cannot rely on dense linear algebra if they are to have any hope of scaling well. This sparsity within the band has likely received less attention because it is entirely problem-dependent and therefore difficult to make general conclusions about. However, many sparse linear algebra algorithms are designed to be general-purpose, applying strategies that are designed for good performance across as many sparsity structures as possible.

Formulating the optimization problem in a sparse manner can also be time-consuming depending on the modeling environment and programming language(s) being used. Modeling tools designed for forward simulation rarely provide the gradients, Jacobians, or Hessians needed by optimization algorithms; special-purpose languages for optimization such as AMPL [23] are unfamiliar to many application engineers and can be difficult to use collaboratively to develop large models.

We have developed a tool called the Berkeley Library for Optimization Modeling (BLOM, see Chapter 4 for more information) so system models, constraints and cost function can be formulated in Simulink [63] using an intuitive block diagram model. BLOM automatically extracts an efficient sparse problem representation (with closed-form Jacobian and Hessian) from the block diagram model and interfaces directly with compiled optimization solvers

such as Ipopt [64]. The problem representation used by BLOM is

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{r} c_i \left(\prod_{j=1}^{n} x_j^{P_{ij}} \right) \\ \text{subj. to} \quad & \sum_{i=1}^{r} K_{li} \left(\prod_{j=1}^{n} x_j^{P_{ij}} \right) = 0, \ \forall \ l \in \{1, \ldots, m\} \\ & \underline{x} \leq x \leq \overline{x}, \end{aligned} \tag{3.7}$$

where $x \in \mathbb{R}^n$ is an optimization vector including all variables over the entire prediction horizon, $c \in \mathbb{R}^r$ is a cost coefficient vector, $P \in \mathbb{R}^{r \times n}$ is a sparse matrix of exponents, $K \in \mathbb{R}^{m \times r}$ is a sparse matrix of constraint coefficients, and $-\infty \leq \underline{x} \leq \overline{x} \leq \infty$ are lower and upper bound vectors with elements equal to $\pm\infty$ for unbounded variables.

With this formulation, any optimization problem with a polynomial-like nonlinearity structure (also including non-integer exponents, and rational functions via implicit equalities) can be encoded in two sparse matrices, a cost vector and two bound vectors. The formulation can also be extended to include transcendental functions by reserving a few very large exponent values to represent special function exception codes.

3.5 Sparse Linear Algebra

Computational algorithms for solving sparse linear systems of equations fall into two categories: direct methods and iterative methods. We present an overview of these methods below.

Direct methods

Direct methods are based on matrix factorizations. In the case of a symmetric indefinite matrix A, the most common approach is the LDL^T factorization. The objective of this method is to find a permutation matrix Π, a lower triangular matrix L with unit diagonals, and a block-diagonal matrix D with 1×1 and 2×2 diagonal blocks, such that

$$A = \Pi L D L^T \Pi^T. \tag{3.8}$$

The permutation matrix Π is generally not unique. For sparse A, the permutation matrix serves two purposes: pivoting for numerical stability (to avoid dividing by zero or very small values), and reordering to reduce fill-in of the triangular factor L. The sparsity pattern of the factor L depends on the sparsity pattern of $\Pi^T A \Pi$ and the locations of any 2×2 blocks in D. The objective of a sparse factorization algorithm is to choose a permutation Π such that the factor L is as sparse as possible while maintaining numerical stability. The amount of memory and number of floating point operations (flops) required to carry out the factorization are both smaller if L has fewer nonzero entries.

It is known that finding the optimal permutation Π is NP-complete [19], but good heuristic strategies are available. Different implementations of sparse direct symmetric indefinite factorizations vary in the details of the reordering strategies and many other aspects, see [25] for an in-depth discussion.

Direct factorization algorithms are typically divided into 3 phases. First, an "analyze" phase (also known as symbolic factorization) examines the sparsity pattern of A to make an initial guess for Π, predict the amount of fill-in that will occur in L, and preallocate memory and data structures. In the context of MPC, the sparsity pattern of the KKT matrix is fixed between interior point iterations and control time steps, so this step can be performed offline and its results can be reused repeatedly.

Next, the numeric factorization is performed to calculate the values of L and D. If a zero or small value appears on the diagonal during the factorization, pivoting will be required to maintain numerical stability. This changes the permutation relative to the prediction from the analyze phase, and may lead to increased fill-in of L, possibly requiring dynamic reallocation of memory to modify the predicted sparsity structure of L and store the additional nonzero entries. The numerical values of the KKT matrix can change between every interior point iteration, so this phase needs to be performed online.

Finally, once the factors are completely calculated, they are used to obtain the solution vector. If we are solving $Ax = b$ and we have computed a factorization of the form (3.8), we solve for x as follows.

$$\begin{aligned} \Pi L D L^T \Pi^T x &= b \\ Ly &= \Pi^T b \\ L^T z &= D^{-1} y \\ x &= \Pi z \end{aligned}$$

The triangular systems $Ly = \Pi^T b$ and $L^T z = D^{-1} y$ are simple to solve by forward and backward substitution, and D is simple to invert since it is block-diagonal with very small blocks. This substitution phase is less computationally demanding than calculating the factorization; in the dense $n \times n$ case the factorization phase is $O(n^3)$ flops and the substitution phase is $O(n^2)$ flops. For the sparse case the flop counts are functions of the number of nonzero entries and the sparsity pattern of L, but factorization is still a higher-order term than back-solve.

Iterative methods

An alternative class of algorithms for solving a sparse system of linear equations $Ax = b$ are iterative methods, also known as Krylov subspace methods. These methods calculate the Krylov subspace by repeated sparse matrix-vector multiplication, forming the sequence

$$\{x, Ax, A^2x, \ldots, A^k x\}.$$

After each multiplication by A, the "best" solution vector in the Krylov subspace is selected based on some criterion. Commonly used criteria include conjugate gradients, minimum residuals, etc.

Iterative linear solvers are attractive because the parallelization and communication patterns of sparse matrix-vector multiplication are much simpler to implement than a direct factorization. There is no pivoting or fill-in to deal with. However the solution vector from an iterative method has a much greater error than the solution from a direct method. The number of iterations k required to obtain a solution with a desired accuracy depends on the condition number of the matrix A. Practical performance of iterative methods therefore relies heavily on preconditioners.

A preconditioner is a matrix M that is simple to invert and results in a low condition number for the matrix $M^{-1}A$. Instead of solving $Ax = b$, we apply the inverse of the precondioner to both sides and solve $M^{-1}Ax = M^{-1}b$.

There has been recent work to adapt interior point algorithms to use iterative linear solvers on the KKT system [17]. This is a topic of continuing research; the experimental implementations are considerably less robust than established algorithms based on direct linear solvers. Convergence of iterative methods can be a challenge, particularly for interior point optimization methods since the logarithmic barrier terms lead to ill-conditioning of the KKT matrix when the solution is close to a constraint.

We believe iterative linear algebra methods have significant potential for optimization and MPC, especially for implementation on advanced parallel architectures such as GPUs and message passing clusters, and as mainstream computers move in the direction of increasing parallelism.

3.6 Numerical Results

For numerical evaluation we use 2 nonlinear MPC problems from HVAC models created using BLOM. The full model has 24826 optimization variables, 23146 equality constraints and 5940 inequality constraints. A reduced-order model of a subset of the same system has 7547 optimization variables, 7147 equality constraints and 1480 inequality constraints. Both models have prediction horizon lengths of 20 time steps. The sparsity patterns of the KKT matrices are shown in Figures 3.1 and 3.2.

Note that the third block-banded section of the Jacobian in these examples contains the dynamic coupling between adjacent time steps. The dynamic equality constraints are grouped after the nonlinear function equalities here. The full model is just over 3 times larger than the reduced model and the Hessian structures are somewhat different, but superficially these problems appear quite similar due to the block-banded sparsity over time.

We solve these two optimization problems on an Intel Xeon E5410 with 4 physical cores (8 virtual cores due to hyperthreading) running Red Hat Enterprise Linux 5. We use Ipopt [64] linked to the optimized Intel Math Kernel Library (MKL) version 10.3 for Basic Linear Algebra Subprograms (BLAS) implementation.

