Distributed Model Predictive Control for Building Temperature Control

Kirsten Carstaedt

LUND UNIVERSITY

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5998--SE
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2016

Affidavit (Eidesstattliche Erklärung)

This thesis has been prepared on initiative and under guidance of my advisors. For the preparation I have not used others than the indicated aids.

Die vorliegende Arbeit habe ich auf Initiative und unter Anleitung meiner Betreuer angefertigt. Bei der Erstellung habe ich keine anderen als die angegebenen Hilfsmittel verwendet.

Kaiserslautern, November 25, 2015

Kirsten Carstaedt

Abstract

This thesis and technical report concentrates on distributed control using a distributed model predictive scheme. The model of a two room house and three room houses is build and a distributed model predictive control (MPC) algorithm is implemented in order to reach specified room temperatures with minimized energy effort in each room. For reference tracking Target Calculation and the delta input scheme are used. The MPC optimization problem is solved at each time step through an iterative method, where the number of iterations is reduced through a stopping criterion guaranteeing stability and a prespecified amount of performance and feasibility. The optimization problem is divided up into subproblems, where each subproblem takes less computational effort than the central optimization problem. Due to the possibility of coupling between subsystems, communication between the subsystems is needed. The reference values are reached and iterations needed to solve the optimization are reduced with the stopping condition. This method saves computing time and gives privacy to each subsystem, since only required information is communicated. Also the subsystems get less susceptible to the failure of one coupled subsystem, since if one subsystem fails, the others could go on. But, due to the needed communication, this method is more suitable for large systems with sparse coupling. For a small system, or too much coupling the communication effort will get to high.

Contents

List of Figures

1 Introduction

1.1 Motivation

A house is a locally distributed system. To control the temperature of all rooms to a specific temperature takes a lot of computational effort. Model Predictive Control can follow a reference value and consider the constraints on temperature and heaters. For large building-systems Distributed Model Predictive control can be used. The house is divided into subsystems which are controlled by their own Model Predictive unit. This way the computational effort of each unit is reduced.

1.2 Objective

The objective of this thesis is to implement the accelerated gradient algorithm introduced in "On feasibility, stability and performance in distributed model predictive control" [GR13]. Reference tracking and disturbance rejection are added to the algorithm. The implemented accelerated gradient method will be compared to the central solution introduced in the script "Model Predictive Control" [Gör15a].

1.3 Outline

The thesis is structured as follows: The Second Chapter introduces convex optimization problems [BV09]. In the third Chapter the concept of Model Predictive control is described and the reformulation of the difference equations into a prediction model suitable for the accelerated gradient method [GR13], [GDK+13] and the central model predictive control [Gör15c] without reference tracking are done. In the fourth Chapter the differential equations for the used house-model is build [Bol06] and discretized[Gör15b] and the corresponding parameters [www] are listed. In Chapter 5 reference tracking through Target Calculation and Delta Input Formulation [Gör15d] is introduced to the accelerated gradient method. Chapter 6 discusses the results. Chapter 7 is conclusions and Future work.

2 Convex Optimization

2.1 Problem Formulation

The general optimization problem is to optimize the cost function f(x) with respect to the equality constraints $h(x) = 0$ and inequality constraints $g(x) \leq 0$. It is given by:

$$\begin{aligned} &\underset{x}{\text{minimize}}\, f(x) \\ &\text{subject to } x \in X \end{aligned} \tag{2.1}$$

$$\text{with } X = \{x \in R^n | h(x) = 0, g(x) \leq 0\}. \tag{2.2}$$

We call X the feasible set, containing all variables that fulfil the constraints given by *h(x)* and *g(x)*. A $x \in X$ is one of the possibilities to minimize the optimization problem satisfying the constraints. We call x^* the minimizing variable to the problem (2.1). x^* is part of the feasible set, but also minimizes the cost function. The equality constraints $h(x)$ contain difference equations describing the behaviour of the modelled plant. The inequality constraints $g(x)$ describe the maximum and minimum value of the states of the plant model. For example:

$$\begin{aligned} &\underset{u,x}{\text{minimize}}\, \tfrac{1}{2}(u_1(k)^T R_1 u_1(k) + x_1(k)^T Q_1 x_1(k)) \\ &\text{subject to } x_1(k+1) = a_{11}x_1(k) + b_{11}u_1(k) + b_{w11}w_1(k) \\ &u_{\min} \leq u(k) \leq u_{\max} \\ &x_{\min} \leq x(k) \leq x_{\max}. \end{aligned} \tag{2.3}$$

If we solve this problem, we minimize the energy of input u_1 and state x_1. Our equality constraint is the differential equation describing our system model. The inequality constraints describe maximum and minimum of x_1 and u_1.
u_1: input variable, stands for the heat flow of heater 1
x_1: state variable, stands for the temperature in room 1
w_1: disturbance variable, stands for the disturbance, outside temperature
The advantage of a convex optimization problem is that the Minimum we find is a global one. An Optimization Problem is called convex if X is a convex set and $f(x)$ is a convex function.

2.2 Convexity

Convex Sets

If C is a convex set, the line segment between any two points x_1,$x_2 \in C$ lies in C. This means $\forall\ x_1, x_2 \in C$:

$$\alpha x_1 + (1-\alpha)x_2 \in C \text{ with } \alpha[0,1] \tag{2.4}$$

Convex Functions

A convex optimization problem has a convex cost function. A cost function $f(x) : C \to R$ is convex if and only if C is a convex set and if $\forall x_1, x_2 \in C$ and $\alpha \in [0,1]$:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2) \tag{2.5}$$

For differentiable f there are two useful conditions:
First Order Condition: f is differentiable, C is convex and open:
$\exists \nabla f$ for each point of C then one has $\forall x_1, x_2 \in C$

$$f(x_2) \geq f(x_2) + \nabla f(x_1)^T (x_2 - x_1) \tag{2.6}$$

∇ is the gradient of $f(x)$
Second Order Condition: f is twice differentiable, C is convex and open:
$\exists H_f = \nabla^2 f\ \forall\ x \in C$ then one has $\forall\ x \in C$ with C convex we have:

$$H_f = \nabla^2 f(x) \succeq 0 \tag{2.7}$$

H_f is the Hessian matrix of $f(x)$.

Some welcome Properties

1. C convex, $\beta \in R$ implies: $\beta C = \{x | x = \beta v, v \in C\}$ convex
2. C_1, C_2 convex implies: $C_1 + C_2 = \{x | x = v_1 + v_2, v_1 \in C_1, v_2 \in C_2\}$ convex
3. For C_1, C_2 convex: $C_1 \cap C_2$ convex
4. The sum of convex functions is convex: $f_i(x)$ is convex on C, then $f(x) = \sum_{i=1}^{N} \alpha_i f_i(x)$ is convex for $\alpha_i \geq 0, i = \{i, ..., N\}$
5. The quadratic function $f(x) = x^T P x$ is convex on X for symmetric and positive semi-definite P. Since due to the second order condition for convex functions we need: $H_f(x) \succeq 0$ for a quadratic and convex function.

Examples

Cost Function
A cost function that would be used in model predictive control is for example:

$$f(x_i, u_i) = f_1(x_i) + f_2(u_i) \tag{2.8}$$
$$f_1(x_i) = \tfrac{1}{2} x_i^T Q_i x_i \tag{2.9}$$
$$f_2(u_i) = \tfrac{1}{2} u_i^T R u_i \tag{2.10}$$

The weighting matrices Q_i, R_i are quadratic and positive semi-definite $Q_i \succeq 0$, $R_i \succeq 0$, so the functions $f_1(x_i)$ and $f_2(x_i)$ are convex functions due to property 4. Due to property 3 sum of convex functions is convex.

Convex Sets
The equality constraints $A^T x = b$ are a intersection of hyperplanes called linear variety. A intersection of convex sets is a convex set.
A hyper plane $\{x \in R^n | A^T x = b\}$is a convex set:
With $A = (a_1^T, a_2^T, \ldots, a_n^T) \in R^n$, $b \in R$ we have

$$A^T x = a_1^T x_1 + a_2^T x_2 + \cdots + a_n^T x_n = b \tag{2.11}$$

To proof convexity we use the variables x_a, x_b: $A^T x_a = b$, $A^T x_n = b$

$$\begin{aligned} y &= \alpha x_a + (1-\alpha) x_b \\ a^T y &= \alpha A^T x_a + (1-\alpha) A^T x_b \\ &= \alpha b + (1-\alpha) b \\ &= \alpha b + b - \alpha b \\ &= b \end{aligned}$$

The inequality constraints $C^x \leq d$ are a convolution of half spaces called Polyhedron. A intersection of convex sets is a convex set.
A half space $\{x \in R | C^T x \leq d\}$is a convex set. With $C = (c_1^T, c_2^T, \ldots, c_n^T)^T \in R^n$ $d \in R$ we have

$$C^T x = c_1 x_1 + c_2 x_2 + ... + c_n x_n = d \tag{2.12}$$

To proof convexity we use the variables x_a, x_b: $C^T x_a \leq d$, $C^T x_b \leq d$

$$\begin{aligned} y &= \alpha x_a + (1-\alpha) x_b \\ C^T y &= \alpha C^T x_a + (1-\alpha) C^T x_b \\ &\leq \alpha d + (1-\alpha) d \\ &= \alpha d + d - \alpha d \\ &= d \end{aligned}$$

2.3 The Lagrange Duality Theory

The Lagrangian relaxation turns the constraints into a part of the cost function, where the violations of the constraints are weighted. Equality constraints $h(x) = 0$ as in (2.1) should be zero such that the minimum can be found, inequality constraints $g(x) \leq 0$ as in (2.1) should hold. The cost function is $f_0(x)$. The Lagrangian is:

$$L(x, \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{p} \mu_i g_i(x) \tag{2.13}$$

- λ_i: Lagrange multiplier, associated with the equality constraints $h_i(x) = 0$
- $\mu_i \geq 0$: Lagrange multiplier, associated with the inequality constraints $g_i(x) \leq 0$

We call

$$v(\lambda, \mu) = \inf_{x \in D} L(x, \lambda, \mu) \tag{2.14}$$

the dual function. The dual problem is now:

$$\max_{\lambda, \mu \leq 0} v(\lambda, \mu) \tag{2.15}$$

The solution to this problem can be used to solve the original problem. Actually the solution to (2.14) is a lower bound.

Lower bound on the Optimal Value

Say the optimal value of the original problem (2.1) is called p^* and the solution to the dual optimization problem (2.14)is v^*. For any λ, $\mu \geq 0$ $v(\lambda, \mu)$ is a lower bound on the primal solution p^*.

$$\forall () \lambda, \mu \geq 0) \in R\text{: } v(\lambda, \mu) \leq p^* \tag{2.16}$$

To verify this we take a $x_1 \in X$, with X the feasible set defined in (2.1). Since x_1 is feasible we have the constraints:

$$h_i(x_1) = 0, \qquad g_i(x_1) \leq 0 \qquad \text{with } \lambda, \mu \in R \tag{2.17}$$

fulfilled. Due to this we can see that:

$$\sum_{i=1}^{m} \lambda_i h_i(x_1) + \sum_{i=1}^{p} \mu_i g_i(x_1) \leq 0. \tag{2.18}$$

Since the first sum consists of $h_i(x_1) = 0$ and the second consists of $g_i(x_1) \leq 0$. Inserting (2.18) into $L(x_1, \lambda, \mu)$ we see that:

$$L(x_1, \lambda, \mu) = f_0(x_1) + \sum_{i=1}^{m} \lambda_i h_i(x_1) + \sum_{i=1}^{p} \mu_i g_i(x_1) \leq f_0(x_1). \tag{2.19}$$

The following equation shows that $g(\lambda, \mu)$ is a lower bound on $f_0(x_1)$:

$$v(\lambda, \mu) = \inf_x(L(x, \lambda, \mu)) \leq L(x_1, \lambda, \mu) \leq f_0(x_1) \tag{2.20}$$

For the special case that the chosen feasible x_1 is the minimizer x^* to (2.1) the solution to the dual problem is a lower bound on the optimal value:

$$v^* \leq p^* \tag{2.21}$$

We can find that lower bound v^* on p^* using the Karush Kuhn Tucker Condition (KKT condition). Note that one can solve the primal problem using the dual problem under certain assumptions discussed in the sequel of the chapter.

The Duality Gap

The difference between the optimal value of primal (p^*) and dual problem (d^*) is called duality gap: $d^* - p^* \leq 0$. d^* and p^* are defined as:

$$p^* = \inf f(x) \qquad \text{s.t. } g(x) \geq 0, h(x) = 0 \tag{2.22}$$

$$d^* = \max v(\lambda, \mu) \qquad \text{s.t. } \mu >= 0. \tag{2.23}$$

To ensure that the duality gap is zero, we need the problem to be convex and slater's condition to hold. Slater's condition says if there is one feasible point:

$$\exists x \in X : g_i(x) < 0, h(x) = 0. \tag{2.24}$$

For a feasible x , the Problem has zero duality gap. If we have some affine inequality constraints $g_j(x)$ the condition becomes:

$$\exists x \in X : g_j(x) \leq 0, g_i(x) < 0, h(x) = 0 \tag{2.25}$$

where $g_j(x) \leq 0$ is for the affine inequality constraints. For convex problems with only affine constraints, e.g linear constraints and open domain of the cost function, slater's condition reduces to the feasibility conditions:

$$\exists x \in X : g_j(x) \leq 0, h(x) = 0 \tag{2.26}$$

and the KKT's are sufficient and necessary.

Karush Kuhn Tucker Conditions to solve the Convex Optimization Problem

We have a x_* that minimizes $L(x^*, \lambda^*, \mu^*)$, so it follows:

$$\nabla f_0(x^*) + \sum \lambda^* \nabla h_i(x^*) + \sum \mu_i^* \nabla g_i(x^*) = 0 \tag{2.27}$$

The following conditions are called "Karush Kuhn Tucker" (KKT) conditions:

$$h_i(x^*) = 0 \tag{2.28}$$
$$g_i(x^*) \leq 0 \tag{2.29}$$
$$\mu_i^* \geq 0 \tag{2.30}$$
$$\mu_i^* g_i(x^*) = 0 \tag{2.31}$$
$$\nabla f_0(x^*) + \sum \lambda_i^* \nabla h_i(x^*) + \sum \mu_i^* g_i(x^*) = 0 \tag{2.32}$$

The solution needs to fulfil the equality and inequality constraints . The weights on the inequality constraints need to be positive, such that the result is a lower bound on p^*. We need $\mu_i^* g_i(x^*) = 0$ which means for $g_i(x^*) < 0$ we need $\mu = 0$. To fulfil $\mu_i^* g_i(x^*) = 0$ we have either $\mu_i = 0$ or $g_i(x^*) = 0$. We also use the gradient of the Lagrange function $L(x^*, \lambda^*, \mu^*)$.
For non-convex problem these conditions are only sufficient. A x^* that minimizes $L(x^*, \lambda^*, \mu^*)$ will satisfy the conditions. But not every x that satisfies the conditions will be a minimizer to $L(x, \lambda, \mu)$.
For convex optimization problems with differentiable objective and constraint functions satisfyin
the KKT conditions provide necessary and sufficient conditions for optimality.The solution we get through the KKT conditions is the optimal value p^*.

3 Model Predictive Control

In Model predictive control we use a mathematical model describing the plant dynamics to predict the behaviour of the plant over a prediction horizon. The current plant states $x(k)$ are measured and a prediction model is build. Based on this prediction model we calculate the optimal input sequence $U(k)$ to reach zero or a specified reference value while minimizing the cost function. The first value of this input sequence $u(0)$ is implemented to the plant. The next time step the new current plant states $x(k)$ are measured and we start again by predicting the behaviour of the plant from this new starting point and calculating the optimal input sequence $U(k)$. For now the reference value is zero.

3.1 Central Prediction Model

The state equations (equality constraints) are reformulated into a prediction model only depending on the current state and input. The state equations for the central problem are:

$$\begin{aligned}
x(k+1) &= ax(k) + bu(k) + b_w w(k) \\
x(k+2) &= ax(k+1) + bu(k+1) + b_w w(k+1) \\
&= a^2 x(k) + abu(k) + ab_w w(k) + bu(k+1) + b_w w(k+1) \\
x(k+2) &= ax(k+2) + bu(k+2) + b_w w(k+2) \\
&= a^3 x(k) + a^2 bu(k) + a^2 b_w w(k) \\
&\quad + abu(k+1) + ab_w w(k+1) + bu(k+2) + b_w w(k+2) \\
&\vdots \\
x(k+N) &= a^N x(k) + a^{N-1} bu(k) + a^{N-1} b_w w(k) + \dots \\
&\quad + a^1 bu(k) + a^1 b_w w(k) + a^0 bu(k) + a^0 b_w w(k).
\end{aligned} \tag{3.1}$$

These state equations put together into one matrix form, where $X(k)$ is depending only on the current measurement $x(k)$ and the input sequence $U(k)$:

$$X(k) = \Phi x(k) + \Gamma_u U(k) + \Gamma_w W(k) \tag{3.2}$$

is called prediction model. The matrices are:

$$X(k) = \begin{pmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N) \end{pmatrix}, U(k) = \begin{pmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{pmatrix}, W(k) = \begin{pmatrix} w(k) \\ w(k+1) \\ \vdots \\ w(k+N-1) \end{pmatrix}, \tag{3.3}$$

$$\Phi = \begin{pmatrix} a \\ a^2 \\ \vdots \\ a^N \end{pmatrix}, \Gamma_u = \begin{pmatrix} a^0 b & 0 & \dots & 0 \\ a^1 b & a^0 b & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ a^{N-1} b & a^{N-2} b & \dots & a^0 b \end{pmatrix}, \Gamma_w = \begin{pmatrix} a^0 b_w & 0 & \dots & 0 \\ a^1 b_w & a^0 b_w & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ a^{N-1} b_w & a^{N-2} b_w & \dots & a^0 b_w \end{pmatrix} \tag{3.4}$$

The cost function (3.5)

$$V_N(x(k), U(k)) = x^T(k+N) P x(k+N) + \sum_{i=0}^{N-1} x^T(k+i) Q x(k+i) + u^T(k+i) R u(k+i) \tag{3.5}$$

$$= x^T(k) Q x(k) + X^T(k) \Omega X(k) + U^T(k) \Psi U(k) \tag{3.6}$$

also needs to be reformulated in a matrix form. The following matrices

$$\Omega = \begin{pmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ 0 & 0 & Q & \ddots & 0 \\ 0 & 0 & 0 & \dots & P \end{pmatrix}, \Psi = \begin{pmatrix} R & 0 & 0 & \dots & 0 \\ 0 & R & 0 & \dots & 0 \\ 0 & 0 & R & \ddots & 0 \\ 0 & 0 & 0 & \dots & R \end{pmatrix} \tag{3.7}$$

build the matrix form (3.6). The matrices Q, P are symmetric and positive-semidefinite, P is symmetric and positive-definite:
$P = P^T \succeq 0$, $Q = Q^T \succeq 0$, $R = R^T \succ 0$.
This implies that $\Omega = \Omega^T \succeq 0$, $\Psi = \Psi^T \succeq 0$.
To eliminate the minimization over x we insert the prediction model $X(k) = \Phi x(k) + \Gamma_u U(k) + \Gamma_w W(k)$ into the cost function $V_N(x(k), U(k)) = x^T(k) Q x(k) + X^T(k) \Omega X(k) + U^T(k) \Psi U(k)$ simplifying the equations:

$$\begin{aligned} V_N(x(k), U(k)) &= x^T(k) Q x(k) \\ &\quad + (x^T \Phi^T + U^T(k) \Gamma_u^T + W^T(k) \Gamma_w^T) \Omega (\Phi x(i) + \Gamma_u^T U(k) \\ &\quad + \Gamma_w^T W(k)) + U^T(k) \Psi U(k) \\ &= x^T(k) Q x(k) + U^T(k) \Psi U(k) \\ &\quad + (x^T(k) \Phi^T \Omega + U^T(k) \Gamma_u^T \Omega + W^T(k) \Gamma_w^T \Omega) \Phi x(k) \\ &\quad + (x^T(k) \Phi^T \Omega + U^T(k) \Gamma_u^T \Omega + W^T(k) \Gamma_w^T \Omega) \Gamma_u U(k) \end{aligned}$$

$$= (x^T(k)\Phi^T\Omega + U^T(k)\Gamma_u^T\Omega + W^T(k)\Gamma_w^T\Omega)\Gamma_w W(k) \tag{3.8}$$

We summarize all terms that do not include $U(k)$ into $f(x(k), W(k))$.

$$\begin{aligned} V_N(x(k), U(k)) &= f(x(k), W(k)) + U^T(k)\Psi U(k) + UT(k)\Gamma_u^T\Omega\Phi x(k) \\ &+ (x^T(k)\Phi^T\Omega\Gamma_u U(k) + U^T(k)\Gamma_u^T\Omega\Gamma_u U(k) \qquad (3.9) \\ &+ W^T(k)\Gamma_w^T\Omega\Gamma_u U(k) + U^T(k)\Gamma_u^T\Omega\Gamma_w W(k) \\ &= U^T(k)(\Psi + \Gamma_u^T\Omega\Gamma_u)U(k) + 2x^T(k)\Phi^T\Omega\Gamma_u U(k) \qquad (3.10) \\ &+ 2W^T(k)\Gamma_w^T\Omega\Gamma_u U(k) + f(x(k), W(k)) \end{aligned}$$

$$V_N(x(k), U(k)) = U^T(k)\frac{1}{2}2(\Psi + \Gamma_u^T\Omega\Gamma_u)U(k) + (2x^T(k)\Phi^T\Omega\Gamma_u + 2W^T(k)\Gamma_w^T\Omega\Gamma_u)U(k) \tag{3.11}$$

$$\begin{aligned} &+ f(x(k), W(k)) \\ &= \frac{1}{2}U^T(k)HU(k) + (F_x + F_R)U(k) + f(x(k), W(k)) \qquad (3.12) \end{aligned}$$

Used in this reshaping is that all $x^T Mx$ in the cost function are scalar, so $x^T Mx = xMx^T$. Where M is a diagonal matrix with $M = M^T$. Without constraints the optimization problem is

$$\min_{U(k)} V_N(x(k), U(k)). \tag{3.13}$$

It can be solved by building the derivative for U(k):

$$U(k) = -H^{-1}(F_x^T + F_R^T). \tag{3.14}$$

Now we build the matrix form of the constraints only depending on $x(k)$ and $U(k)$. The used model has boxed constraints, limiting the temperature and heat flow of the heaters we do the constraints reformulation as matrix form for boxed constraints. The constraints are:

$$-u(k) \leq u_{min} \tag{3.15}$$
$$u(k) \leq u_{max} \tag{3.16}$$
$$-y(k) \leq y_{min} \tag{3.17}$$
$$y(k) \leq y_{max}. \tag{3.18}$$

The goal is to reformulate this a: $M(k+i)x(k+i) + E(k+i)u(k+i) \leq b(k+i)$. With $y(k+i) = cx(k+i)$. The matrices are:

$$M(k+i) = \begin{pmatrix} \mathbf{0}_{mxn} \\ \mathbf{0}_{mxn} \\ -c \\ c \end{pmatrix}, E(k+i) = \begin{pmatrix} -\mathbf{I}_{mxm} \\ \mathbf{I}_{mxm} \\ \mathbf{0}_{pxm} \\ \mathbf{0}_{pxm} \end{pmatrix}, b(k+i) = \begin{pmatrix} -u_{min} \\ u_{max} \\ -y_{min} \\ y_{max} \end{pmatrix} \tag{3.19}$$

The prediction form of these inequalities is:

$$D(k)x(k) + \Lambda(k)X(k) + \epsilon(k)U(k) \leq \beta(k) \tag{3.20}$$

$$D(k) = \begin{pmatrix} M(k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \Lambda(k) = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ M(k+1) & 0 & 0 & \dots & 0 \\ 0 & M(k+2) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & M(k+N) \end{pmatrix}, \tag{3.21}$$

$$\beta(k) = \begin{pmatrix} b(k) \\ b(k+1) \\ b(k+2) \\ \vdots \\ b(k+N) \end{pmatrix}, \epsilon(k) = \begin{pmatrix} E(k) & 0 & 0 & \dots & 0 \\ 0 & E(k+1) & 0 & \dots & 0 \\ \dots & \dots & \dots & \ddots & 0 \\ 0 & 0 & 0 & \dots & E(k+N) \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \tag{3.22}$$

To get a constraints form only involving $U(k)$ and $x(k)$ the prediction Model (3.2) is inserted into the constraints matrix form (3.20):

$$D(k)x(k) + \Lambda(k)\Phi x(k) + M(k)(\Gamma_u U(k) + \Gamma_w W(k)) + \epsilon(k)U(k) \leq \beta(k) \tag{3.23}$$

$$(M(k)\Gamma_u + \epsilon U(k)) \leq \beta(k) + (-D(k) - \Lambda(k)\Phi)x(k) + -\Lambda(k)\Gamma_w W(k) \tag{3.24}$$

$$AU(k) \leq \beta(k) + Wx(k) + W_w W(k). \tag{3.25}$$

This results in a quadratic program:

$$\min_{U(k)} = V_N(x(k), U(k)) \text{ s.t. } AU(k) \leq b(k) + Wx(k) \tag{3.26}$$

A problem formed like this can be solved using the matlab function "quadprog".

3.2 Separable Prediction Model

The following prediction model is build such that it is separable:

$$\begin{aligned} &\min_y \tfrac{1}{2} y^T H y \\ &\text{subject to } Ay = Bz_0 \\ &Cy \leq D. \end{aligned} \tag{3.27}$$

The vector y combines all $x(k), u(k)$ for all time steps of the prediction horizon. The optimization problem is divided into parts that are solved by their own

computational unit. The number of computational units we use to solve the distributed Model predictive control is M. Each computational unit (corresponding to one room in our model) has the state z_i and input v_i.
z_i, v_i contain all states and inputs that correspond to this computational unit.
The Dimensions are: $z_i \in R^{n_i}$, $v_i \in R^{m_i}$.
In the case of normal Target Calculation we have $n_i = 1, m_i = 1$,
in case of delta Input + Target Calculation: $n_i = 2$, $m_i = 1$.
The state equations transform into:

$$\begin{pmatrix} z_1(k+1) \\ z_2(k+1) \\ \vdots \\ z_M(k+1) \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1M} \\ a_{21} & a_{22} & \dots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MM} \end{pmatrix} \begin{pmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ z_M(k) \end{pmatrix} \tag{3.28}$$

$$+ \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1M} \\ b_{12} & b_{22} & \dots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \dots & b_{MM} \end{pmatrix} \begin{pmatrix} v_1(k) \\ v_2(k) \\ \vdots \\ v_M(k) \end{pmatrix} \tag{3.29}$$

The related model predictive cost function without terminal constraint reads as:

$$V_N(z(k), v(k)) = \tfrac{1}{2} \sum_{i=1}^{M} \sum_{\tau=1}^{N-1} \left(z_i(\tau)^T Q_i z_i(k) + v_i(k)^T R_i v_i(k) \right) \tag{3.30}$$

$$= \tfrac{1}{2} \sum_{i=1}^{M} \left(z_i^T \Omega_i z_i + v_i^T \Psi_i v_i \right) \tag{3.31}$$

where

$$z_i = \left(z_i(0)^T, \quad z_i(1)^T, \quad \dots \quad z_i(N-1)^T \right)^T \tag{3.32}$$

$$v_i = \left(v_i(0)^T, \quad v_i(1)^T, \quad \dots, \quad v_i(N-1)^T \right)^T \tag{3.33}$$

$$\Omega_i = \text{blkdiag}(Q_i, Q_i, \dots, Q_i) \tag{3.34}$$

$$\Psi_i = \text{blkdiag}(R_i, R_i, \dots, R_i) \tag{3.35}$$

"blkdiag$(Q_i, .., Q_i, R_i, ..., R_i)$" means a blockdiagonal matrix with the blocks Q_i and R_i. We start the reformulation with defining the combined vector:

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_M \end{pmatrix}, y_i = \begin{pmatrix} z_i(0) \\ z_i(1) \\ \dots \\ z_i(N-1) \\ v_i(0) \\ v_i(0) \\ \dots \\ v_i(N-1) \end{pmatrix} \qquad \forall i = 1, ..., M \tag{3.36}$$

The matrices H, A, B, C, D are build such that they equal (3.28) and (3.30):

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix}^T \begin{pmatrix} H_{11} & 0 & \dots & 0 \\ 0 & H_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & H_{MM} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} = V_N(y), \tag{3.37}$$

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1M} \\ A_{21} & A_{22} & \dots & A_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \dots & A_{MM} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1M} \\ B_{21} & B_{22} & \dots & B_{MM} \\ \vdots & \vdots & \ddots & \vdots \\ B_{M1} & B_{M2} & \dots & B_{MM} \end{pmatrix}, \tag{3.38}$$

$$\begin{pmatrix} C_{11} & 0 & \dots & 0 \\ 0 & C_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{MM} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} = \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_M \end{pmatrix} \tag{3.39}$$