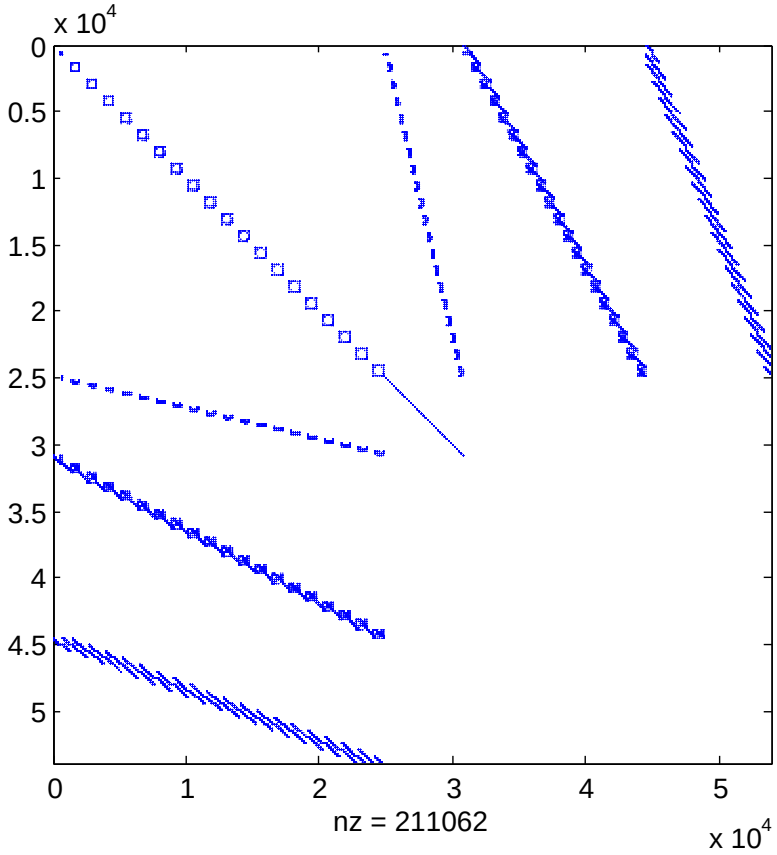


Figure 3.1: Full model KKT matrix sparsity pattern

Ipopt can use the following sparse direct linear solvers:

1. MUMPS [4] has a distributed-memory parallel version available, but this would require using an unsupported message-passing branch of Ipopt. The version of MUMPS supported by conventional Ipopt is serial.

2. HSL [30] MA27 is a serial algorithm.

3. HSL [30] MA57 [21] is a serial algorithm, the successor to MA27. An additional feature of MA57 is that it performs aggregation of neighboring sparse elements into small dense blocks, to improve computational efficiency using BLAS operations.

4. HSL [30] MA77 is a serial out-of-core algorithm, meaning it is designed to save intermediate factorization results to disk during the factorization process. This is only beneficial for extremely large problems that are too large to hold in memory, which is not the case for KKT matrices from optimization problems that can be solved in real-time.

5. HSL [30] MA86 is a parallel multithreaded algorithm using OpenMP. Due to finite precision arithmetic and unpredictable order of operations in this algorithm, results from MA86 can actually vary slightly between different invocations, although all of the results will be equally numerically accurate. Compounded over many iterations of Ipopt, the overall optimization may take a variable number of iterations.

6. HSL [30] MA97 is a parallel multithreaded algorithm using OpenMP that, unlike MA86, is specifically designed to obtain binary-identical results using any number of threads. MA97 is not yet officially supported by Ipopt, but a prototype interface is available from the HSL team, which we are using here.

7. PARDISO [34, 60, 59] is a parallel multithreaded algorithm using OpenMP.

8. WSMP [27, 28] is a parallel multithreaded algorithm using POSIX threads.

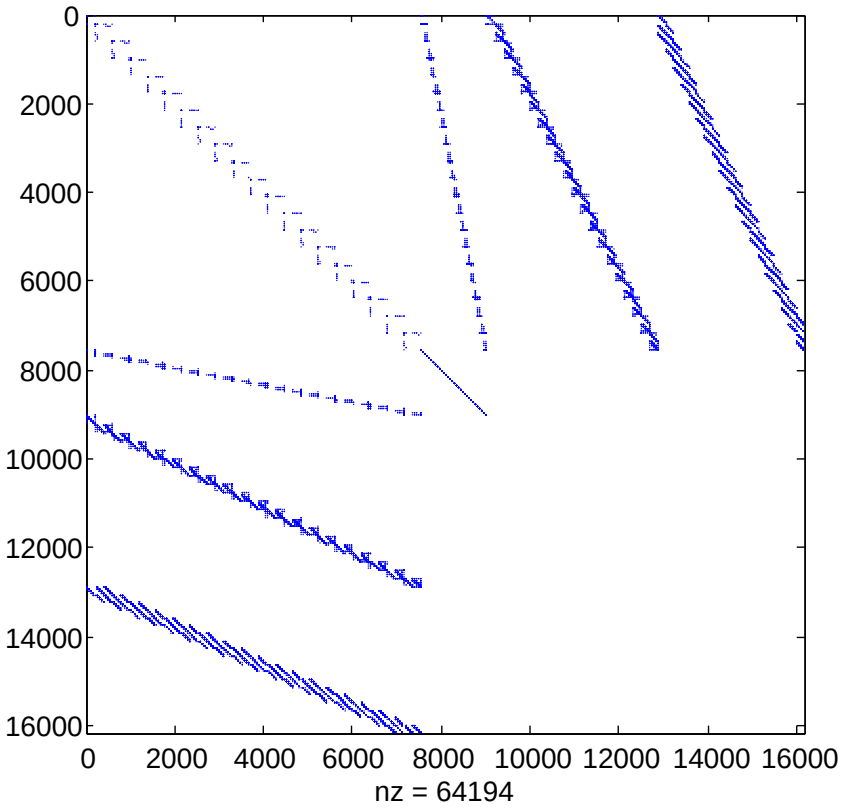


Figure 3.2: Reduced model KKT matrix sparsity pattern

Refer to [41] for a more complete survey of available sparse direct linear solvers. The non-convex constrained nonlinear programming algorithm in Ipopt [64] requires inertia information (the number of positive, negative, and zero eigenvalues) from the KKT linear systems

that it factorizes at each iteration in order to guarantee robust convergence properties. Only sparse direct solvers that use a symmetric indefinite Bunch-Kaufman [13] factorization are able to calculate inertia information efficiently, so the number of solvers that are capable of being used in Ipopt is limited.

For the first 4 serial algorithms, the only potential parallelism is in BLAS operations, assuming a multithreaded BLAS implementation such as Intel MKL is being used. We experimented with other optimized BLAS implementations, but found the performance less consistent and the multithreading behavior more difficult to control with available libraries other than MKL. We also experimented with nested parallelism, using multithreaded BLAS operations with the multithreaded sparse solvers, but this resulted in worse performance than when we allocated all parallel threads to the sparse solver.

The results for the full and reduced problems, varying the linear solver and number of threads (BLAS threads for serial algorithms, solver threads for parallel algorithms) are shown in Figure 3.3. All Ipopt algorithm and linear solver options were left at their default values, with the exception of the choice of linear solver, and `wsmp_num_threads` to control multithreading with WSMP. We show the Ipopt wall time not counting the function evaluations. The function evaluations are performed by interface code that calculates the Jacobian and Hessian values from the BLOM problem formulation (3.7). This user code took less than 5% of the total execution time on average for the full problem, and around 11% of the total time on average for the reduced problem. The performance of this code can be further improved, but that is not the focus of this study.

We can see from the results that multithreading by BLAS operations alone provides only a negligible performance improvement on the full problem with MUMPS or MA57, and in fact leads to worse performance on the reduced problem or with MA77. The reductions to dense sub-block operations performed by these sparse solvers are evidently too small for multithreaded BLAS to help much. MA27 does not use BLAS operations, so there is no noticeable change in its performance with different numbers of threads.

The sparse linear solvers that are designed as parallel algorithms do show meaningful parallel speedup on the full problem, and on the reduced problem as well in the case of MA86 and WSMP. The choice of linear solver can make up to a factor of 3 difference in serial performance, and WSMP demonstrates a parallel speedup of roughly another factor of 3, using 8 threads on the large problem. We can clearly see less benefit beyond 4 threads; this is likely due to the hyperthreading architecture of this processor. These sparse factorization algorithms are computationally demanding enough that the 4 virtual cores are not as capable as the original 4 physical cores.

3.7 Conclusions

In this work we demonstrated a parallel method for the solution of constrained nonlinear model predictive control, using sparse direct factorization algorithms that are available to use with the interior point optimization solver Ipopt. Careful consideration must be paid to

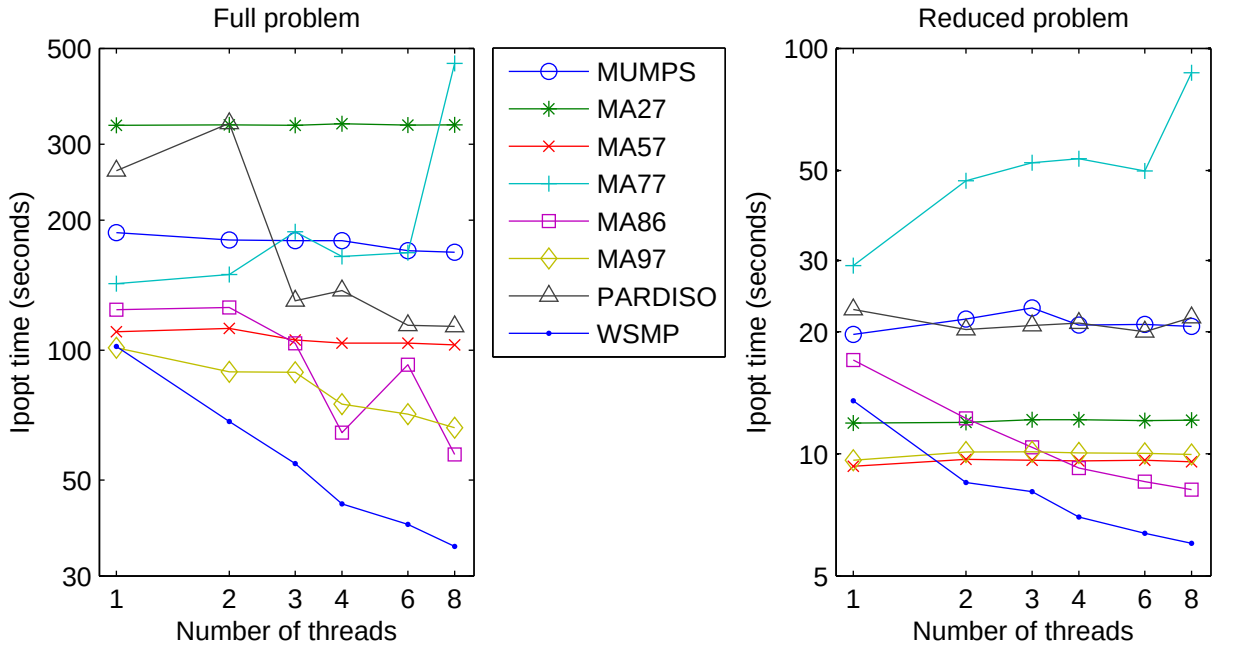


Figure 3.3: Ipopt wall time (not counting function evaluations) on full and reduced problem, varying linear solver and number of threads

sparse formulation of large scale problems, for which we have developed a modeling tool and efficient structured sparse representation for nonlinear programs.