,

$$H_{ii} = \text{blkdiag}(Q_i, ..., Q_i, R_i, ..., R_i), \tag{3.40}$$

$$C_{ii} = \text{blkdiag}(Cz, ..., Cz, Cv, ..., Cv), \tag{3.41}$$

$$D_i = \left(D_1^T, D_2^T, \dots, D_M^T\right)^T. \tag{3.42}$$

The system model of this thesis has no input and no inequality constraints coupling. This means $\forall i = 1, \dots, M, j = 1, \dots, M$ with $i \ /j \ B_{ij} = 0$, $C_{ij} = 0$. The cost function is decoupled by definition of the weighting matrices. To build $A_{ij}, B_{ii}, C_{ii}, D_i$ we look at the equality constraints/system equations:

$$\begin{aligned} -z_1(1) + a_{11}z_1(0) + a_{12}z_2(0) + b_{11}v_1(0) &= 0 \\ -z_1(2) + a_{11}z_1(1) + a_{12}z_2(1) + b_{11}v_1(1) &= 0 \\ &\vdots \\ -z_1(N-1) + a_{11}z_1(N-2) + a_{12}z_1(N-2) + b_{11}v_1(N-2) &= 0 \\ \hline -z_2(1) + a_{21}z_1(0) + a_{22}z_2(0) + b_{22}v_2(0) &= 0 \\ -z_2(2) + a_{21}z_1(1) + a_{22}z_2(1) + b_{22}v_2(1) &= 0 \\ &\vdots \\ -z_2(N-1) + a_{21}z_1(N-2) + a_{22}z_2(N-2) + b_{22}v_2(N-2) &= 0 \end{aligned}$$

constraints/boxed constraints:

$$\begin{aligned} -z(\tau) &\leq -z_{min} \\ z(\tau) &\leq z_{max} \end{aligned} \tag{3.43}$$

$$-v(\tau) \leq -v_{min}$$
$$v(\tau) \leq v_{max}$$

$$C_z z_i \leq d_z \tag{3.44}$$
$$C_v v_i \leq d_v \tag{3.45}$$

Now we form matrices out of these equations with the vector y as specified.

$$A_{ii} = \left[\begin{array}{cccccc|cccccc} -\mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ a_{ii} & -\mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & b_{ii} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{ii} & -\mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & b_{ii} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & a_{ii} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & b_{ii} & \mathbf{0} \end{array}\right], \tag{3.46}$$

$$A_{ij} = \left[\begin{array}{cccccc|cccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ a_{ij} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & b_{ij} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{ij} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & b_{ij} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & a_{ij} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & b_{ij} & \mathbf{0} \end{array}\right] \tag{3.47}$$

,

$$B_{ii} = \left[\begin{array}{c} -\mathbf{Ix} \\ \mathbf{0x} \\ \vdots \\ \mathbf{0x} \end{array}\right], B_{ij} = \left[\begin{array}{c} \mathbf{0x} \\ \mathbf{0x} \\ \vdots \\ \mathbf{0x} \end{array}\right] \tag{3.48}$$

Where Ix = $[1; 1; ....1] \in R^{n_i}$, 0x = $[0; 0; ...; 0] \in R^{n_i}$.

3.2.1 Dual Decomposition

For two rooms the optimization problem is:

$$\begin{aligned} \underset{y}{\text{minimize}} \ & \tfrac{1}{2} y^T H y \\ & Ay = Bz_0 \\ & Cy \leq D \end{aligned} \tag{3.49}$$

And as problem with relaxed constraints (Lagrange):

$$L(y, \lambda, \mu) = \tfrac{1}{2} y^T H y + \lambda(Ay - Bz_0) + \mu(Cy - D) \tag{3.50}$$
$$L(y_1, y_2, \lambda_1, \lambda_2, \mu_1, \mu_2) = \tfrac{1}{2} y_1^T H_1 y_1 + \lambda_1(A_{11} y_1 + A_{12} y_2 - B_{11} z_{01}) \tag{3.51}$$

$$+ \mu_1(C_{11}y_1 - D_1) \tag{3.52}$$
$$+ \tfrac{1}{2}y_2^T H_2 y_1 + \lambda_2(A_{22}y_2 + A_{21}y_1 - B_{22}z_{02}) + \tag{3.53}$$
$$\mu_2(C_{22}y_2 - D_2) \tag{3.54}$$

We define the dual function $G(\lambda, \mu)$:

$$G(\lambda, \mu) = \inf_{y_1, y_2} L(y, \lambda, \mu) \tag{3.55}$$

And the dual Problem:

$$max_{\lambda,\mu} G(\lambda, \mu) = \max_{\lambda,\mu} \inf_{y_1,y_2} L(y, \lambda, \mu) \tag{3.56}$$

Thus we can use the dual decomposition to get distributed problems which can be solved using independent computational units. First we declare y_1 and y_2 as public variables and name them different for each system.
Computational unit 1: y_{1a}, y_{2a},
computational unit 2: y_{1b}, y_{2b} with the new consistency constraint: $y_{1a} = y_{1b}$; $y_{2a} = y_{2b}$. This way we can separate the dual optimization problem.The Lagrange function is:

$$\begin{aligned} L(y_{1a}, y_{2a}, y_{2b}, y_{1b}, \lambda_1, \lambda_2, \mu_1, \mu_2) = {} & \tfrac{1}{2}y_{1a}^T H_1 y_{1a} + \lambda_1(A_{11}y_{1a} + A_{12}y_{2a} - B_{11}z_{01}) \\ & + \mu_1(C_{11}y_{1a} - D_1) \\ & + \tfrac{1}{2}y_{2b}^T H_2 y_{2b} + \lambda_2(A_{22}y_{2b} + A_{21}y_{1b} - B_{22}z_{02}) \\ & + \mu_2(C_{22}y_{2b} - D_2) \end{aligned} \tag{3.57}$$

The new constraints,which we call the "consistency constraint" are: :

$$y_{1a} = y_{1b} \tag{3.58}$$
$$y_{2a} = y_{2b} \tag{3.59}$$

meaning that the y_{1a} and y_{1b} should have the same value since they stand for the same state y_1.
y_{2a} should have the same value as y_{2b}. Due to these new definition of public variables, the function $L(y_{1a}, y_{21}, y_{2b}, y_{1b}, \lambda_1, \lambda_2, \mu_1, \mu_2)$ becomes completely separable:

$$\begin{aligned} L_1(y_{1a}, y_{2a}, \lambda_1, \mu_1,) = {} & \tfrac{1}{2}y_{1a}^T H_1 y_{1a} + \lambda_1^T(A_{11}y_{1a} + A_{12}y_{2a} - B_{11}z_{01}) + \\ & \mu_1^T(C_{11}y_{1a} - D_1) \end{aligned} \tag{3.60}$$

$$\begin{aligned} L_2(y_{1b}, y_{2b}, \lambda_2, \mu_2) = {} & \tfrac{1}{2}y_{2b}^T H_2 y_{2b} + \lambda_2^T(A_{22}y_{2b} + A_{21}y_{1b} - B_{22}z_{02}) + \\ & \mu_2^T(C_{22}y_{2b} - D_2) \end{aligned} \tag{3.61}$$

We can use the gradient descent method to solve this problem. We use it in two steps. First we use it to solve the "outer" optimization, which tries to fulfil the consistency constraints. Again we construct a dual function:

$$g(\nu_1, \nu_2) = L_1(y_{1a}, y_{2a}, \lambda_1, \mu_1,) + L_2(y_{1b}, y_{2b}, \lambda_2, \mu_2) \tag{3.62}$$
$$+ \nu_1^T(y_{1a} - y_{1b}) + \nu_2^T(y_{2a} - y_{2b}) \tag{3.63}$$
$$= g_1(\nu_1, \nu_2) + g_2(\nu_1, \nu_2) \tag{3.64}$$

The goal is now to solve the two problems:

$$g_1(\nu_1, \nu_2) = \inf_{y_{1a}, y_{2a}} L_1(y_{1a}, y_{2a}, \lambda_1, \mu_1,) + \nu_1^T y_{1a} + \nu_2^T y_{2a} \tag{3.65}$$
$$g_2(\nu_1, \nu_2) = \inf_{y_{1b}, y_{2b}} L_2(y_{1a}, y_{2b}, \lambda_2, \mu_2) - \nu_1^T y_{1b} - \nu_2^T y_{2b} \tag{3.66}$$

This gives us the following algorithm:
repeat

1. solve distributed optimization in parallel, constant ν_1, ν_2:

   computational unit 1 solves $g_1(\nu_1, \nu_2)$ to get minimizing y_{1a}, y_{2a}

   computational unit2 solves $g_2(\nu_1, \nu_2)$ to get minimizing y_{1b}, y_{2b}

2. exchange of the results between the computational units

3. update the dual variables ν_1, ν_2 in parallel, constant $y_{1a}, y_{2a}, y_{1b}, y_{2b}$:

   computational unit1 calculates $\nu_1 = \nu_1 + \alpha_k(y_{1a} - y_{1b})$

   computational unit2 calculates $\nu_2 = \nu_2 + \alpha_k(y_{2a} - y_{2b})$

4. exchange of the results between the computational units

5. go back to 1.

All variables can be initialized as zero-vectors of corresponding dimension. The gradient method is a steepest descent method, so the iterative solution to $g_1(\nu_1, \nu_2), g_2(\nu_1, \nu_2)$ can be obtained by the steepest descent method. We want to solve the minimization of $f(y)$ using this iterative formula:
repeat

$$yk = yk - \alpha_k(\nabla f(y)) \tag{3.67}$$

for a maximization respectively:

$$yk = yk + \alpha_k(\nabla f(y)) \tag{3.68}$$

This means, at each iteration, we use $-\nabla f(y)$ as the direction in which the minimum of f(y) lies and take a weighted step into this direction. The weight α_k

should not be to large, otherwise the method will not converge. How to choose α_k is described in the background chapter of [Gis12] .
For now, we use $\alpha_k = 0.02\frac{k-1}{k+2}$. The constant 0.02 is one we got through some numerical experiments. For this α_k the dual decomposition iterations converge to the optimal value. To solve the optimizations $g_1(\nu_1, \nu_2), g_2(\nu_1, \nu_2)$ we have to **repeat** the following iterations:
Computational unit 1 - optimization variables Iterations:

$$\begin{aligned} y_{1a} &= y_{1a} - \alpha_k(\nabla_{y_{1a}} g_1(\nu_1, \nu_2)) \\ y_{2a} &= y_{2a} - \alpha_k(\nabla_{y_{2a}} g_1(\nu_1, \nu_2)) \end{aligned} \tag{3.69}$$

computational unit 1 - dual variable updates

$$\begin{aligned} \lambda_1 &= \lambda_1 + \alpha_k(\nabla_{\lambda_1} g_1(\nu_1, \nu_2)) \\ \mu_1 &= \mu_1 + \alpha_k(\nabla_{\mu_1} g_1(\nu_1, \nu_2)) \\ \nu_1 &= \nu_1 + \alpha_k(\nabla_{nu_1} g_1(\nu_1, \nu_2)) \end{aligned} \tag{3.70}$$

computational unit 2 - optimization variable Iterations

$$\begin{aligned} y_{1b} &= y_{1b} - \alpha_k(\nabla_{y_{1b}} g_2(\nu_1, \nu_2)) \\ y_{2b} &= y_{2b} - \alpha_k(\nabla_{y_{2b}} g_2(\nu_1, \nu_2)) \end{aligned} \tag{3.71}$$

computational unit 2 - dual variable updates

$$\begin{aligned} \lambda_2 &= \lambda_2 + \alpha_k(\nabla_{\lambda_2} g_2(\nu_1, \nu_2)) \\ \mu_1 &= \mu_1 + \alpha_k(\nabla_{\mu_1} g_2(\nu_1, \nu_2)) \\ \nu 2 &= \nu_2 + \alpha_k(\nabla_{nu_2} g_2(\nu_1, \nu_2)) \end{aligned} \tag{3.72}$$

The respective gradients are for

computational unit1

$$\begin{aligned} \nabla_{y_{1a}} &= H_1 y_{1a} + A_{11}^T \lambda_1 + C_{11}^T \mu_1 + \nu_1 \\ \nabla_{y_{2a}} &= A_{12}^T + \nu_2 \\ \nabla_{\lambda_1} &= A_{11} y_{1a} + A_{12} y_{2a} - B_{11} z_{01} \\ \nabla_{\mu_1} &= C_{11} y_{1a} - D_1 \\ \nabla_{\nu_1} &= y_{1a} - y_{2a} \end{aligned}$$

computational unit1

$$\begin{aligned} \nabla_{y_{1b}} &= A_{21}^T \lambda_2 - \nu_2 \\ \nabla_{y_{2b}} &= H_2 y_{2a} + A_{22}^T \lambda_2 + C_{22}^T \mu_2 - \nu_1 \\ \nabla_{\lambda_2} &= A_{21} y_{1b} + A_{22} y_{2b} - B_{22} z_{02} \\ \nabla_{\mu_2} &= C_{11} y_{1a} - D_1 \\ \nabla_{\nu_2} &= y_{1b} - y_{2b}. \end{aligned}$$

The MPC procedure is:

1. measure new state of z(k)
2. start optimization with prediction form of state equations, inequalities, cost functions
   repeat for Δk Iterations

   computational unit 1 iterates (3.69)

   computational unit 1 iterates (3.71)

   communication of current iteration-values of $y_{1a}, y_{2a}, y_{1b}, y_{2b}$

   computational unit 2 iterates (3.70)

   computational unit 2 iterates (3.72)

   communication of current iterations-values of $\lambda_1, \lambda 2, \mu_1, \mu_2$
3. get $u_1(0)$ and $u_2(0)$ from y_1, y_2
4. implement $u_1(k), u_2(k)$
5. go back to 1.

This was used to calculate the reference value in the distributed case.