Our numerical results demonstrate useful parallel speedup of a large scale MPC problem on a typical multicore workstation, particularly with the linear solver WSMP. A worthwhile investigation area will be to examine what specific aspects of the WSMP algorithm result in its good performance on our problems, to see if we can further exploit tailored factorization strategies. It remains to be seen how well parallel sparse direct methods will scale to an increasing number of processors. We could look into using the message passing Ipopt branch and conducting numerical experiments with the linear solvers that have distributed-memory versions available, namely MUMPS, PARDISO, and WSMP, on larger clusters.

Our results only made use of direct linear solvers, but we anticipate iterative linear solvers will play an increasing role in optimization and control algorithms. This may require further development of optimization approaches that do not suffer from the ill-conditioning of interior point methods, [67] is one potential example.

Chapter 4

Optimization Modeling Tools

We present the Berkeley Library for Optimization Modeling (BLOM), an open-source tool for optimization-based modeling and control formulation implemented in Simulink [63]. The underlying structure for BLOM is a novel way of representing linear and nonlinear mathematical functions that allows for easy computation of closed form gradients, Jacobians and Hessians. This formulation provides an efficient problem representation for optimization-based modeling and is scalable to large optimization problems. With BLOM, an optimization-based controller for a dynamic system can be developed and exported from the same model that is used in forward simulation.

BLOM is capable of solving several types of optimization problems, including static optimization problems and optimization problem with dynamics. Its intended use is for nonlinear model predictive control. We present results where BLOM is able to handle problems with tens of thousands of variables.

This chapter is based on work that has been previously published in [38].

4.1 Introduction

In order to design an optimization-based controller, a researcher or engineer can choose a tool from two main groups: simulation oriented tools or optimization oriented tools. Model-based optimization is typically difficult to do with existing tools. Requirements not commonly met by modeling tools designed for conventional forward simulation include: representation of constraints and a cost function, distinguishing unknown input signals that can be freely chosen from signals with known input values, and efficient calculation of derivatives for gradient, Jacobian, and Hessian information used by optimization algorithms. There are several modeling languages that use block diagram or objected oriented approaches to describe systems. Examples include: Simulink [63], Modelica [24], and ASCEND [55].

Many languages and tools exist for optimization modeling [33, 15], but a common weakness of existing methods is that they can be cumbersome to use for forward simulation, and model validation and verification is more difficult than in a simulation environment. The

language characteristics of optimization-oriented tools can be unfamiliar and difficult to use for engineers who are familiar with system modeling but are not optimization experts, which limits collaborative model development opportunities on large scale systems. A large model that is developed using a conventional simulation and technical computing environment like Matlab or Simulink can be difficult to translate into an optimization formulation in languages like AMPL [23], GAMS [58], or AIMMS [10].

Recently the Optimica [2, 3] extension for the Modelica language was introduced. This extension facilitates the conversion of a dynamic model into an optimization problem.

In this paper, we propose a tool called the Berkeley Library for Optimization Modeling (BLOM) which bridges the gap between simulation oriented tools and optimization oriented tools. BLOM is based on a new formulation for representing linear and nonlinear mathematical functions that aims to address some of the limitations of simulation-oriented tools. This formulation allows for direct computation of closed form gradients, Jacobians and Hessians. The initial model formulation interface is based on Simulink, and BLOM provides a set of Matlab functions which transform a Simulink model into an optimization problem using our representation format. This problem representation is then used in a compiled interface to an optimization solver such as Ipopt [64]. BLOM can be expanded in the future to use the same internal problem representation with other model formulation interfaces and optimization solvers.

The primary intended use of BLOM is for nonlinear model predictive control (MPC) problems, but static optimization problems with no system dynamics or time horizon can also be represented, formulated, and solved. BLOM currently requires all functions to be in $\mathbb{C}^2$ to allow for the computation of the Hessian. Another limitation of BLOM is that it does not allow for the use of all Simulink blocks, but many commonly-used blocks are supported.

In Section 4.2, we describe the proposed mathematical formulation used in BLOM. We then describe the Simulink interface implementation in Section 4.3. The way that BLOM exports models for optimization problems is further elaborated in Section 4.4. In 4.5 we present a few problems that BLOM is capable of handling. We conclude in Section 4.6.

4.2 Internal mathematical representation

We consider optimization problems of the following form:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{4.1a}$$

$$\text{s.t.} \quad g(x) = 0 \tag{4.1b}$$

$$x_l \leq x \leq x_u \tag{4.1c}$$

$$x_l \in \{\mathbb{R} \cup -\infty\}^n, x_u \in \{\mathbb{R} \cup \infty\}^n \tag{4.1d}$$

where x_l and x_u are the upper and lower bound vectors for the optimization variables x. Elements of x that are unbounded above or below have the corresponding elements of x_u or

x_l equal to $+\infty$ or $-\infty$, respectively.

Consider the function, $\mathbf{y} = f(\mathbf{u})$ where $y \in \mathbb{R}^m, u \in \mathbb{R}^n$ and $f \in \mathbb{C}^2$, with the scalar input elements denoted by u_j and the scalar output elements denoted by y_j. We propose that the input-output relationship of this function can be represented in the following way:

$$0 = \sum_{k=1}^{r} K_{ik} \left(\prod_{j=1}^{n} v(u_j, P_{kj}) \right) \left(\prod_{j=n+1}^{n+m} v(y_{j-n}, P_{kj}) \right), \quad \forall\, i \in \{1, \ldots, m\}. \tag{4.2}$$

The parameterized function v is defined as

$$v(x, p) = \begin{cases} x^p & \text{if } p \text{ is not an exception code} \\ \exp(x) & \text{if } p \text{ is the code for exp} \\ \log(x) & \text{if } p \text{ is the code for log} \\ \text{etc.} & \end{cases} \tag{4.3}$$

We define a list of exception codes to represent transcendental functions. This list can be extended to include any differentiable single-operand function.

The matrices $K \in \mathbb{R}^{m \times r}$ and $P \in \mathbb{R}^{r \times (n+m)}$ contain, respectively, the coefficient and exponent data of a multivariable polynomial-like function. The number of monomial terms in this function is given by r. Typically the matrix P will be sparse, and K may be sparse as well.

This representation is versatile enough to represent rational functions as well. If the desired input-output relationship is $y = f(u)/g(u)$ where $f(u)$ and $g(u)$ are polynomial-like functions and $g(u) \neq 0$, this can be captured by encoding the constraint $0 = f(u) - y \cdot g(u)$ in the P and K matrices.

In addition, the formulation facilitates the calculation of closed-form gradients, Jacobians and Hessians for solvers that make use of derivative information, such as Ipopt.

In the optimization formulation, there is no need to distinguish between input and output variables (u and y in (4.2)). Therefore, (4.2) for a single row i of K can be restated as

$$f_i(\mathbf{x}) = \sum_{k=1}^{r} K_{ik} \left(\prod_{j=1}^{n+m} v(x_j, P_{kj}) \right), \tag{4.4}$$

where $f_i(\mathbf{x})$ is the constraint function of vector $\mathbf{x}$. The derivative of a $f_i(\mathbf{x})$ with respect to a variable x_d is

$$\frac{\partial f_i(\mathbf{x})}{\partial x_d} = \sum_{k=1}^{r} K_{ik} \frac{\partial v(x_d, P_{kd})}{\partial x_d} \left(\prod_{j \in \mathcal{J}} v(x_j, P_{kj}) \right), \tag{4.5}$$

where $\mathcal{J}$ is the set of variable indexes excluding d, $\mathcal{J} = \{1, \ldots, d-1, d+1, \ldots, n+m\}$. The second derivative can be computed in a similar way.

Only the non zero elements of the Jacobian and Hessian matrices are exported to a solver, because sparsity structure is immediately available from the K and P matrices. This sparse structure results in efficient performance for very large problems.

4.3 Simulink interface implementation

BLOM consists of three main parts. First, there is the Simulink front end, where a dynamic model is represented using built-in Simulink blocks and our BLOM library blocks. Second, a set of Matlab functions is used to convert a Simulink model into the internal mathematical representation described in Section 4.2. Lastly, this problem representation is used by an interface to an optimization solver such as Ipopt. In the following Sections, we describe how we implement different parts of an optimization problem in Simulink.

Mathematical representation applied to Simulink blocks

Most commonly used Simulink blocks can be represented using the P and K matrices described in Section 4.2. For example, an addition block with scalar inputs x_1, x_2, x_3 and output y can be represented internally in BLOM as

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

$$K = \begin{pmatrix} 1 & 1 & 1 & -1 \end{pmatrix} \tag{4.7}$$

where the columns of P represent the inputs x_1, x_2, x_3 and output y, respectively. Many mathematical blocks in Simulink can be represented by appropriate P and K matrices. Many commonly used mathematical blocks in Simulink are currently supported in BLOM. The P and K matrix formulation allows for the expansion of BLOM's functionality of Simulink blocks for many other Simulink blocks that have yet to be supported.

We formulate an optimization problem by introducing variables for the output signals of every mathematical block in the Simulink model, and equality constraints to enforce the input-output relationship of each block. This implicit approach increases the size of the optimization problem compared to an expression-tree approach, but the system model sparsity structure is preserved in the optimization formulation. An expression-tree approach would lead to an optimization problem with fewer variables and equality constraints, but the constraints and cost function would be denser and the calculation of gradient, Jacobian, and Hessian information would be more complicated due to nested nonlinearities.

Inequality constraints

Inequality constraints are marked in a Simulink model using the *Bound* block from the BLOM Library. The bound block has two user-defined parameters, *Upper Bound* and *Lower Bound*, with default values `inf` and `-inf` respectively. There are also three check boxes which specify whether the bound is enforced at the first time step, intermediate time steps, and/or the final time step of a dynamic problem.

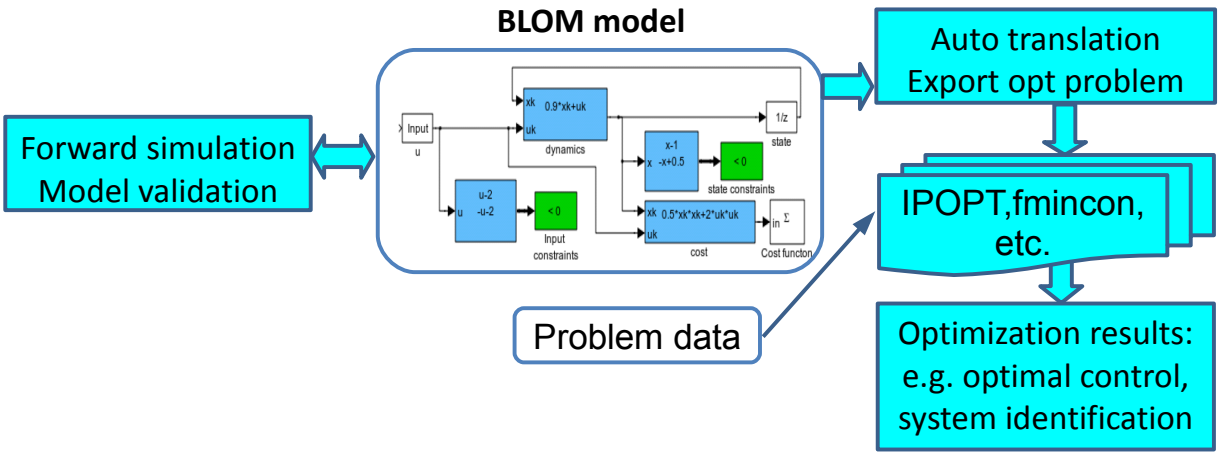


Figure 4.1: Work flow with BLOM. First a model is created and validated using the BLOM library. Then, it is converted to an optimization problem and exported to one of the supported solvers.

Cost function

The cost function for optimization is represented in Simulink by a BLOM Library block called *Cost* with one input signal. This block does not play any role in forward simulation mode, but the input signal is treated as a cost function in an optimization problem. If the input signal is a vector, the current version sums the input elements. In the future, an option setting will be available to determine which norm to take over the input elements (1, 2, or ∞).

For models with system dynamics and time horizons longer than 1 step, the cost function is a discrete accumulator of this signal. This functionality can be further expanded to continuous integrator, the peak value of the input signal over the time horizon, or norm (1, 2, or ∞) over time of the input signal. Like the Bound block, an option allows the user to choose whether to include initial time steps, intermediate time steps, and/or final time steps in this summation or integral.

If there are multiple cost function blocks within the same BLOM model, the optimization cost function is treated as the sum of the values over all cost function blocks.

Control and external inputs

There are two classes of input signals to a BLOM model: unknown signals that the optimization algorithm is free to choose in order to minimize the cost function subject to constraints, and time-varying signals with a known trajectory of values over a future horizon. We will refer to the former as control inputs, and the latter as external inputs. In a MPC problem, external inputs correspond to predicted future model parameters or disturbances.

Both classes of inputs are parameterized as uniformly-sampled time series. Time variation for each input signal within one time step is important for discretization accuracy. It can be either piecewise linear and continuous (first order hold (FOH)), or piecewise constant with discontinuities at the sample times (zero order hold (ZOH)).

We assume all input signals have the same sample rate, with the exception of an optional simple implementation of move blocking on control inputs [14]. In order to reduce the number of optimization variables, the user can specify that certain control inputs should have constant values over some integer number of time steps.

In forward simulation, control input and external input blocks take input signals from non-BLOM sources in Simulink, such as Constant or From Workspace blocks. Optimization formulation only requires the dimensions of control and external input signals. The values used for external input trajectories over the optimization horizon are communicated between the Matlab workspace and the optimization solver, and do not need to be the same values as used in the Simulink block diagram.

Discrete-time and continuous-time states

BLOM can support both discrete-time and continuous-time models. Discrete states are implemented using the $1/z$ Unit Delay block in Simulink, and continuous states are implemented using the $1/s$ Integrator block in Simulink. If a BLOM model contains multiple Unit Delay blocks, they are all assumed to have the same sample rate. For hybrid discrete/continuous-time models, the inter-sample behavior of a discrete state block will be set to either zero order hold or first order hold as an optional field.

Continuous-time models are converted into finite dimensional optimization problems using a fixed-timestep discretization method. If there are both discrete and continuous states in the same model, the discretization step length for the continuous states must be equal to the sample time of the discrete states. The discretization method and the timestep length are specified by the user. The discretization method can be any general Runge-Kutta method, either explicit or implicit, of arbitrary order, specified by the user in the form of a Butcher tableau. Runge-Kutta methods require the introduction of additional variables for intermediate function calculations at minor time steps.

4.4 Automated generation of an efficient optimization problem

As shown in Figure 4.1, after a model is created in Simulink and validated in BLOM, that model can be converted to an optimization problem and exported to a solver. This Section details this process.

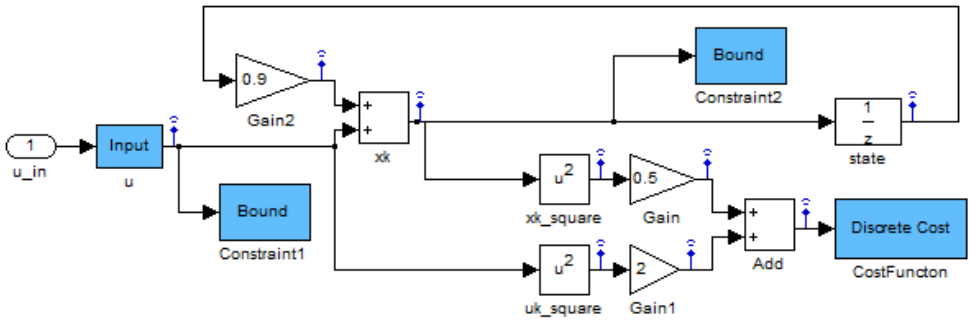


Figure 4.2: Simple example of dynamic system with state and input constraints and a quadratic cost function.

Model structure extraction

To extract the model, BLOM first locates the Cost and Bound blocks. From there, BLOM does a standard breadth first search that begins at these blocks and ends at the control and external input blocks. BLOM considers each scalar variable of every block found in the breadth first search as an optimization variable to consider in the formulation. Thus, for each relevant block, BLOM will generate the relevant P and K matrices relating the inputs and outputs of that block.

When the breadth first search is complete, BLOM then has a set of variables for the model in one time step. These variables are then replicated for each time step, with state variables being related between adjacent time steps to account for the propagation of model dynamics over time. This causes the P and K matrices to scale linearly across time steps and retain a sparse structure that allows for efficient computation.

Solver initialization and model validation

BLOM uses Simulink's forward simulation capabilities to obtain equality-feasible starting guesses for an optimization solver. Ensuring inequality feasibility may be intractable in forward simulation mode, especially for non-trivial models. Therefore, this step is not mandatory. The forward simulation is also used for model validation.

Once the forward simulation is done in Simulink for a feasible starting guess, the extracted model can be considered a standalone process and does not require the use of Simulink anymore. The remainder of the problem is done in Matlab and Ipopt. In addition, if the model does not change, the model only needs extracted once. Once extracted, it is possible to change any external values and run the optimization problem as a standalone process.