3.2.2 Accelerated Gradient Method

The accelerated gradient method uses not only the current gradient but a linear combination of the previous and the new gradient as descent direction. Since the gradients are independent of each other, we do not need dual decomposition to seperate the two subproblems. The dual function is

$$V_N(\lambda, \mu, y) = \tfrac{1}{2} y^T H y^T + \lambda^T (Ay - B) + \mu^T (Cy - D) \tag{3.73}$$

We calculate the minimizing y of the dual function through derivation:

$$0 = Hy + A^T y + C^T y \tag{3.74}$$

$$y = -H^{-1}(A^T y + C^T y) \tag{3.75}$$

This yield the following Iterations:

$$\begin{aligned} y &= -H^{-1}(A^T y + C^T y) \\ \overline{y} &= y + \alpha_k (y - y_{past}) \\ \lambda &= \lambda + \alpha_k (\lambda - \lambda_{past}) + \tfrac{1}{L}(A\overline{y} - B) \\ \mu &= \mu + \alpha_k (\mu - \mu_{past}) + \tfrac{1}{L}(C\overline{y} - D) \end{aligned} \tag{3.76}$$

These are used to solve the central Problem:

1. measure the current state vector x(k)
2. start new optimization with $z_0 = x(k)$

   calculate (3.76)
3. get $u_0(k)$ out of y
4. implement $u(k)$
5. go back to 1.

To implement the distributed MPC we seperate (3.76):
computational unit 1:

$$\begin{aligned} y_1 &= -H_1^{-1}(A_{11}^T\lambda_1 + A_{21}^T\lambda_2 + C_{11}^T\mu_1) \\ \overline{y}_1 &= y_1 + \alpha_k(y_1 - y_{1past}) \\ \lambda_1 &= \lambda_1 + \alpha_k(\lambda_1 - \lambda_{1past}) + \tfrac{1}{L}(A_{11}y_1 + A_{12}y_2 - B_{11}z_{01})) \\ \mu_1 &= \mu_1 + \alpha_k(\mu_1 - \mu_{1past}) + \tfrac{1}{L}(C_{11}y_1 - D_1) \end{aligned} \tag{3.77}$$

computational unit 2:

$$\begin{aligned} y_2 &= -H_2^{-1}(A_{12}^T\lambda_1 + A_{22}^T\lambda_2 + C_{22}^T\mu_2) \\ \overline{y}_2 &= y_2 + \alpha_k(y_2 - y_{2past}) \\ \lambda_2 &= \lambda_2 + \alpha_k(\lambda_2 - \lambda_{2past}) + \tfrac{1}{L}(A_{21}y_1 + A_{22}y_2 - B_{22}z_{02}) \\ \mu_2 &= \mu_2 + \alpha_k(\mu_2 - \mu_{2past}) + \tfrac{1}{L}(C_{22}y_2 - D_2) \end{aligned} \tag{3.78}$$

α_k is the step size:

$$\alpha_k = \tfrac{k-1}{k+2} \tag{3.79}$$

and $\frac{1}{L}$ with L the Liptschitz constant is uesd as second stepsize. An upper bound on the Lipschitz constant is calculated as follows:

$$L = \|[A^T, C^T]^T H [A^T, C^T]\| \tag{3.80}$$

The distributed model predictive control repeats the following steps:

1. measure $z_0 = x(k)$
2. solve the Iterations

   computational unit 1 iterates (3.77)

   computational unit 2 iterates (3.78)
3. get $u_1(0), u_2(0)$ out of y_1, y_2
4. implemenet u_1, u_2
5. go back to 1.

4 System Model for two Rooms

4.1 Heat Flow Models

The flow of heat that occurs to compensate the difference in temperature between two places, for example two rooms which are separated by a wall, is calculated as:

$$q = \frac{T_2 - T_1}{R_{12}}$$

T_1: temperature of one side, e.g room 1
T_2: temperature of the other side, e.g room 2
R_{12}: thermal resistance between the two sides, e.g a wall

The damping of heat flow through the wall is described by the thermal resistance "R".

$$R = \frac{L}{Ak}$$

L: length of piece of wall
A: area of piece of wall
k: coefficient of thermal heat conductivity

The change of heat energy in the system is described as:

$$q_1 - q_2 = mc\frac{T}{dt} \tag{4.1}$$

c = specific thermal capacity
m = mass of heat storage
T = temperature
q = heat flow

The following picture 4.1 shows how
a) thermal resistances in series would look like. The example is to walls behind each other. One of concrete and one of glas wool. The heat flow goes through both of them. b) shows how two thermal resistances in parallel would look like.

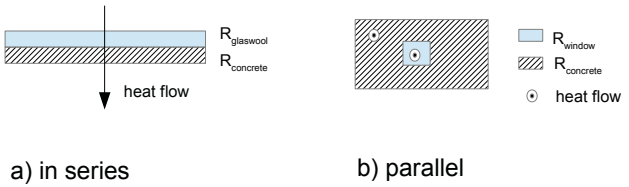

Figure 4.1: example: thermal resistances

The example is a wall and a window. The heat flow divides. One part goes through the window, another part through the wall.
a) Thermal resistances in series are calculated as follows:

$$R_g = R_1 + R_2 \tag{4.2}$$

b) Thermal resistances in parallel are calculated as follows:

$$\frac{1}{R_g} = \frac{1}{R_1} + \frac{1}{R_2} \tag{4.3}$$

That is all we need to create our first room model:
The house is shown in 4.2 The differential equations for this example are:

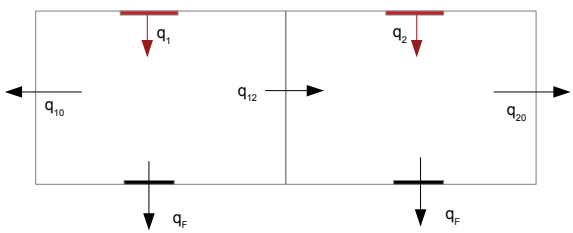

Figure 4.2: example to show how to get a system model of the room model

$$q_1 - q_{12} - q_{10} - q_F = C_1 \frac{dT_1}{dt} \tag{4.4}$$

$$q_2 + q_{12} - q_{20} - q_F = C_2 \frac{dT_2}{dt} \tag{4.5}$$

We define the following heat flows:

$$q_1 = \text{heater room 1}$$
$$q_2 = \text{heater room 2}$$
$$q_{10} = \frac{T_1 - T_0}{R10}$$
$$q_{20} = \frac{T_2 - T_0}{R20}$$
$$q_{12} = \frac{T_1 - T_2}{R12}$$

and insert them into differential equations (4.4), (4.5):

$$q1 - \frac{T_1 - T_0}{R_{10}} - \frac{T_1 - T_2}{R_{12}} = C_1 \frac{dT_1}{dt} \tag{4.6}$$
$$q2 - \frac{T_2 - T_0}{R_{20}} + \frac{T_1 - T_2}{R_{12}} = C_2 \frac{dT_2}{dt} \tag{4.7}$$

This is the differential equation describing the dynamics of the model shown in 4.2.

4.2 The Two-Room Model

The two room model from before will be expanded by a glass wool isolation as shown in picture 4.3.

Parameter

First we need to specify some parameters [www]:

Parameter	glass wool	concrete	window glass	air
density [kg/m^3] ρ	2.5	1700	2500	1.28
specific thermal conductivity k [W/m K]	0.04	0.7	0.96	0.024
specific thermal capacity c [J/kg K]	760	660	700	1010

There after the materials will have the indices:
glass wool will have the indices: dA_{wo}, k_{wo}, c_{wo}
concrete will have the indices: $dA_{con}, k_{con}, c_{con}$
window glass will have the indices: $dA_{win}, k_{win}, c_{win}$
air will have the indices: $dA_{air}, k_{air}, C_{air}$

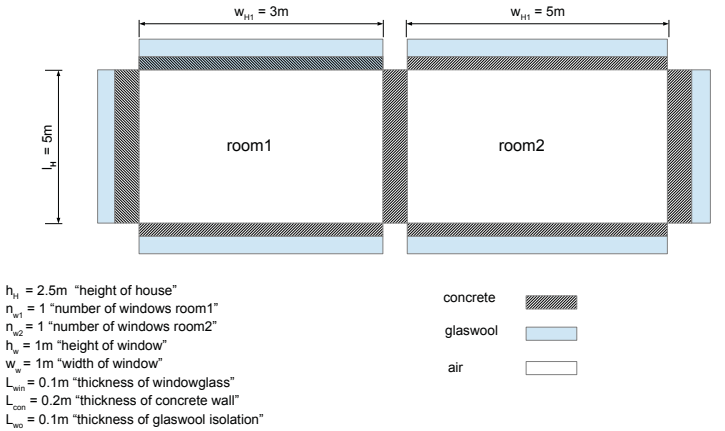

Figure 4.3: house model with concrete walls and glaswool isolation

Areas through which Heat flows

$$\textbf{room1 to outside: } Awall_{o1} = l_H h_H + 2w_{H1} h_H - n_{w1} h_W w_W \tag{4.8}$$

$$\textbf{room2 to outside: } Awall_{o2} = l_H h_H + 2w_{H2} h_H - n_{w2} h_W w_W \tag{4.9}$$

$$\textbf{room1 to room2: } Awall_{12} = l_H h_H \tag{4.10}$$

$$\textbf{window1: } Awin_1 = n_{W1} hW wW \tag{4.11}$$

$$\textbf{window2: } Awin_2 = n_{W2} hW wW \tag{4.12}$$

Masses that store Heat

$$\textbf{concrete mass room 1: } m_{con1} = (2L_{con} h_H w_{H1} + 2L_{con} h_H l_H) dA_{con} \tag{4.13}$$

$$\textbf{concrete mass room 2: } m_{con2} = (2L_{con} h_H w_{H2} + 2L_{con} h_H l_H) dA_{con} \tag{4.14}$$

$$\textbf{glass wool mass room 1: } m_{wo1} = (2L_{wo} h_H w_{H1} + L_{wo} h_H l_H) dA_{wo} \tag{4.15}$$

$$\textbf{glass wool mass room 2: } m_{wo2} = (2L_{wo} h_H w_{H2} + L_{wo} h_H l_H) dA_{wo} \tag{4.16}$$

$$\textbf{air mass room 1: } M_1 = (l_h w_{H1} h_H) dA_{air} \tag{4.17}$$

$$\textbf{air mass room 1: } M_2 = (l_H w_{H2} h_h) dA_{air} \tag{4.18}$$

Heat Capacities

The heat capacities consist of the heat capacity of the air inside each room and the heat capacit fo the surrounding concrete walls around each room and the heat capacity of the glass wool isolation around each room.
The heat capacity is calculated as: $C = mc$, where c is specific heat capacity and m is the mass of the element. The heat flow splits into single heat flows "going through" each of the masses. This means the masses behave as if they are parallel. Based on the analogy to electric circuits we now that parallel capacities are added:

$$\textbf{heat capacity room1: } C_{th1} = M_1 c_{air} + m_{wo1} c_{wo} + m_{con1} c_{con} \tag{4.19}$$
$$\textbf{heat capacity room2: } C_{th2} = M_2 c_{air} + m_{wo2} c_{wo} + m_{con2} c_{con} \tag{4.20}$$

Thermal Resistances

The thermal resistance $R_{o1}$for heat flow from room 1 to outside splits up into

$$R_{con1} = \frac{L_{con}}{k_{con} A_{wall\_o1}} \tag{4.21}$$
$$R_{wo1} = \frac{L_{wo}}{k_{wo} A_{wall\_o1}} \tag{4.22}$$
$$R_{win1} = \frac{L_{win}}{k_{win} A_{win1}}, \tag{4.23}$$
$$\tag{4.24}$$

the thermal resistance R_{o2} for heat flow from room 2 to outside:

$$R_{con2} = \frac{L_{con}}{k_{con} A_{wall\_o2}} \tag{4.25}$$
$$R_{wo2} = \frac{L_{wo}}{k_{wo} A_{wall\_o2}} \tag{4.26}$$
$$R_{win2} = \frac{L_{win}}{k_{win} A_{win2}}. \tag{4.27}$$
$$\tag{4.28}$$

The heat flow from room 2 to outside splits ab in the heat flow through the wall(concrete + glass wool) and the window.

$$R_{o1} = (\frac{1}{R_{con1} + R_{wo1}} + \frac{n_{w1}}{R_{win1}})^{-1} \tag{4.29}$$

$$R_{o2} = (\frac{1}{R_{con2} + R_{wo2}} + \frac{n_{w2}}{R_{win2}})^{-1} \tag{4.30}$$

Differential Equation and State Space Model

The model shown in 4.4 describes the heat flows of 4.3. This model 4.3is used to

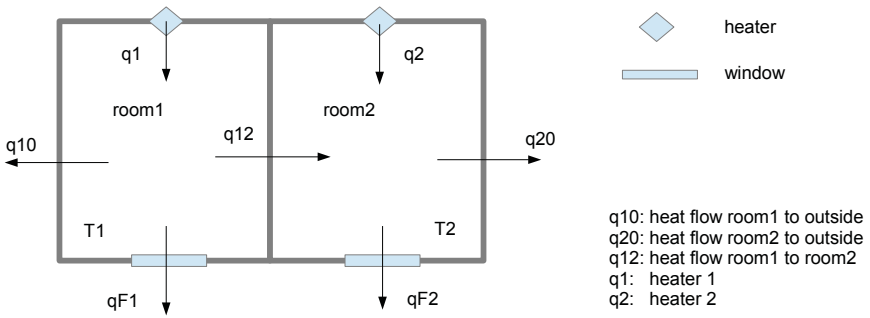

Figure 4.4: two room model

create the differential equations.

$$C_{th1}\frac{dT_1}{dt} = q_1 - q_{12} - q_{10} \tag{4.31}$$

$$C_{th1}\frac{dT_1}{dt} = q_1 - \frac{T_1 - T_2}{R_{12}} - \frac{T_1 - T_0}{R_{10}} \tag{4.32}$$

$$C_{th1}\frac{dT_1}{dt} = q_1 - \frac{R_{12} + R_{10}}{R_{12}R_{10}}T_1 + \frac{1}{R_{12}}T_2 + \frac{1}{R_{10}}T_0 \tag{4.33}$$

$$C_{th2}\frac{dT_1}{dt} = q2 + q_{12} - 1_{20} \tag{4.34}$$

$$C_{th2}\frac{dT_1}{dt} = q2 + \frac{T_1 - T_2}{R_{12}} - \frac{T_2 - T_0}{R_{20}} \tag{4.35}$$

$$C_{th2}\frac{dT_1}{dt} = q2 + \frac{R_{12} - R_{20}}{R_{12}R_{20}}T_2 + \frac{1}{R_{12}}T_1 - \frac{1}{R_{20}}T_0 \tag{4.36}$$

which can be written as:

$$a_{11}x_1 + a_{12}x_2 + b_{11}u_1 + b_{w11}d_1 = \dot{x}_1 \tag{4.37}$$

$$a_{21}x_1 + a_{21}x_2 + b_{22}u_2 + b_{w22}d_2 = \dot{x}_2 \tag{4.38}$$
$$ax + bu + b_w w = \dot{x} \tag{4.39}$$

where x_1, x_2 are the states corresponding to the temperatures T_1, T_2 and u_1, u_2 are the inputs corresponding to the heat flows of the heaters q_1, q_2.