Table 4.1: BLOM with Ipopt performance on a large HVAC problem for various prediction horizon lengths

Prediction horizon length	5	10	15	20	25	30
Number of variables in solver	7147	12176	18553	24846	31223	37516
Number of constraints	7329	15151	22974	30796	38619	46441
Non-zeros in Jacobian and Hessian	24057	52208	80442	108593	136827	164978
Number of solver iterations	46	230	89	82	65	68
Total solution time [sec]	2.5	8.6	26.2	48	42	97
Time spent in BLOM callbacks	11%	11%	9%	6.8%	8%	5%

Interface to optimization solvers

Once a model is created in Simulink and the optimization cost, constraints, and inputs are specified using BLOM Library blocks, the problem can be exported to an optimization solver. The interface requirements will vary depending on the solver being used, but generally a solver requires a set of user-defined functions to evaluate the cost, constraints, gradients, Jacobian, and Hessian for the optimization problem.

The BLOM problem representation allows these functions to be written in a general-purpose and straightforward way. Specialized code generation for a particular problem can be performed, but is not mandatory and in our experience code generation does not scale well to very large problems. We have implemented a compiled C++ interface to the solver Ipopt [64], which is a state of the art open source interior point solver for sparse nonlinear programming. This interface reads the P and K matrices and bound vectors from the BLOM representation for any general problem and evaluates the nonlinear functions that Ipopt requires at each iteration of its algorithm. These function evaluations contribute only a small fraction of the time required to solve an optimization problem.

4.5 Examples

Simple MPC

Consider the discrete dynamic system $x_{k+1} = 0.9x_k + u_k$ with bounded input $|u| \leq 2$ and state constraints $0.5 \leq x \leq 1$. We want to create a constrained finite time optimal control (CFTOC) problem with cost $J = \sum_{k=0}^{N-1} 0.5x_{k+1}^2 + 2u_k^2$.

Figure 4.2 shows a BLOM model that is used to create the CFTOC problem. The system dynamics and cost function are implemented using built-in Simulink math function blocks. Two constraint blocks (called "Bound" in Figure 4.2) are enforcing maximum and minimum inequality conditions. This system has a single input u that is marked by an input block to let BLOM know that it is a free optimization variable. In order to convert this model to an optimization problem, the user specifies the prediction horizon (N) and the following

CFTOC problem is created

$$\begin{aligned} &\min_{u_k,x_k} \sum_{k=0}^{N-1} 0.5x_{k+1}^2 + 2u_k^2 \\ &s.t. : -2 \leq u_k \leq 2,\ 0.5 \leq x_k \leq 1, \\ &\quad x_{k+1} = 0.9x_k + u_k,\ x_0 = x(0) \\ &\quad k = 0, \ldots, N. \end{aligned} \tag{4.8}$$

Large scale HVAC example

BLOM is used for nonlinear MPC design of a large HVAC system [6]. The system consists of 42 thermal zones and the system dynamics are modeled with 430 state variables that represent thermal masses of elements in a building. In addition the model includes an air handling unit (AHU) model, fan model and 41 variable air volume (VAV) box models with one reheating coil each. The thermodynamic model is bilinear and additional nonlinear terms exist in the model. This system has 85 control variables that need to be determined at each time step.

Table 4.1 presents performance of the BLOM library with Ipopt solver on this problem. We present the execution time of problem preparation and solution for various problem sizes. The table shows that even for very large problems with more than 20000 variables and constraints, the library achieves good performance and Ipopt converges quickly to a KKT point of the CFTOC problem.

Future development

Although the BLOM library is already used in large scale industry projects, we plan to further improve and expand on its functionality for a greater scope of problems. BLOM can be developed to include support of advanced MPC techniques, such as stochastic MPC. Support of stochastic MPC will require properly handling stochastic external variables such as weather and occupancy load predictions. These stochastic variables need to be propagated properly and must be done in a way clear to the user.

There are a variety of other control problems that BLOM can be developed for, such as for use with integer problems. We believe that the core foundation of BLOM is robust and thus allows for future expansion.

4.6 Conclusions

We proposed a new formulation for linear and nonlinear functions which allows efficient computation of closed form gradients, Jacobians, and Hessians often required by optimization solvers. The P and K sparse matrices described in Section 4.2 are portable and scalable. This formulation has proven useful when creating a Simulink library for optimization modeling.

BLOM is a good bridge between simulation modeling and optimization tools. It uniquely uses forward simulation for model validation. BLOM has yielded successful results for large scale problems with tens of thousands of variables and due to its integration with Simulink, can be easy to develop with. The ease of development helps engineers and researchers to iterate models quickly and be able to see how their system performs under different sets of conditions.

Although BLOM currently interfaces with Simulink, its underlying structure is to implement the mathematical formulation we've described given some model. Thus, it is possible to expand BLOM's functionality to include other model based designs or even have a text based structure. Interfaces to optimization solvers such as Ipopt can be reused between multiple modeling front-ends.

The most updated version of BLOM is currently available for download at `http://mpclab.net/Trac/wiki`. While BLOM was developed and successfully used for several experimental projects, it is no longer being actively maintained. Since BLOM was initially created, newer alternatives such as JuMP.jl [22], [45] based on the open source Julia language [9] have implemented nonlinear optimization modeling functionality while achieving good performance scalability on large problems. JuMP.jl solves similar problems as BLOM but in a more general way and without relying on the proprietary Matlab or Simulink environments for model creation.

Chapter 5

Specialized Optimization Algorithms for Linear MPC

In this chapter we discuss tailored methods for solving MPC optimization problems that have linear system dynamics, convex quadratic cost functions, and polyhedral constraints. This special case of the constrained finite time optimal control problem (1.2) has the form

$$\begin{aligned}
\min_{\mathbf{U},\mathbf{X}} \ & \frac{1}{2}\sum_{k=0}^{N-1}(x_{k+1}^T Q_{k+1} x_{k+1} + u_k^T R_k u_k) && \text{(5.1a)}\\
& \text{subj. to, } \ \forall\, k \in \{0,\dots,N-1\}, \\
& x_{k+1} = A_k x_k + B_k u_k + w_k && \text{(5.1b)}\\
& F_k x_k + G_k u_k \le f_k && \text{(5.1c)}\\
& F_N x_N \le f_N && \text{(5.1d)}\\
& x_0 = x(t). && \text{(5.1e)}
\end{aligned}$$

With nominal disturbance predictions w_k, this is a quadratic program (QP).

5.1 Condensed Versus Sparse MPC Formulation

Problem (5.1) can be expressed in several equivalent formulations. The propagation of linear state dynamics (5.1b) allows for simple substitution to define future state values in terms of the initial conditions x_0, inputs u_k and disturbances w_k. In the time invariant case where $A_{k+1} = A_k = A$ and $B_{k+1} = B_k = B$, the substitution has the form

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} I & 0 & \cdots & 0 \\ A & I & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1} & \cdots & A & I \end{bmatrix} \begin{bmatrix} Ax_0 + w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix}. \tag{5.2}$$

Using (5.2) we can state problem (5.1) as a minimization only over $\mathbf{U}$ with only inequality constraints. The total number of variables in this case is reduced from $(n+m)N$ to mN, however the resulting QP will have completely dense cost and constraint matrices. The cost of solving a general dense linear system of equations scales with the cube of the dimension when using direct factorization-based methods, if there is no special structure to exploit. The matrices in (5.2) have a block Toeplitz structure, but that structure would be difficult to recover from the final combined QP cost and constraint matrices.

Longer prediction horizons N allow for predictive control actions to be made further in advance of expected disturbances, and shorter time steps Δt results in lower discretization error and a higher frequency controller. So it would be preferable to achieve an optimization algorithm that scales linearly in the length of the prediction horizon N. Leaving the states x_k as optimization variables with implicit equality constraints enforcing the dynamics over time results in a much more structured problem from a linear algebra standpoint. The dynamic propagation over time couples states and inputs only at neighboring time steps, so the equality constraints have a sparse structure, specifically block banded over time.

Interleaving the state and input vectors for each time step illustrates that block banded structure as follows.

$$\begin{bmatrix} B & -I & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & A & B & -I & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & A & & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & A & B & -I \end{bmatrix} \begin{bmatrix} u_0 \\ x_1 \\ u_1 \\ x_2 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} = - \begin{bmatrix} Ax_0 + w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{N-1} \end{bmatrix} \tag{5.3}$$

As long as the block structure is preserved, solving (5.3) has a cost that scales linearly with the prediction horizon length N, and quadratically in the bandwidth $(2n+m)$.

Expressing the equality constrained QP from (5.1) in a standard form,

$$\begin{aligned} \min_{\mathbf{z}} \ & \frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} \\ \text{s.t.} \ & \mathbf{J}\mathbf{z} = \mathbf{r}(t) \\ & \mathbf{K}\mathbf{z} \leq \mathbf{c}. \end{aligned} \tag{5.4}$$

If the cost function and constraints are also time invariant, except possibly at the last time

step N, then $\mathbf{H}$ and $\mathbf{J}$ have the following structure.

$$\text{Let } s_k = \begin{cases} Fx_k + Gu_k & 0 \le k \le N-1 \\ F_N x_N & k = N \end{cases}$$

$$\mathbf{z} = [u_0; s_0; x_1; \ldots; u_{N-1}; s_{N-1}; x_N; s_N]$$

$$\mathbf{H} = \text{diag}\left(R,\ 0,\ I_{N-1} \otimes \begin{bmatrix} Q & & \\ & R & \\ & & 0 \end{bmatrix},\ Q_N,\ 0\right)$$

$$\mathbf{J} = \text{diag}\left(\begin{bmatrix} G & -I \\ B & 0 \end{bmatrix}, I_{N-1} \otimes \begin{bmatrix} F & G & -I \\ A & B & 0 \end{bmatrix}, \begin{bmatrix} F_N & -I \end{bmatrix}\right) +$$
$$\text{diag}\left(\begin{bmatrix} 0 & 0 \end{bmatrix}, I_{N-1} \otimes \begin{bmatrix} -I & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}\right),$$

$$\mathbf{r}(t) = -[Fx(t);\ Ax(t) + w_0;\ 0;\ w_1;\ \ldots;\ 0;\ w_{N-1};\ 0],$$

$$\mathbf{K} = \text{diag}\left(\begin{bmatrix} 0 & I \end{bmatrix},\ I_{N-1} \otimes \begin{bmatrix} 0 & 0 & I \end{bmatrix},\ \begin{bmatrix} 0 & I \end{bmatrix}\right),$$

$$\mathbf{c} = [c_0;\ \ldots;\ c_N]$$

The consolidated cost and constraint matrices have a repeated Kronecker block structure in linear time invariant (LTI) MPC problems. The right hand side vector $\mathbf{r}(t)$ changes at each time step with new state measurements and disturbance predictions.

5.2 Algorithm Choices for Convex Quadratic Programs

As with the choice of variables and constraint structure, we similarly have a choice and set of tradeoffs to make when selecting which class of optimization algorithm to use to solve the QP. In Chapters 2 and 3 we focused on interior point methods because they have shown in real world use cases to be the most robust choice on challenging non-convex problems with many nonlinear equality and inequality constraints. For convex QPs, more classes of algorithms will converge reliably. Interior point methods are expected to converge in a small number of iterations on convex problems, and the worst-case number of iterations is polynomial in the number of constraints. However each iteration is expensive, requiring the solution of a linear system which changes values at each iteration (due to the barrier terms for inequality constraints) to calculate the Newton step direction. Interior point methods can also be difficult to warm start from previous solutions.

Active set algorithms, including extensions of the simplex method to handle quadratic cost functions, have a long history and perform well in practice when constraints are linear. The worst-case number of iterations of an active set method can be exponential in the number of constraints, but pathological problems are uncommon. When solving a sequence of closely related optimization problems as in closed-loop MPC, active set methods are attractive because warm-starting from a previous solution is effective at reducing the number of

iterations. The linear algebra operations at each iteration of an active set method are irregular and change size as the active set of constraints changes, which can pose implementation challenges.

There has been substantial recent interest in first order methods, which converge more slowly than interior point or active set methods but have simpler iterations that can be easier to parallelize. The alternating direction method of multipliers (ADMM) [12] has been applied in many variations. In [31] an ADMM method was applied specifically to MPC problems and designed to run on FPGA platforms. In the next section we will follow a similar algorithm construction but paying more specific attention to problem structure, and targeting conventional multicore parallel processors.

5.3 Alternating Direction Method of Multipliers

We first introduce a redundant copy of the optimization variables into problem (5.4), with an equality constraint and penalty cost so that the optimal solution of this modified problem is equal to the solution to (5.4). The trick here is that the inequality constraints are applied on one copy of the variables, and the equality constraints are applied on the other copy.

$$\begin{aligned} \min_{\mathbf{z},\mathbf{y}} \ & \frac{1}{2}\mathbf{y}^T\mathbf{H}\mathbf{y} + \frac{\rho}{2}||\mathbf{y}-\mathbf{z}||^2 \\ \text{s.t. } & \mathbf{J}\mathbf{y} = \mathbf{r}(t) \\ & \mathbf{K}\mathbf{z} \leq \mathbf{c} \\ & \mathbf{z} = \mathbf{y} \end{aligned} \tag{5.5}$$

The dual of problem (5.5) is given by

$$\max_{\nu} \min_{\mathbf{z},\mathbf{y}} \ L(\mathbf{z},\mathbf{y},\nu), \tag{5.6}$$

$$\text{where } L(\mathbf{z},\mathbf{y},\nu) = \frac{1}{2}\mathbf{y}^T\mathbf{H}\mathbf{y} + \frac{\rho}{2}||\mathbf{y}-\mathbf{z}||^2 + \mathbb{I}_{\text{eq}}(\mathbf{y}) + \mathbb{I}_{\text{ineq}}(\mathbf{z}) + \nu^T(\mathbf{y}-\mathbf{z}),$$

$$\mathbb{I}_{\text{eq}}(\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{J}\mathbf{y} = \mathbf{r}(t) \\ \infty & \text{otherwise} \end{cases},$$

$$\text{and } \mathbb{I}_{\text{ineq}}(\mathbf{z}) = \begin{cases} 0 & \text{if } \mathbf{K}\mathbf{z} \leq \mathbf{c} \\ \infty & \text{otherwise} \end{cases}.$$

The alternating direction method of multipliers applies a splitting method to problem (5.6). First, the Lagrangian is minimized over $\mathbf{y}$ while holding $\mathbf{z}$ constant. Then it is minimized over $\mathbf{z}$ while holding $\mathbf{y}$ constant. The dual multiplier variables are updated based on the difference between the split primal variable vectors, and the process is repeated until the two sets of variables converge to equaling one another at the solution to the problem. Refer to [12] for detailed convergence properties of the method.

$$\mathbf{y}^{(i+1)} = \arg\min_{y} L(\mathbf{z}^{(i)}, y, \nu^{(i)}) \tag{5.7a}$$

$$\mathbf{z}^{(i+1)} = \arg\min_{z} L(z, \mathbf{y}^{(i+1)}, \nu^{(i)}) \tag{5.7b}$$

$$\nu^{(i+1)} = \nu^{(i)} + \rho(\mathbf{y}^{(i+1)} - \mathbf{z}^{(i+1)}) \tag{5.7c}$$

Computationally, step (5.7a) is an equality constrained least squares problem, step (5.7b) is a projection into the inequality feasible set, and step (5.7c) is dual gradient ascent.

The equality constrained least squares step (5.7a) is the most expensive step in this algorithm. The solution is given by the following linear system.

$$\begin{bmatrix} \mathbf{H} + \rho I & \mathbf{J}^T \\ \mathbf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}^{(i+1)} \\ \lambda^{(i+1)} \end{bmatrix} = \begin{bmatrix} \rho\mathbf{z}^{(i)} - \nu^{(i)} \\ \mathbf{r}(t) \end{bmatrix} \tag{5.8}$$

5.4 Linear System Solution for Equality Constrained Least Squares

We are again presented with a choice and tradeoffs in how to solve this linear system of equations at each ADMM iteration. The authors of [31] chose to calculate the explicit inverse of the matrix on the left hand side of (5.8). If the penalty parameter ρ is held constant then this can be performed offline, however the inverse of a sparse structured matrix does not preserve that structure in general cases. The block bandedness of the original matrix, and the repeated Kronecker structure in the time invariant case, are destroyed when an explicit inverse is calculated. The memory consumption of storing the dense inverse will scale quadratically with the prediction horizon. Multiplying a new right hand side vector by the stored inverse matrix is at least trivially parallelizable, but unless the number of parallel processors is larger than the length of the prediction horizon it is not clear whether this is worth the additional work caused by failing to preserve the problem structure.

Rather than storing the inverse of the matrix, one could calculate and store a structure-preserving factorization. A modified Cholesky factorization applied to the quasidefinite block banded matrix will result in factors that have a bandwidth at most twice as wide as the original matrix. The total number of floating point operations will therefore be much lower for problems with long horizons than if an explicit inverse were used, but the backward substitution operation that is applied to each new right hand side is inherently recursive so more difficult to parallelize than dense matrix-vector multiplication. And while the factorization preserves the block banded structure, it does not preserve the repeated Kronecker structure, since the dynamic propagation over the time horizon will result in different magnitudes in the triangular factors for each time step.

We chose to investigate Krylov subspace methods as an alternative approach. These solve the linear system of equations approximately with an iterative method based on repeated

matrix-vector multiplication, but using the original structured matrix rather than a dense unstructured inverse. If we are solving the linear systems iteratively as an inner loop inside an outer loop optimization algorithm (here ADMM), we expect that the linear system solution can be warm started using the previous outer iteration's solution as a good starting guess.

The sparse matrix-vector multiplication kernel can effectively reuse block structure, and eliminate the need to store duplicate copies of the repeated time invariant problem data.

$$\left(I_{N-1} \otimes \begin{bmatrix} F & G & -I \\ A & B & 0 \end{bmatrix}\right) \begin{bmatrix} x_1 \\ u_1 \\ s_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ s_{N-1} \end{bmatrix} = \text{vec}\left(\begin{bmatrix} F & G & -I \\ A & B & 0 \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_{N-1} \\ u_1 & \cdots & u_{N-1} \\ s_1 & \cdots & s_{N-1} \end{bmatrix}\right)$$

The Kronecker product structure means we can reshape a large sparse matrix-vector product into a small matrix-matrix product. Matrix-matrix operations are known to have better cache efficiency due to performing $O(n^3)$ floating point operations on $O(n^2)$ bytes of data. Matrix-vector operations do $O(n^2)$ work on $O(n^2)$ data, and vector-vector operations do $O(n)$ work on $O(n)$ data. Operations with a low ratio of work per byte, so-called arithmetic intensity, are limited by memory bandwidth rather than peak floating point throughput on modern processor architectures.