The coefficients are:

$$a_{11} = -\frac{R_{12} + R_{10}}{R_{12}R_{10}C_{th1}} \qquad a_{12} = \frac{1}{R_{12}C_{th1}} \qquad a_{21} = \frac{1}{R_{12}C_{th2}} \qquad a_{22} = \frac{R_{12} - R_{20}}{R_{12}R_{20}C_{th2}} \tag{4.40}$$

$$b_{11} = \frac{1}{C_{th1}} \qquad b_{12} = 0 \qquad b_{21} = 0 \qquad b_{22} = \frac{1}{C_{th2}} \tag{4.41}$$

$$b_{w11} = \frac{1}{C_{th1}R_{10}} \qquad b_{w12} = 0 \qquad b_{w21} = 0 \qquad b_{w22} = \frac{1}{C_{th2}R_{20}} \tag{4.42}$$

4.3 Discretization

To discretize the system we interpret the disturbance through the outside temperature T_0 as an input. The following matrices will be used:

$$u_{\text{hilf}} = \begin{pmatrix} u_1 \\ u_2 \\ T_0 \end{pmatrix}, b_{\text{hilf}} = \begin{pmatrix} b_{11} & b_{12} & b_{w11} \\ b_{21} & b_{22} & b_{w22} \end{pmatrix} \tag{4.43}$$

with the system: $\dot{x} = ax + b_{\text{hilf}}u_{\text{hilf}}$ The matlab function "c2d" can be used to discretize this system. But this may destroy the separable structure of our system and result in b_{12}, b_{21} non zero. Thus we use the approximation of zero-order-hold discretization, which is explained in the following:
Say we have the continuous system: $a_c x + b_c u = \dot{x}$ and $c_c x = yc$. The solution to this differential equation is:

$$x(t) = \exp^{a_c(t-t_k)} x(t_k) + \int_{t_k}^{t} \exp^{a_c(t-s))} b_c u(s) ds \tag{4.44}$$

Since we are interested in the solution to the discretized system we consider a discrete input of the form:

$$u(t) = u(t_k) \qquad \forall t_k \leq t \leq t_{k+1} \tag{4.45}$$

This means our input has the same value between the time steps t_k and t_{k+1}. We insert this $u(t)$ into (4.44).

$$x(t_{k+1}) = \exp^{a_c(t_{k+1}-t_k)}\, x(t_k) + \int_{t_k}^{t_{k+1}} \exp^{a_c(t_{k+1}-z)}\, b_c u(t_k) dz \tag{4.46}$$

and define: $h_k = t_{k+1} - t_k$ to get:

$$x(t_{k+1}) = \exp^{a_c h_k}\, x(t_k) + \int_{t_k}^{t_{k+1}} \exp^{a_c(t_{k+1}-z)}\, dz b_c u(t_k) \tag{4.47}$$

Via the substitution: $s = t_{k+1} - z$ we get $\frac{ds}{dz} = -1$so, it follows $dz = -ds$. With the new integration limits: $\underline{s} = t_{k+1} - t_k = h_k$ and $\overline{s} = 0$:

$$x(t_{k+1}) = \exp^{a_c h_k}\, x(t_k) - \int_{h_k}^{0} \exp^{a_c s}\, ds\ b_c u(t_k) \tag{4.48}$$

$$x(t_{k+1}) = \exp^{a_c h_k}\, x(t_k) + \int_{0}^{h_k} \exp^{a_c s}\, ds\ b_c u(tk) \tag{4.49}$$

$$x(t_{k+1}) = a x(t_k) + b u(t_k) \tag{4.50}$$

The discrete system matrices are:

$$a = \exp^{a_c h_k} \qquad b = \int_0^{h_k} \exp^{a_c s}\, ds b_c u(t_k) \tag{4.51}$$

$$c = c_c \qquad d = d_c \tag{4.52}$$

The Approximation uses the first two elements of the Taylor series expansion:

$$a = \exp^{a_c h_k} \qquad = I + a_c ts \tag{4.53}$$

$$b = \int_0^{h_k} \exp^{a_c s}\, ds\ b_c = a_c^{-1}(\exp^{a_c h_k} - I)\ \ b_c = a_c^{-1}(I + a_c ts - I)\ \ b_c \quad = ts\ b_c \tag{4.54}$$

Afterwards isolate the disturbance term from the input and have the discretized system:

$$x(k+1) = a x(k) + b_{\text{hilf}} u_{\text{hilf}}(k) \tag{4.55}$$

$$a = I + a_c ts \tag{4.56}$$

$$b_{\text{hilf}} = ts b_c \tag{4.57}$$

$$c = c_c \tag{4.58}$$

$$d = d_c \tag{4.59}$$

$$b = b_{\text{hilf}}[:, 1:2] \tag{4.60}$$

$$b_w = b_{\text{hilf}}[:, 3] \tag{4.61}$$

4.4 Sampling Time

To calculate a useful sampling time we need to measure the rise time. We solve this with matlab:
sys = ss(a_c,b_c,c_c,0)
step(sys)
We pick the time t_{10} where the step response is 10 % and the time t_{90} where the step response is 90 % of the maximum value (stationary end state). We will use the smaller t_A.
The sampling time should fulfill $ts \leq 10t_A$.

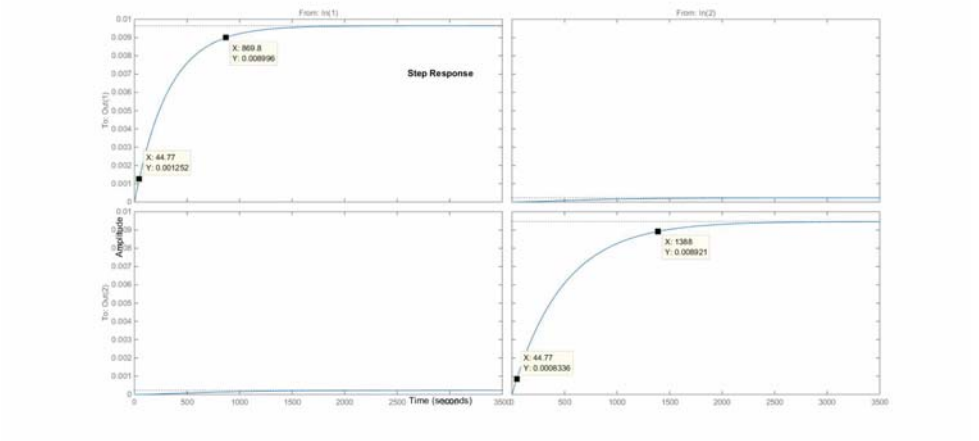

Figure 4.5: step response of the built system

We calculate the sampling time for the system model (4.39) using the reformulation as (4.43). The points in the upper left step response of 4.5 are at the timepoints: $t_{10} = 44.8s$
$t_{90} = 870s$
The Rise time is calculated as: $t_A = t_{90} - t_{10} = 825.1s$
The sampling time is calculated as: $ts = \frac{t_A}{20} = 41.26s$

5 Reference Tracking

The goal is to use Model Predictive Control (MPC) to regulate the room temperature to a specified reference temperature in each room. The used algorithm is designed for a reference value zero. To use it we reformulate the "reference Tracking problem" into a regulation to zero. We will use two ways:
target Calculation: The reference value is subtracted from the state and input variables such that we regulate $\tilde{x} = x - x_{ss}$ and $\tilde{u} = u - u_{ss}$ to zero.
Delta Input: The delta Input equation is $u(k) = u(k-1) - \Delta u(k)$ and this equation is joined with the state equations. The new input is $\Delta u(k)$ and $u(k)$ becomes a state.

We want to do reference tracking for central and distributed model predictive control. For that we need some reformulations. We will do reformulations into a "Target Calculation" and "delta Input formulation" for the central and distributed case.
central case: The central case is formulated as a quadratic program and solved with matlab. Central means, that only "one MPC" is used. This MPC works central to calculate all values for all rooms. For a large system (e.g. a huge building with many rooms) the matrix computations become very time consuming and use a lot of computational capacity.
distributed case: To avoid those time consuming computations we use one MPC for each room. With dual decomposition the calculations can be separated into optimization problems only involving state and input variables of one system (room).

5.1 Target Calculation

5.1.1 Central Case

The Problem formulation, with the system equations (4.39) of chapter 3 is:

$$\begin{aligned} &\underset{\overline{u},\overline{x}}{\text{minimize}}\, V_N(\overline{x}(k),\overline{u}(k)) \\ &\text{subject to } \overline{x}(k+1) = a\overline{x}(k) + b\overline{u}(k) + b_w w(k) \\ &\overline{y}(k) = c\overline{x}(k) \\ &\overline{u}_{\min} \leq \overline{u}(k) \leq \overline{u}_{\max} \\ &\overline{x}_{\min} \leq \overline{x}(k) \leq \overline{x}_{\max} \end{aligned} \tag{5.1}$$

with the matrices:

$$a = \begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix}, b = \begin{pmatrix} b11 & 0 \\ 0 & b22 \end{pmatrix}, bw = \begin{pmatrix} bw11 & 0 \\ 0 & bw22 \end{pmatrix}, c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5.2}$$

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}, P = \begin{pmatrix} P_1 & 0 \\ 0 & P_2 \end{pmatrix}, R = \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix} \tag{5.3}$$

and the dimensions: $x \in R^n, u \in R^m, w \in R^{nw}, y \in R^p$ with $n = 2, m = 1, nw = 2, p = 2$.
The cost function for reference tracking is defined as:

$$\begin{aligned} V_N(\overline{x}(k),\overline{u}(k)) &= \tfrac{1}{2}(\overline{y}(k+N) - r(k))Q(\overline{y}(k+N) - r(k)) \\ &+ \tfrac{1}{2}(\overline{y}(k+i) - r(k))Q(\overline{y}(k+i) - r(k)) \\ &+ \tfrac{1}{2}\overline{u}^T(k+i)R\overline{u}(k+i)) \\ &= \tfrac{1}{2}(c\overline{x}^T(k+N) - r(k))P(\overline{x}(k+N) - r(k)) \\ &+ \tfrac{1}{2}\sum_{i=0}^{N-1}(c\overline{x}^T(k+i) - r(k))Q(c\overline{x}(k+i) - r(k)) \\ &+ \overline{u}^T(k+i)R\overline{u}(k+i). \end{aligned} \tag{5.4}$$

where $r(k)$ is the reference value for $y(k)$ that we want to reach.
The reference value for the output is $y_{ss} = x_{ss} = r = (r_1^T, r_2^T)^T$. We use the difference between current state/temperature $\overline{x}$ and reference temperature called "state deviation" $\tilde{x} = \overline{x} - xss$ as new variable which should be regulated to zero.

$$\overline{x}(k+1) = a\overline{x}(k) + b\overline{u}(k) + b_w w(k) \tag{5.5}$$

$$xss = ax_{ss} + bu_{ss} + b_w w \tag{5.6}$$

Since our output is the room temperature and we have no disturbance through state measurement our output matrix c is the identity matrix. So $x_{ss} = r$ and u_{ss} is the input needed to hold the state/temperature at the value x_{ss}.

$$\tilde{x}(k+1) = \overline{x}(k+1) - x_{ss}$$

$$\begin{aligned}
&= a\overline{x}(k) + bu(k) + b_w w(k) - ax_{ss} - bu_{ss} - b_w w(k) \\
&= a\overline{x}(k) - ax_{ss} + bu(k) - bu_{ss} \\
&= a\tilde{x}(k) + b\tilde{u}(k)
\end{aligned} \tag{5.7}$$

Important: disturbance is cancelled out!
Since we now regulate the state deviation to zero, the cost function looks like:

$$\begin{aligned}
V_N(x(k), u(k)) = &(\tilde{x}^T(k+N))P(\tilde{x}(k+N)) \\
&+ \sum_{i=0}^{N-1} (\tilde{x}^T(k+i))Q(\tilde{x}(k+i)) + \tilde{u}^T(k+i)R\tilde{u}(k+i)
\end{aligned} \tag{5.8}$$

The next step is to construct a central prediction model for $\tilde{x}(k+1) = a\tilde{x}(k) + b\tilde{u}(k)$.

$$\begin{aligned}
\tilde{x}(k+1) &= a\tilde{x}(k) + b\tilde{u}(k) \\
\tilde{x}(k+2) &= a\tilde{x}(k+1) + b\tilde{u}(k+1) \\
&= a^2\tilde{x}(k) + ab\tilde{u}(k) + b\tilde{u}(k+1) \\
\tilde{x}(k+3) &= a\tilde{x}(k+2) + b\tilde{u}(k+2) \\
&= a^3\tilde{x}(k) + a^2b\tilde{u}(k) + ab\tilde{u}(k+1) + b\tilde{u}(k+2) \\
\tilde{x}(k+4) &= a^4\tilde{x}(k) + a^3b\tilde{u}(k) + a^2b\tilde{u}(k+1) + ab\tilde{u}(k+2) + b\tilde{u}(k+3)
\end{aligned}$$

In matrix form:

$$\tilde{X}(k) = \Phi\tilde{x}(k) + \Gamma\tilde{U}(k) \tag{5.9}$$

$$\Phi = \begin{pmatrix} a \\ a^2 \\ a^3 \\ a^4 \end{pmatrix}, \Gamma = \begin{pmatrix} b & 0 & 0 & 0 \\ ab & b & 0 & 0 \\ a^2b & ab & b & 0 \\ a^3 & a^2b & ab & b \end{pmatrix}, \tilde{X}(k) = \begin{pmatrix} \tilde{x}(k+1) \\ \tilde{x}(k+2) \\ \tilde{x}(k+3) \\ \tilde{x}(k+4) \end{pmatrix}, \tilde{U}(k) = \begin{pmatrix} \tilde{u}(k+0) \\ \tilde{u}(k+1) \\ \tilde{u}(k+2) \\ \tilde{u}(k+3) \end{pmatrix} \tag{5.10}$$

The cost function becomes:

$$\begin{aligned}
V_N(\tilde{x}(k), \tilde{u}(k)) = &(\tilde{x}^T(k+N))P(\tilde{x}(k+N)) \\
&+ \sum_{i=0}^{N-1} (\tilde{x}^T(k+i))Q(\tilde{x}(k+i)) + \tilde{u}^T(k+i)R\tilde{u}(k+i)
\end{aligned} \tag{5.11}$$