5.5 Implementation and Numerical Results

We implemented a Krylov subspace method based ADMM algorithm using the templated C++ MINRES [54] code available from `https://code.google.com/p/tminres/`. We applied a simple diagonal row norm preconditioner to the linear system (5.8). The penalty parameter ρ was held constant. Dense and sparse linear algebra operations were performed using Intel MKL version 10.3.

We applied the algorithm to the coupled masses test MPC example from [65]. We ex-

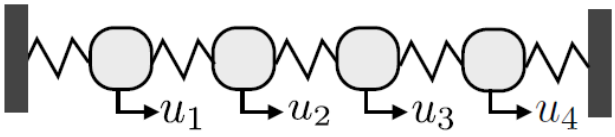


Figure 5.1: Coupled masses MPC example from [65]

tended the problem to 20 masses (40 states) and a 20 step prediction horizon. Results are shown in Fig. 5.5. As expected, the number of Krylov iterations at each outer ADMM

iteration starts high when both the linear system solution and the ADMM problem data are changing rapidly in early iterations. We limited the maximum number of Krylov iterations to 100. As the ADMM iterations proceed, the linear system warm starting behavior becomes apparent, and later ADMM iterations are several times faster than early iterations.

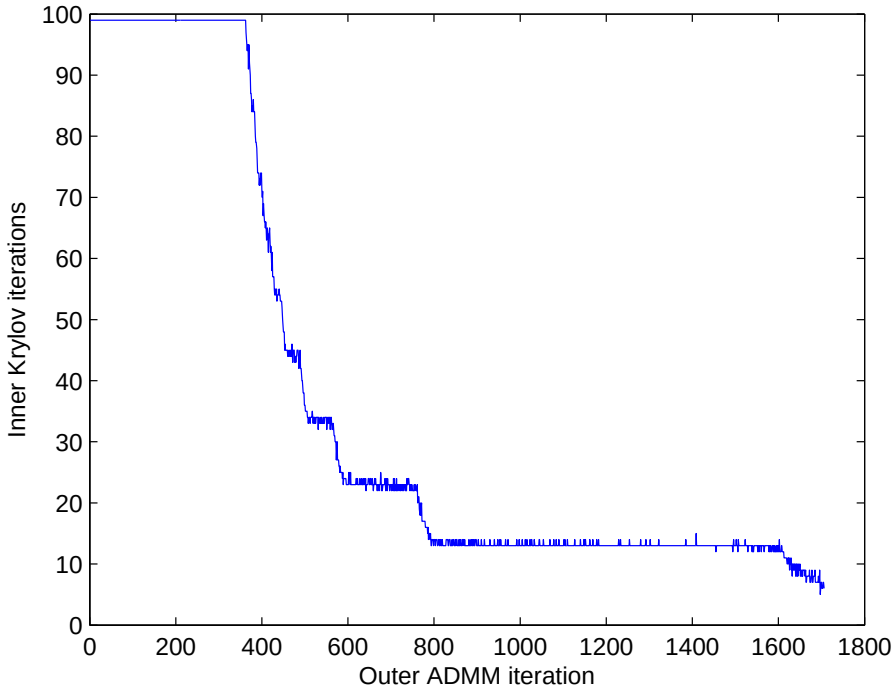


Figure 5.2: Number of inner Krylov iterations at each outer ADMM iteration

On an Intel Xeon E5410, our implementation takes an average of 72 μs per inner Krylov iteration. Comparing the performance to a block structured factorization based interior point algorithm from [20] on the same test problem, our algorithm is 80 times faster per inner iteration, but requires over 6000 times as many total inner iterations. In this case the ADMM algorithm's convergence was too slow to be worth the improved utilization of problem structure.

It is possible that this particular test problem was poorly conditioned, or not large enough to properly take advantage of parallelism. Experiments than enabled multithreading in the MKL linear algebra resulted in net slowdowns on this example due to synchronization overhead. With a more sophisticated problem-dependent preconditioner the Krylov method would likely have converged faster. Preconditioner design for optimization algorithms is not yet a solved problem; there are not yet many general-purpose approaches that work robustly. For control applications it is worth investigating automated optimal preconditioner design,

which could be an expensive optimization problem of its own but would be performed offline given a specific system, cost function and constraints.

Chapter 6

Summary of Contributions and Future Outlook

In this work we have shown the effectiveness of model predictive control for addressing complex linear and nonlinear constrained multivariable control problems, using engineering knowledge of system dynamics and disturbance predictions as an integral part of the online control calculation. Solving optimization problems online is necessary for MPC of nonlinear or large-scale systems, and poses a computational challenge to solve in real time for large systems or fast dynamics.

Beyond the simplest simulation examples or demonstrations of the existence of local optima, existing optimization modeling tools in Matlab did not scale well. We developed the Berkeley Library for Optimization Modeling to fill a need for automatically translating system model, cost function, and constraint specifications from mathematical representations into efficient function and derivative evaluations that can be used by optimization solvers. Today, newer tools such as JuMP [22] and the Julia language are lowering the barrier to entry to sophisticated automatic differentiation techniques, and the ability to write high performance algorithm implementations in a high productivity language.

The important task that these tools perform is preserving the structural sparsity information of a user specified model all the way to the low level optimization algorithm and underlying linear algebra. This work has explored the boundaries of the current state of the art in optimization algorithms and sparse linear algebra and how to reach that point with reduced engineering effort. The HVAC control application will require an entirely automated data collection, model identification, control design and deployment process before the energy savings can recoup the engineering investment to develop and deploy advanced controls at scale.

There remain many open questions before the current state of the art can be dramatically improved upon. Will existing direct solvers for sparse linear systems scale to larger numbers of cores, or solving the same problem faster, or solving larger problems in the same amount of time? Will high performance sparse solvers and optimization algorithms be possible to write entirely within a higher level language, in an open source redistributable environment for wide

deployment on large clusters or cloud servers or small embedded systems? Will alternate parallel architectures like graphics processors be possible to use for solving complicated non-convex optimization problems? Will better methods for designing preconditioners be developed to enable first order optimization methods and/or iterative linear algebra to clearly outperform interior point methods with sparse direct factorizations?

There is not yet enough of a corpus of open model formulations and test cases to make these questions clearly answerable. The operations research community has used standard sets of difficult test problems as a yardstick with which to measure new developments for many years. Control and MPC researchers should work towards the same ideal of openness and reproducibility.

Bibliography

[1] V. Adetola et al. "Model Predictive Control and Fault Detection and Diagnostics of a Building Heating, Ventilation, and Air Conditioning System". In: *International High Performance Buildings Conference*. July 2014, p. 152.

[2] J. Åkesson. "Optimica An Extension of Modelica Supporting Dynamic Optimization". In: *In 6th International Modelica Conference 2008*. Modelica Association, 2008, pp. 57–66.

[3] J. Åkesson et al. "Modeling and Optimization with Optimica and JModelica.org—Languages and Tools for Solving Large-Scale Dynamic Optimization Problem". In: *Computers and Chemical Engineering* 34.11 (Nov. 2010), pp. 1737–1749.

[4] P. Amestoy et al. "MUMPS: A general purpose distributed memory sparse solver". In: *Proceedings of the 5th International Workshop on Applied Parallel Computing*. PARA '00. London, UK: Springer-Verlag, 2001, pp. 121–130.

[5] K. Asanovic et al. *The Landscape of Parallel Computing Research: A View from Berkeley*. Tech. rep. UCB/EECS-2006-183. EECS Department, University of California, Berkeley, Dec. 2006. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html`.

[6] S. Bengea et al. "Implementation of Model Predictive Control for an HVAC System in a Mid-Size Commercial Building". In: *HVAC&R Research* 20.1 (2014), pp. 121–135.

[7] S. Bengea et al. "Model Predictive Control for Mid-Size Commercial Building HVAC: Implementation, Results and Energy Savings". In: *Second International Conference on Building Energy and Environment*. Aug. 2012, pp. 979 –986.

[8] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Englewood Cliffs, NJ: Springer-Verlag, 1989.

[9] J. Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *CoRR* abs/1411.1607 (2014). URL: `http://arxiv.org/abs/1411.1607`.

[10] J. Bisschop. *AIMMS Optimization Modeling*. LULU PR, 2006. ISBN: 9781847539120.

[11] F. Borrelli. *Constrained optimal control of linear and hybrid systems*. Vol. 290 of Lecture Notes in Control and Information Sciences. Springer-Verlag, 2003.

[12] S. Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Found. Trends Mach. Learn.* 3.1 (Jan. 2011), pp. 1–122.

[13] J. R. Bunch and L. Kaufman. "Some stable methods for calculating inertia and solving symmetric linear systems". English. In: *Mathematics of Computation* 31.137 (1977), pp. 163–179.

[14] R. Cagienard et al. "Move blocking strategies in receding horizon control". In: *Journal of Process Control* 17.6 (2007), pp. 563 –570.

[15] M. Čižniar et al. "A MATLAB Package for Orthogonal Collocations on Finite Elements in Dynamic Optimisation". In: *Proceedings of the 15th International Conference Process Control '05*. June 2005.

[16] G. Constantinides. "Parallel architectures for model predictive control". In: *European Control Conference, 2009*. Aug. 2009, pp. 138–143.

[17] F.E. Curtis, O. Schenk, and A. Wachter. "An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations". In: *SIAM Journal on Scientific Computing* 32.6 (2010), pp. 3447–3475.

[18] G.G. Dahlquist. "A special stability problem for linear multistep methods". In: *BIT Numerical Mathematics* 3 (1 1963), pp. 27–43. ISSN: 0006-3835. URL: `http://dx.doi.org/10.1007/BF01963532`.

[19] J.W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics, 1997.

[20] A. Domahidi et al. "Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control". In: *Decision and Control, Proceedings of the 51st IEEE Conference on*. Dec. 2012.

[21] I.S. Duff. "MA57—a code for the solution of sparse symmetric definite and indefinite systems". In: *ACM Transactions on Mathematical Software* 30 (2 June 2004), pp. 118–144.

[22] I. Dunning, J. Huchette, and M. Lubin. "JuMP: A modeling language for mathematical optimization". In: *arXiv:1508.01982 [math.OC]* (2015). URL: `http://arxiv.org/abs/1508.01982`.

[23] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: a modeling language for mathematical programming*. Thomson/Brooks/Cole, 2003.

[24] P. Fritzson and V. Engelson. "Modelica: A unified object-oriented language for system modeling and simulation". In: *ECOOP 98 Object-Oriented Programming*. Ed. by Eric Jul. Vol. 1445. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, pp. 67–90.

[25] N.I. Gould, J.A. Scott, and Y. Hu. "A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations". In: *ACM Transactions on Mathematical Software* 33 (2 June 2007).

[26] W.T. Grondzik. *Air-conditioning system design manual*. ASHRAE professional series. Butterworth-Heinemann, 2007.

[27] A. Gupta, M. Joshi, and V. Kumar. *WSMP: A High-Performance Shared- and Distributed-Memory Parallel Sparse Linear Equation Solver*. Tech. rep. IBM Research Report, 2001.

[28] A. Gupta, G. Karypis, and V. Kumar. *Highly scalable parallel algorithms for sparse matrix factorization*. Tech. rep. IEEE Transactions on Parallel and Distributed Systems, 1994.

[29] G.P. Henze, C. Felsmann, and G. Knabe. "Evaluation of optimal control for active and passive building thermal storage". In: *International Journal of Thermal Sciences* 43.2 (2004), pp. 173–183.

[30] HSL. *A collection of Fortran codes for large scale scientific computation*. http://www.hsl.rl.ac.uk. 2011.

[31] J.L. Jerez et al. "Embedded Online Optimization for Model Predictive Control at Megahertz Rates". In: *CoRR* abs/1303.1090 (2013). URL: `http://arxiv.org/abs/1303.1090`.

[32] J.L. Jerez et al. "Parallel MPC for real-time FPGA-based implementation". In: *Proceedings of the 18th IFAC World Congress*. Vol. 18. 2011.

[33] J. Kallrath. *Modeling Languages in Mathematical Optimization*. Applied Optimization. Kluwer Academic Publishers, 2004.

[34] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM J. Sci. Comput.* 20.1 (Dec. 1998), pp. 359–392.

[35] A. Kelman and F. Borrelli. "Bilinear model predictive control of a HVAC system using sequential quadratic programming". In: *18th IFAC World Congress*. Aug. 2011, pp. 9869 –9874.

[36] A. Kelman and F. Borrelli. "Parallel nonlinear predictive control". In: *Communication, Control, and Computing, 2012 50th Annual Allerton Conference on*. Oct. 2012, pp. 71–78.

[37] A. Kelman, Y. Ma, and F. Borrelli. "Analysis of local optima in predictive control for energy efficient buildings". In: *Journal of Building Performance Simulation* 6.3 (2013), pp. 236–255.

[38] A. Kelman et al. "BLOM: The Berkeley Library for Optimization Modeling". In: *American Control Conference (ACC), 2014*. June 2014, pp. 2900–2905.

[39] M.J. Koegel and R. Findeisen. "Parallel architectures for model predictive control". In: *4th IFAC Nonlinear Model Predictive Control Conference*. Vol. 4. 1. Aug. 2012.

[40] L.S. Lasdon. "Duality and Decomposition in Mathematical Programming". In: *Systems Science and Cybernetics, IEEE Transactions on* 4.2 (July 1968).

[41] X. Li. *Direct Solvers for Sparse Matrices*. Tech. rep. `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf`. Lawrence Berkeley National Laboratory, July 2013.

[42] X. Lin, M. Johansson, and S.P. Boyd. "Simultaneous routing and resource allocation via dual decomposition". In: *Communications, IEEE Transactions on* 52.7 (July 2004), pp. 1136–1144.

[43] J. Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB". In: *Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004. URL: `http://users.isy.liu.se/johanl/yalmip`.

[44] S. Longo et al. "Parallel Move Blocking Model Predictive Control". In: *Decision and Control, Proceedings of the 50th IEEE Conference on* (Dec. 2011), pp. 1239–1244.

[45] M. Lubin and I. Dunning. "Computing in Operations Research Using Julia". In: *INFORMS Journal on Computing* 27.2 (2015), pp. 238–248.

[46] Y. Ma and F. Borrelli. "A Distributed Predictive Control Approach to Building Temperature Regulation". In: *2011 American Control Conference*. June 2011.

[47] Y. Ma et al. "Model predictive control for the operation of building cooling systems". In: *2010 American Control Conference*. June 2010, pp. 5106 –5111.

[48] Y. Ma et al. "Predictive control for energy efficient buildings with thermal storage". In: *Control Systems Magazine, IEEE* 32.1 (2012), pp. 44–64.

[49] D.Q. Mayne et al. "Constrained model predictive control: Stability and Optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[50] J.M. McQuade. *A Systems Approach to High Performance Buildings*. Tech. rep. `http://gop.science.house.gov/Media/hearings/energy09/april28/mcquade.pdf`. United Technologies Corporation, Apr. 2009.

[51] A.S. Nemirovski and M.J. Todd. "Interior-point methods for optimization". In: *Acta Numerica* 17 (2008), pp. 191–234.

[52] J. Nocedal and S. J. Wright. "Numerical Optimization". In: Springer-Verlag, 2006. Chap. 18.

[53] F. Oldewurtel et al. "Energy Efficient Building Climate Control using Stochastic Model Predictive Control and Weather Predictions". In: *2010 American Control Conference*. June 2010, pp. 5100–5105.

[54] C.C. Paige and M.A. Saunders. "Solution of Sparse Indefinite Systems of Linear Equations". In: *SIAM J. Numerical Analysis* 12 (1975), pp. 617–629.

[55] P.C. Piela et al. "ASCEND: an object-oriented computer environment for modeling and analysis: The modeling language". In: *Computers & Chemical Engineering* 15.1 (1991), pp. 53 –72.

[56] S.J. Qin and T.A. Badgwell. "A survey of industrial model predictive control technology". In: *Control Engineering Practice* 11.7 (2003).

[57] A. Rantzer. "Dynamic dual decomposition for distributed control". In: *American Control Conference, 2009*. June 2009, pp. 884–888.

[58] R.E. Rosenthal. *GAMS: A User's Guide*. GAMS Development Corporation, 2008.

[59] O. Schenk and K. Gärtner. "On fast factorization pivoting methods for symmetric indefinite systems". In: *Electronic Transactions on Numerical Analysis* 23.1 (2006), pp. 158–179.

[60] O. Schenk and K. Gärtner. "Solving unsymmetric sparse systems of linear equations with PARDISO". In: *Future Generation Computer Systems* 20.3 (2004), pp. 475–487.

[61] S.T. Taylor. "Increasing Efficiency With VAV System Static Pressure Setpoint Reset". In: *ASHRAE Journal* 49.6 (2007), pp. 24–33.

[62] M. J. Tenny, J. B. Rawlings, and S. J. Wright. "Closed-loop behavior of nonlinear model predictive control". In: *AIChE Journal* 50.9 (2004), pp. 2142–2154.

[63] The MathWorks, Inc. *Simulink*. Version 8.0. Sept. 11, 2012. URL: `http://www.mathworks.com/products/simulink/`.

[64] A. Wächter and L. T. Biegler. "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[65] Y. Wang and S. Boyd. "Fast Model Predictive Control Using Online Optimization". In: *Control Systems Technology, IEEE Transactions on* 18.2 (Mar. 2010), pp. 267–278.

[66] S.J. Wright. "Interior Point Methods For Optimal Control Of Discrete-Time Systems". In: *Journal of Optimization Theory and Applications* 77 (1993), pp. 161–187.

[67] V.M. Zavala and M. Anitescu. *Scalable nonlinear programming via exact differentiable penalty functions and trust-region Newton methods*. Tech. rep. Argonne National Laboratory, July 2012.