The matrices have the following properties:
$P = P^T \succeq 0 \in R^{nxn}$,
$Q = Q^T \succeq 0 \in Q^{nxn}$,
$R = R^T \succ 0 \in R^{mxm}$

The sums are written as matrix-vector product:

$$\begin{aligned} V_N(\tilde{X}(k), \tilde{U}(k)) &= (\tilde{x}^T(k))Q(\tilde{x}(k)) \\ &\quad + (\tilde{X}(k))^T\Omega(\tilde{X}(k)) + \tilde{U}^T(k)\Psi\tilde{U}(k) \end{aligned} \tag{5.12}$$

$$\Omega = \begin{pmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & Q & 0 \\ 0 & 0 & 0 & P \end{pmatrix}, \Psi = \begin{pmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{pmatrix}, C = \begin{pmatrix} c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & c \end{pmatrix} \tag{5.13}$$

Due to the properties of Q, R, P we have: $\Omega \succeq 0, \Psi \succeq 0$
To create a cost function that is only minimized over $\tilde{U}(k)$ we will insert $\tilde{X}(k)$ into the cost function. In the following equations (k) is omitted and $\tilde{x} = x$, $\tilde{u} = u$, to simplify the equations. First we look at the term $(CX(k))^T\Omega(CX(k))$:

$$\begin{aligned} (X(k))^T\Omega(X(k)) &= (X^T\Omega X) \\ &= (x^T\Phi^T + U^T\Gamma^T)\Omega(\Phi x + \Gamma U) \\ &= (x^T\Phi^T\Omega\Phi x + U^T\Gamma^T\Omega\Phi x + x^T\Phi^T\Omega\Gamma U + U^T\Gamma^T\Omega\Gamma U) \\ &= x^T\Phi^T\Omega\Phi x + 2U^T\Gamma^T\Omega\Phi x + U^T\Gamma^T\Omega\Gamma U \end{aligned}$$

All terms that are independent of U(k) are collected in one function, $f(x) = x^T\Phi^T\Omega\Phi x + (x)^T Q(x)$ such that the cost function reads as:

$$\begin{aligned} V_N(x, U) &= 2U^T\Gamma^T\Omega\Phi x + U^T\Gamma^T\Omega\Gamma U + U^T\Psi U + f(x) \\ &= U^T(\Gamma^T\Omega\Gamma + \Psi)U + (2x^T\Phi^T\Omega C\Gamma)U + f(x) \\ &= U^T HU + F_x U + f(x) \end{aligned}$$

Therefore the problem formulation is:

$$min_U V_N(x(k), U(k)) = U^T(k)HU(k) + F_x U(k) + f(x) \tag{5.14}$$

Up to now the inequality constraints are missing.The box constraints for each time step are:

$$\tilde{u}_{min} \leq \tilde{u}(k) \leq \tilde{u}_{max} \tag{5.15}$$

$$\tilde{y}_{min} \leq \tilde{y}(k) \leq \tilde{y}_{max} \tag{5.16}$$

The required form is:

$$M(k+i)\tilde{x}(k+i) + E(k+i)\tilde{u}(k+i) \leq b(k+i), \qquad \forall i \in 1, ..., N$$

With the matrices:

$$M(k+i) = \begin{pmatrix} 0_{m+n} \\ 0_{mxn} \\ -c \\ c \end{pmatrix}, E(k+i) = \begin{pmatrix} -I_{mxm} \\ I_{mxm} \\ 0_{pxm} \\ 0_{pxm} \end{pmatrix}, b(k+i) = \begin{pmatrix} -u_{min} \\ u_{max} \\ -y_{min} \\ y_{max} \end{pmatrix}$$

The prediction form is $\overline{D}(k)\tilde{x}(k) + \overline{M}(k)\tilde{X}(k) + \overline{E}(k)\tilde{U}(k) \leq \overline{b}(k)$
For simplicity we say, the terminal constraints are the same as the constraint for other time steps and receive the matrices:

$$\overline{D}(k) = \begin{pmatrix} M(k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \overline{M}(k) = \begin{pmatrix} 0 & 0 & \dots & 0 \\ M(k+1) & 0 & \dots & 0 \\ 0 & M(k+2) & \dots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & M(k+N) \end{pmatrix}, \quad (5.17)$$

$$\overline{E}(k) = \begin{pmatrix} E(k) & 0 & \dots & 0 \\ 0 & E(k+1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E(k+N-1) \end{pmatrix}, \bar{b}(k) = \begin{pmatrix} b(k) \\ b(k+1) \\ b(k+2) \\ \vdots \\ b(k+N) \end{pmatrix}$$

We now insert the prediction form of equality constraints $\tilde{X}(k) = \Phi\tilde{x}(k) + \Gamma\tilde{U}(k)$ into the prediction form of inequality constraints $\overline{D}(k)\tilde{x}(k) + \overline{M}(k)\tilde{X}(k) + \overline{E}\tilde{U}(k) \leq \overline{b}(k)$:

$$\begin{aligned} \overline{D}\tilde{x}(k) + \overline{M}(k)(\Phi\tilde{x}(k) + \Gamma\tilde{U}(k)) + \overline{E}(k)\tilde{U}(k) &\leq \overline{b}(k) \\ (\overline{D}(k) + \overline{M}(k)\Phi)\tilde{x}(k) + (\overline{M}(k)\Gamma + \overline{E}(k))\tilde{U}(k) &\leq \overline{b}(k) \\ (\overline{M}(k)\Gamma + \overline{E}(k))\tilde{U}(k) &\leq \overline{b}(k) + (-\overline{D}(k) - \overline{M}(k)\Phi)\tilde{x}(k) \\ \overline{A}(k)\tilde{U}(k) &\leq \overline{b}(k) + \overline{W}(k)\tilde{x}(k) \end{aligned} \quad (5.18)$$

The Problem formulation wit inequalities is:

$$\begin{aligned} \min_{\tilde{U}} \quad & V_N(\tilde{x}(k), \tilde{U}(k)) = \tilde{U}^T(k)H\tilde{U}(k) + (F_x)\tilde{U}(k) + f(\tilde{x}) \\ & s.t. \overline{A}(k)\tilde{U}(k) \leq \overline{b}(k) + \overline{W}(k)x(k) \end{aligned} \quad (5.19)$$

This Problem can be implemented directly in matlab using the function "quadprog".
sequence of central mpc tasks

1. measure state/temperature $\overline{x}$ of each room
2. calculate $\tilde{x} = \overline{x} - x_{ss}$, $\tilde{u} = \overline{u} - u_{ss}$
3. solve (5.19) to get $U^*(k)$ with quadprog
4. extract $\tilde{u} * (k) = U^*(0)$
5. calculate $\overline{u} = \tilde{u} + u_{ss}$
6. implement $\overline{u}$ in plant
7. go back to 1

5.1.2 Distributed Case

We still want to solve (5.1) but now in a distributed fashion. By using the reformulation in (5.7) we avoid the disturbance term as seen in the previous case. We need to build the prediction model in a seperable form and will use $z(k) = \tilde{x}(k)$ and $v(k) = \tilde{u}(k)$ to build the prediction form of the equality constraints.
The prediction model is: $AY(k) = Bz_0$ wit the state equations:

$$\begin{aligned}
-z_1(0) &= -x_1(0) \\
-z_1(1) &= a_{11}z_1(0) + a_{12}z_2(0) + b_{11}v_1(0) \\
-z_1(2) &= a_{11}z_1(1) + a_{21}z_2(1) + b_{11}v_1(1)
\end{aligned}$$

$$\begin{aligned}
-z_2(0) &= -x_1(0) \\
-z_2(1) &= a_{21}z_1(0) + a_{22}z_2(0) + b_{22}v_2(0) \\
-z_2(2) &= a_{21}z_1(1) + a_{22}z_2(1) + b_{22}v_2(1)
\end{aligned}$$

To write this in matrix form we put $z(k)$ and $v(k)$ into one combined vector build as follows:

$$Y = (Y_1^T, Y_2^T)^T \tag{5.20}$$

$$Y_1 = (z_1(0)^T, z_1(1)^T, ..., z_1(N-1)^T, v_1(0)^T, v_1(1)^T, ..., v_1(N-1)^T)^T \tag{5.21}$$

with the matrices for $N = 4$ looking like:

$$A = \left[\begin{array}{cccccccc|cccccccc}
-I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{11} & -I & 0 & 0 & b_{11} & 0 & 0 & 0 & a_{12} & 0 & 0 & 0 & b_{12} & 0 & 0 & 0 \\
0 & a_{11} & -I & 0 & 0 & b_{11} & 0 & 0 & 0 & a_{12} & 0 & 0 & 0 & b_{12} & 0 & 0 \\
0 & 0 & a_{11} & -I & 0 & 0 & b_{11} & 0 & 0 & 0 & a_{12} & 0 & 0 & 0 & b_{12} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{21} & 0 & 0 & 0 & b_{21} & 0 & 0 & 0 & a_{22} & -I & 0 & 0 & b_{22} & 0 & 0 & 0 \\
0 & a_{21} & 0 & 0 & 0 & b_{21} & 0 & 0 & 0 & a_{22} & -I & 0 & 0 & b_{22} & 0 & 0 \\
0 & 0 & a_{21} & 0 & 0 & 0 & b_{21} & 0 & 0 & 0 & a_{22} & -I & 0 & 0 & b_{22} & 0
\end{array}\right], \tag{5.22}$$

$$B = \left[\begin{array}{c|c} -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \hline 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right] \tag{5.23}$$

The inequalities are reformulated as:

$$Cy(k) \leq D \tag{5.24}$$

Our constraints in (5.1) were in the form:

$$x_{min} \leq x(k) \leq x_{max} \tag{5.25}$$
$$u_{min} \leq u(k) \leq u_{max} \tag{5.26}$$

and we want to rewrite them for $z(k)$ and $v(k)$:

$$v_{min} \leq v(k) \leq v_{max} \tag{5.27}$$
$$z_{min} \leq z(k) \leq z_{max} \tag{5.28}$$

First we need those written as:

$$C_z z(k) \leq d_z \tag{5.29}$$
$$D_v v(k) \leq d_v \tag{5.30}$$

With

$$C_z = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, C_v = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, d_z = \begin{pmatrix} -z_{min} \\ z_{max} \end{pmatrix}, d_v = \begin{pmatrix} -v_{min} \\ v_{max} \end{pmatrix} \tag{5.31}$$

Now the C and D look like:

$$\begin{array}{ll} C_1 = diag(C_z, C_z, \ldots, C_z, C_v, C_v, \ldots, C_v) & D_1 = (d_z^T, d_z^T, \ldots, d_z^T, d_v^T, d_v^T, \ldots, d_v^T) \\ C_2 = diag(C_z, C_z, \ldots, C_z, C_z, C_z, \ldots, C_z) & D_2 = (d_z^T, d_z^T, \ldots, d_z^T, d_v^T, d_v^T, \ldots, d_v^T) \\ C = \begin{pmatrix} C_1 & \mathbf{0} \\ \mathbf{0} & C_2 \end{pmatrix} & D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \end{array} \tag{5.32}$$

The reformulated problem is:

$$\begin{array}{c} \min_y V_N = \frac{1}{2} y^T H y \\ s.t. Ay = Bz0 \\ Cy = D \end{array} \tag{5.33}$$

We use the Lagrangian relaxation for the constraints and get the dual function $L(y, \lambda, \mu)$.

$$L(y, \lambda, \mu) = \tfrac{1}{2}(y^T H y) + \lambda^T(Ay - Bz0) + \mu^T(Cy - D) \tag{5.34}$$

Where λ and μ are the lagrange multipliers, weighting the violation of the constraints. The Problem formulation is now a maximization over λ and minimization over y.

$$\max_{\lambda,\mu\geq 0}\left(\min_{y} L(y, \lambda, \mu)\right) \tag{5.35}$$

We start with looking at the minimization part and build the gradient over y:

$$\nabla L(y, \lambda) = Hy + A'\lambda + C'\mu \tag{5.36}$$
$$= 0 \tag{5.37}$$

and calculate the minimizing y.

$$y^+ = -\mathrm{inv}(H)(A'\lambda + C'\mu) \tag{5.38}$$

Now we apply this y^+ into $L(y, \lambda)$ to eliminate the y and get:

$$L(\lambda, \mu, z_0) = -\tfrac{1}{2}\left((A'\lambda + C'\mu)'H^{-1}(A'\lambda + C'\mu)\right) - \lambda'(Bz_0) - \mu'(D)$$

We get the optimization results for $L(\lambda, \mu, z_o)$ through gradient descent method.

$$\begin{aligned}
y^k &= -H^{-1}(A^T\lambda^k)\\
\overline{y}^k &= y^k + \tfrac{k-1}{k+2}(y^k - y^{k-1})\\
\lambda^{k+1} &= \lambda^k + \tfrac{k-1}{k+2}(\lambda^k - \lambda^{k-1}) + \frac{1}{L}(A\overline{y^k} - Bz0)\\
\mu^{k+1} &= \max(0, \mu^k + \tfrac{k-1}{k+2}(\mu^k - \mu^{k-1}) + \tfrac{1}{L}(C\overline{y} - D))
\end{aligned}$$

For L we use the "Lipschitz constant" to the gradient of $L(\lambda, \mu, y)$.

$$L = \mathrm{norm}([A^T, B^T]^T H^{-1}[A^T, B^T])$$

Choosing this Lipschitz constant L as constant step size, we have a convergent gradient method. We choose the step size to be constant because of the computational effort to compute a suitable step size on line. Since this is the distributed case, we need to separate the variables:

$$\begin{aligned}
y_1^k &= -H_{11}^-1(A_{11}^T\lambda + A_{21}^T\lambda + C_{11}^T\mu)\\
y_2^k &= -H_{22}^-1(A_{12}^T\lambda + A_{22}^T\lambda + C_{11}^T\mu)\\
\overline{y_1^k} &= y_1^k + \frac{k-1}{k+2}(y_1^k - y_1^{k-1})\\
\overline{y_2^k} &= y_2^k + \frac{k-1}{k+2}(y_2^k - y_2^{k-1})
\end{aligned} \tag{5.39}$$

$$\begin{aligned}
\lambda_1^k &= \lambda_1^k + \frac{k-1}{k+2}(\lambda_1^k - \lambda_1^{k-1}) + \frac{1}{L}(A_{11}\overline{y}_1 + A_{12}\overline{y}_2 - Bz0_1) \\
\lambda_2^k &= \lambda_2^k + \frac{k-1}{k+2}(\lambda_1^k - \lambda_1^{k-1}) + \frac{1}{L}(A_{21}\overline{y}_1 + A_{22}\overline{y}_2 - Bz0_2) \\
\mu_1^k &= \max(0, \mu_1^k + \tfrac{k-1}{k+2}(\mu_1^k - \mu_1^{k-1}) + \tfrac{1}{L}(C_{11}\overline{y}_1 - D1)) \\
\mu_2^k &= \max(0, \mu_2^k + \tfrac{k-1}{k+2}(\mu_2^k - \mu_2^{k-1}) + \tfrac{1}{L}(C_{22}\overline{y}_2 - D2))
\end{aligned} \tag{5.40}$$

sequence of distributed mpc tasks

1. each computational unit measures state/temperature $\overline{xi}$ of its room
2. each computational unit calculates $\tilde{xi} = \overline{xi} - xi_{ss}$, $\tilde{ui} = \overline{ui} - ui_{ss}$
3. solve (5.19) to get $U^*(k)$ using the seperable Iterations.repeat:

   calculate y_i through (5.39) .

   communication

   calculate λ_i, μ_i through (5.40)

   communication
4. extract $\tilde{u} * (k) = U^*(0)$
5. calculate $\overline{u} = \tilde{u} + u_{ss}$
6. implement $\overline{u}$ in plant
7. go back to 1

5.2 Delta Input Formulation together with Target Calculation

Reformulate the system such that we have

- a quadratic program for the matlab function quadprog
- we have a central system we can solve with Iterations
- we can seperate the system to solve it with distributed Iterations

This chapter is structured as follows:
First we need to change the reference tracking problem into a regulation problem through Target Calculation. On top of this we will build the delta Input system mentioned [Gör15d] through this formulation we can set constraints on the heat flow from the heater. We reformulate it such that the system is separa-

ble. Afterwards we build the prediction form used in [GR13]. This form can be used to solve the central case with the matlab function "quadprog" and with the accelerated gradient descent method. Since it is separable we can also use the accelerated gradient descent method mentioned in [GDK+13].

5.2.1 Reformulate into Regulation Problem

The room model is:

$$\begin{pmatrix}\overline{x_1}(k+1)\\ \overline{x_2}(k+1)\end{pmatrix} = \begin{pmatrix}\overline{a}_{11} & \overline{a}_{12}\\ \overline{a}_{21} & \overline{a}_{22}\end{pmatrix}\begin{pmatrix}\overline{x}_1(k)\\ \overline{x}_2(k)\end{pmatrix} + \begin{pmatrix}\overline{b}_{11} & \overline{b}_{12}\\ \overline{b}_{21} & \overline{b}_{22}\end{pmatrix}\begin{pmatrix}\overline{u}_1(k)\\ \overline{u}_2(k)\end{pmatrix} \tag{5.41}$$

$$+ \begin{pmatrix}b_{w11} & b_{w12}\\ b_{w21} & b_{w22}\end{pmatrix}\begin{pmatrix}w_1(k)\\ w_2(k)\end{pmatrix} \tag{5.42}$$

Say we want to reach $y = r = \overline{c}x$ with $\overline{c}$ as identity matrix, then we want to reach $x = r$.To be consistent with the precious variable declaration we writex_{ss}. x_{ss} is the sate we want to reach and hold, u_{ss} is the required input to obtain this reference value:

$$x_{ss} = \overline{a}x_{ss} + \overline{b}u_{ss} + b_w w \tag{5.43}$$

By introducing the state deviation: $z(k+1) = \overline{x}(k+1) - x_{ss}$ we get:

$$\tilde{x}(k+1) = \overline{x}(k+1) - x_{ss} \tag{5.44}$$

$$= \overline{a}\overline{x}(k) + \overline{b}\overline{u} + b_w w - -(\overline{a}x_{ss} + \overline{b}u_{ss} + b_w w) \tag{5.45}$$

$$= \overline{a}(\overline{x}(k) - x_{ss}) + \overline{b}(\overline{u}(k) - u_{ss}) \tag{5.46}$$

$$= \overline{a}\tilde{x}(k) + \overline{b}\tilde{u}(k) \tag{5.47}$$

Now we define the delta Input equation $\delta u = u(k) - u(k-1)$ and use a new state vector: $x(k) = (\overline{x}_1(k)^T, \overline{x}_2(k)^T, \overline{u}_1(k)^T, \overline{u}_2(k)^T)^T$, and the new input vector $\Delta u(k) = (\Delta u_1(k)^T, \Delta u_2(k)^T))^T$.

$$\begin{pmatrix}\overline{x}_1(k+1)\\ \overline{x}_2(k+1)\\ \overline{u}_1(k)\\ \overline{u}_2(k)\end{pmatrix} = \begin{pmatrix}\overline{a}_{11} & \overline{a}_{12} & \overline{b}_{11} & \overline{b}_{12}\\ \overline{a}_{21} & \overline{a}_{22} & \overline{b}_{21} & \overline{b}_{22}\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\overline{x}_1(k)\\ \overline{x}_2(k)\\ \overline{u}_1(k-1)\\ \overline{u}_2(k-1)\end{pmatrix} + \begin{pmatrix}\overline{b}_{11} & \overline{b}_{12}\\ \overline{b}_{21} & \overline{b}_{22}\\ 1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}\Delta u_1(k)\\ \Delta u_2(k)\end{pmatrix} \tag{5.48}$$

Inserting zero for $\overline{b}_{12}$, $\overline{b}_{21}$, $\overline{bw}_{12}$ and $\overline{bw}_{21}$ yields the following separable system:

$$\left(\begin{array}{c}\overline{x}_1(k+1)\\ \overline{u}_1(k+1)\\ \hline \overline{x}_2(k)\\ \overline{u}_2(k)\end{array}\right) = \left(\begin{array}{cc|cc} \overline{a}_{11} & \overline{b}_{11} & \overline{a}_{12} & 0\\ 0 & 1 & 0 & 0\\ \hline \overline{a}_{21} & 0 & \overline{a}_{22} & \overline{b}_{22}\\ 0 & 0 & 0 & 1\end{array}\right)\left(\begin{array}{c}\overline{x}_1(k)\\ \overline{u}_1(k)\\ \hline \overline{x}_2(k-1)\\ \overline{u}_2(k-1)\end{array}\right) + \left(\begin{array}{c|c} \overline{b}_{11} & 0\\ 1 & 0\\ \hline 0 & \overline{b}_{22}\\ 0 & 1\end{array}\right)\begin{pmatrix}\Delta u_1(k)\\ \Delta u_2(k)\end{pmatrix} \tag{5.49}$$

$$\begin{pmatrix} z_1(k+1) \\ z_2(k+1) \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} z_1(k) \\ z_2(k) \end{pmatrix} + \begin{pmatrix} b_{11} & 0 \\ 0 & b_{22} \end{pmatrix} \begin{pmatrix} v_1(k) \\ v_2(k) \end{pmatrix} \tag{5.50}$$

$$z(k+1) = az(k) + b\Delta u \tag{5.51}$$

Combining z and Δu into the vector y we get the prediction model$Ay = Bz_0$ with the following vectors and matrices:

$$y = (y_1^T, y_2^T)^T \tag{5.52}$$

$$y_1 = (z_1(0)^T, z_1(1)^T, \ldots, z_1(N-1)^T, v_1(0)^T, v_1(1)^T, \ldots, v_1(N-1)^T)^T \tag{5.53}$$

$$y_2 = (z_2(0)^T, z_2(1)^T, \ldots, z_2(N-1)^T, v_2(0)^T, v_2(1)^T, \ldots, v_2(N-1)^T)^T \tag{5.54}$$

$$A_{11} = \begin{pmatrix} -I & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ a_{11} & -I & 0 & \ldots & 0 & 0 & b_{11} & 0 & \ldots & 0 & 0 \\ 0 & a_{11} & -I & \ldots & 0 & 0 & 0 & b_{11} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & a_{11} & -I & 0 & 0 & \ldots & b_{11} & 0 \end{pmatrix} \tag{5.55}$$

$$A_{22} = \begin{pmatrix} -I & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ a_{22} & -I & 0 & \ldots & 0 & 0 & b_{22} & 0 & \ldots & 0 & 0 \\ 0 & a_{22} & -I & \ldots & 0 & 0 & 0 & b_{22} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & a_{22} & -I & 0 & 0 & \ldots & b_{22} & 0 \end{pmatrix} \tag{5.56}$$

$$A_{12} = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ a_{12} & 0 & 0 & \ldots & 0 & 0 & b_{12} & 0 & \ldots & 0 & 0 \\ 0 & a_{12} & 0 & \ldots & 0 & 0 & 0 & b_{12} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & a_{12} & 0 & 0 & 0 & \ldots & b_{12} & 0 \end{pmatrix} \tag{5.57}$$

$$A_{12} = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ a_{21} & 0 & 0 & \ldots & 0 & 0 & b_{21} & 0 & \ldots & 0 & 0 \\ 0 & a_{21} & 0 & \ldots & 0 & 0 & 0 & b_{21} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & a_{21} & 0 & 0 & 0 & \ldots & b_{21} & 0 \end{pmatrix} \tag{5.58}$$

$$B_{11} = \begin{pmatrix} -I \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, B_{12} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, B_{21} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, B_{22} = \begin{pmatrix} -I \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.59}$$

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right)\left(\begin{array}{c} y_1 \\ y_2 \end{array}\right) = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) \tag{5.60}$$

The cost function is: $V_N = \frac{1}{2} y^T H y$. With:

$$H_1 = \text{blkdiag}(Q_1, Q_1, \ldots, Q_1, R_1, R_1, \ldots, R_1) \tag{5.61}$$
$$H_2 = \text{blkdiag}(Q_2, Q_2. \ldots, Q_2, R_2, R_2, \ldots, R_2) \tag{5.62}$$
$$H = \text{blkdiag}(H1, H2); \tag{5.63}$$

Where Q_1 is the weighting matrix for z_1, Q_2 for z_2, R_1 for v_1, R_2 for v_2.
Q_1, Q_2, R_1, R_2 are multiples of the identity matrix.
We still need to reformulate the constraints:

$$-\overline{x} \leq -\overline{x}_{min} \tag{5.64}$$
$$\overline{x} \leq \overline{x}_{max} \tag{5.65}$$
$$-\overline{u} \leq -\overline{u}_{min} \tag{5.66}$$
$$\overline{u} \leq \overline{u}_{max} \tag{5.67}$$

into $Cy \leq D$. We start with separating them and using the reference values x_{ss} and u_{ss} to rewrite them as:

$$\begin{aligned} -\overline{xi} + xi_{ss} &\leq -\overline{x}i_{min} + xi_{ss} \\ -(\overline{xi} - xi_{ss}) &\leq -(\overline{xi}_{min} - xi_{ss}) \\ \tilde{xi} &\leq \tilde{xi}_{min} \\ \overline{xi} - xi_{ss} &\leq \overline{xi}_{max} - xi_{ss} \\ \tilde{xi} &\leq \tilde{xi}_{max} \end{aligned} \tag{5.68}$$

$$\begin{aligned} -\overline{ui} + ui_{ss} &\leq -\overline{u}i_{min} + ui_{ss} \\ -(\overline{ui} - ui_{ss}) &\leq -(\overline{ui}_{min} - ui_{ss}) \\ \tilde{ui} &\leq \tilde{ui}_{min} \\ \overline{ui} - ui_{ss} &\leq \overline{ui}_{max} - ui_{ss} \\ \tilde{ui} &\leq \tilde{ui}_{max}. \end{aligned} \tag{5.69}$$

These equations can be put into matrix form:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{xi} \\ \tilde{ui} \end{pmatrix} \leq \begin{pmatrix} -\tilde{xi}_{min} + \tilde{xi}_{ss} \\ -\tilde{ui}_{min} + u_{ss} \\ \tilde{xi}_{max} - x_{ss} \\ \tilde{ui}_{max} - u_{ss} \end{pmatrix} \tag{5.70}$$

$$C_{zi} zi \leq d_{zi} \tag{5.71}$$

The constraints on Δu are:

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix} vi \leq \begin{pmatrix} -u_{min} \\ u_{max} \end{pmatrix} \tag{5.72}$$

$$C_{vi} vi \leq d_{vi}. \tag{5.73}$$

Now we can put $C_{z_i} z_i \leq d_{z_i}$ and $C_{v_i} v_i \leq d_{v_i}$ into one constraints $Cy \leq D$ with the matrices as follows:

$$C = \text{blkdiag}(C_1, C_2) \tag{5.74}$$

$$C_1 = \text{blkdiag}(C_{z1}, C_{z1}, \ldots, C_{z1}, C_{v1}, C_{v1}, \ldots, C_{v1}) \tag{5.75}$$

$$C_2 = \text{blkdiag}(C_{z2}, C_{z2}, \ldots, C_{z2}, C_{v2}, C_{v2}, \ldots, C_{v2}) \tag{5.76}$$

$$D = (D_1^T, D_2^T)^T \tag{5.77}$$

$$D_1 = (d_{z1}, d_{z1}, \ldots, d_{z1}, d_{u1}, d_{u1}, \ldots, d_{u1}) \tag{5.78}$$

$$D_2 = (d_{z2}, d_{z2}, \ldots, d_{z2}, d_{u2}, d_{u2}, \ldots, d_{u2}) \tag{5.79}$$

The final problem formulation for the separable model predictive control is:

$$\begin{aligned} &\underset{y}{\text{minimize}} \; V_N = y^T H y \\ &\text{subject to } Ay = B \\ &Cy \leq D \end{aligned} \tag{5.80}$$

5.2.2 Central Solution using quadprog

Using matlab we can simply use quadprog (see documentation):
repeat

- measure current temperature/states $\tilde{x}(k)$
- build current deviation $z(k) = \tilde{x}(k) - xss$
- get optimal y* from (5.80) through quadprog
- get $\Delta u(0)$ from y*
- implement $\Delta u(0)$
- go back to one

5.2.3 Central Solution using Accelerated Gradient Method

The iterations we use are:

$$\begin{aligned} y^k &= -H^{-1}(A^T\lambda + C^T\mu) \\ \overline{y}^k &= y^k + \alpha_k(y^k - y^{k-1}) \\ \lambda^k &= \lambda^k + \alpha_k(\lambda^k - \lambda^{k-1}) + \tfrac{1}{L}(A\overline{y}^k - Bz_0) \\ \mu^k &= max(0, \mu^k + \alpha_k(\mu^k - \mu^{k-1}) + \tfrac{1}{L}(C\overline{y}^k - D) \end{aligned} \tag{5.81}$$

- measure current temperature/states $\tilde{x}(k)$
- build current deviaten $z(k) = \tilde{x}(k) - xss$
- get optimal y* from (5.80) through the Iterations
- get $\Delta u(0)$ from y*
- implement $\Delta u(0)$
- fo back to one

5.2.4 distributed Solution using Accelerated Gradient Method

We can separate the Iterations 5.81 as follows:

$$\begin{aligned} \begin{pmatrix} \overline{y}_1 \\ \overline{y}_2 \end{pmatrix} &= -H^{-1}(A^T\lambda + C^T\mu) + \alpha_k(y - y^{past}) \\ &= -\begin{pmatrix} H_1^{-1}(A_{11}^T\lambda_1 + A_{21}^T\lambda_2 + C_{11}^T\mu_1) \\ H_2^{-1}(A_{12}^T\lambda_1 + A_{22}^T\lambda_2 + C_{22}^T\mu_2) \end{pmatrix} + \alpha_k \begin{pmatrix} y_1 - y_1^{past} \\ y_2 - y_2^{past} \end{pmatrix} \\ \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} &= \lambda + \alpha_k\left(\lambda - \lambda^{past}\right) + \tfrac{1}{L}\left(A\overline{y} - Bz_0\right) \\ &= \begin{pmatrix} \lambda_1 + \alpha_k(\lambda_1 - \lambda_1^{past}) \\ \lambda 2 + \alpha_k(\lambda_2 - \lambda_2^{past}) \end{pmatrix} + \tfrac{1}{L}\begin{pmatrix} A_{11}\overline{y}_1 + A_{12}\overline{y}_2 - B_{11}z_{01} \\ A_{21}\overline{y}_1 + A_{22}\overline{y}_2 - B_{22}z_{02} \end{pmatrix} \\ \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} &= \mu + \alpha_k(\mu - \mu^{past}) + \tfrac{1}{L}(C\overline{y} - D) \\ &= \begin{pmatrix} \mu_1 + \alpha_k(\mu_1 - \mu_1^{past}) \\ \mu_2 + \alpha_k(\mu_2 - \mu_2^{past}) \end{pmatrix} + \tfrac{1}{L}\begin{pmatrix} C_{11}\overline{y}_1 - D_1 \\ C_{22}\overline{y}_2 - D_2 \end{pmatrix} \end{aligned} \tag{5.82}$$

- measure current state x
- calculate deviation $z = x - x_{ss}$

- calculate y* through iteration (5.82)

  computational unit 1 calculates $\overline{y}_1, \lambda_1, \mu_1$

  computational unit 2 calculates $\overline{y}_2, \lambda_2, \mu_2$
- get $\Delta u(0)$ from y*a
- implement $\Delta u(0)$
- fo back to one

6 Results

6.1 Setup

In this chapter we will discuss the results of the two reference tracking methods. For each method "delta Input" and "Target Calculation" we discuss the behaviour for the central problem solved with quadprog, the central problem solved with the accelerated gradient method, and the distributed problem solved with accelerated gradient method. All Iterative methods use 100 Iterations to calculate the optimal value for each time step. Only the distributed accelerated gradient method for the Target Calculation reference tracking problem needs more than 100 Iterations to converge. It's implemented with 500 Iterations. At time step 20 the reference value changes from 22 °C in each room to 20 °C in room 1, and 27 °C in room 2. The outside temperature is $10°C$ and the predictin horizon is $N = 12$. First we will discuss the results of Target Calculation solved with a central MPC and distributed MPC, then results of delta Input formulation solved with central MPC and distributed MPC.

6.2 Target Calculation

The pictures 6.1, 6.2, 6.3 show the results for central model predictive control using quadprog to solve the optimization problem of each time step. The pictures 6.4, 6.5, 6.6 show the results for central model predictive control using the accelerated gradient method to solve the optimization problem of each time step. The pictures 6.7, 6.8, 6.9 show the results for distributed model predictive control using the distributed accelerated gradient method to solve the distributed optimization problem of each time step.
The reference value is reached exactly. Since there are no constraints of the change the input can make in one time step the input can change to the specified reference value in one time step. Due to this the results of central MPC solved with quadprog, central MPC solved with Iterations and distributed MPC solved with Iterations are alike.

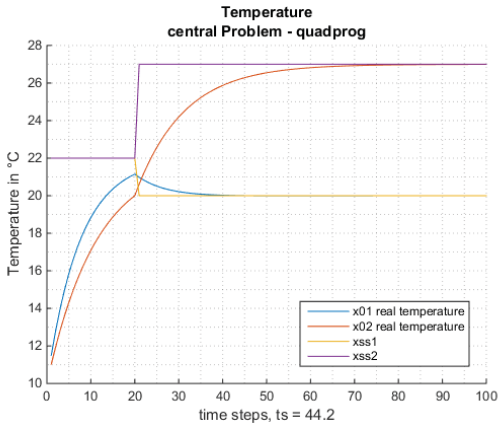

Figure 6.1: Target Calculation - central Problem solved with quadprog, Temperature profile

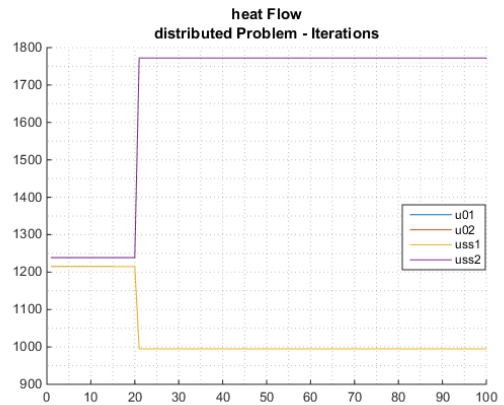

Figure 6.2: Target Calculation - central Problem solved with quadprog, heat Flow

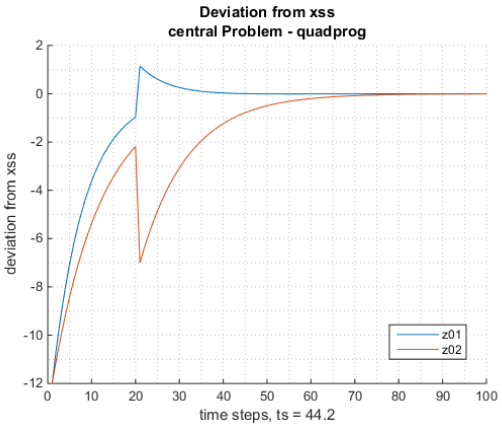

Figure 6.3: Target Calculation - central Problem solved with quadprog, deviation of Temperature to reference value

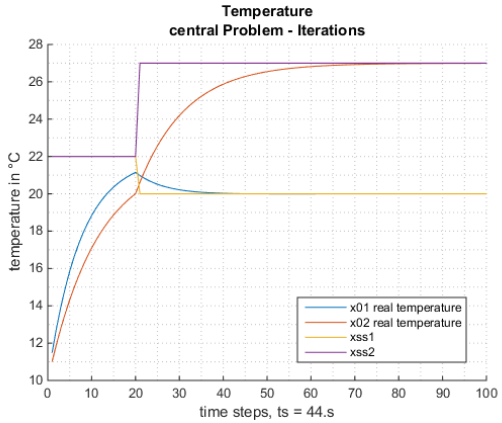

Figure 6.4: Target Calculation - central Problem solved with accelerated gradient method, Temperature profile

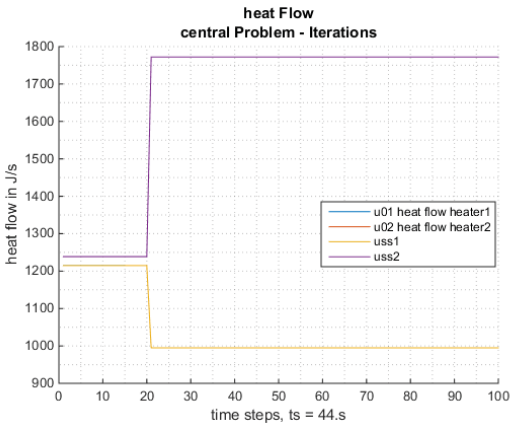

Figure 6.5: Target Calculation - central Problem solved with accelerated gradient method, heat Flow

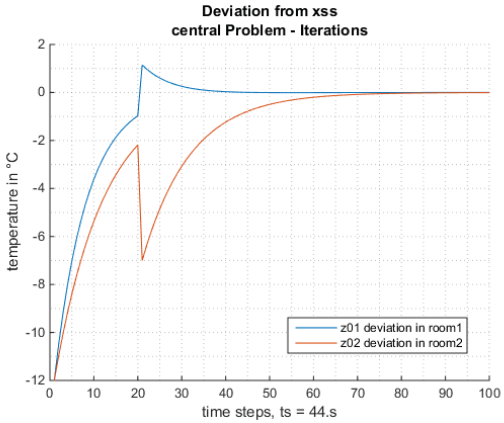

Figure 6.6: Target Calculation - central Problem solved with accelerated gradient method, deviation of Temperature from reference value

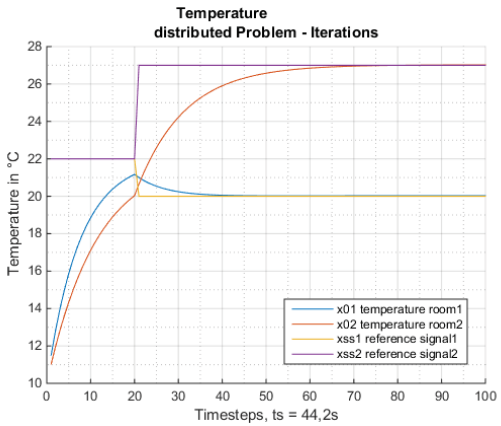

Figure 6.7: Target Calculation - distributed Problem solved with accelerated gradient method, Temperature profile

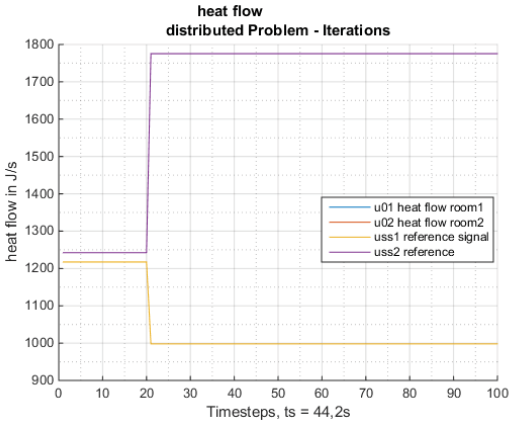

Figure 6.8: Target Calculation - distributed Problem solved with accelerated gradient method, heat Flow

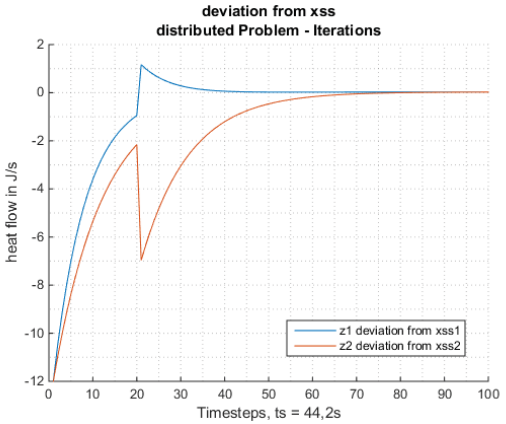

Figure 6.9: Target Calculation - distributed Problem solved with accelerated gradient method, deviation of Temperature from reference value

6.3 Delta Input Formulation

6.3.1 Central MPC, quadprog

The pictures 6.10, 6.2, 6.3 show the results for the central Problem solved with quadprog. One can see that Temperature and heat flow of room 2 have a overshoot but reach the reference value. Room 1, which reference temperature is changed from 22 °C to 20 °C does not have an overshoot.

6.3.2 Central MPC, Iterations

The pictures 6.13, 6.14, 6.15 show the results of the central problem solved with the accelerated gradient method. Neither the Temperature nor the heat flow show an overshoot. And the reference value is faster then with central MPC and quadprog. The largest difference one can see in the comparison of 6.12 and 6.15. The dU of the System solved with quadprog reaches zero at time step 50 for the first time, but due to the overshoot it leaves zero again. The dU of the System solved with central accelerated gradient method reaches zero at 50 and does not leave it. The behaviour of the system controlled with central MPC solved with quadprog and the central MPC solved with the accelerated gradient method is expected to be the same. But the solution in both cases is not optimal, and the number of iterations for the accelerated gradient method was chosen arbitrary,

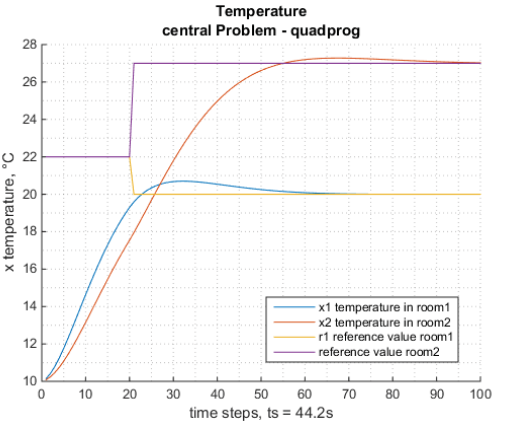

Figure 6.10: Delta Input Formulation - central Problem solved with quadprog, Temperature profile

which might have influenced the behaviour. We may need more optimization iterations to result in a optimal solution. Also the prediction models of both methods were built independent of each other, so there might be a difference in the coefficients which produced this different behaviour.

6.3.3 Distributed MPC, Iterations

The distributed system reaches the reference value even faster without overshoot.

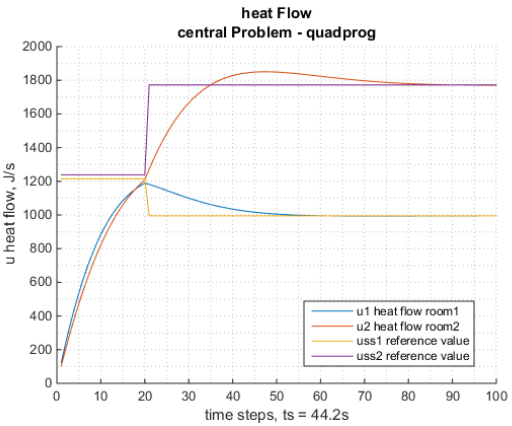

Figure 6.11: Delta Input Formulation - central Problem solved with quadprog, heat Flow

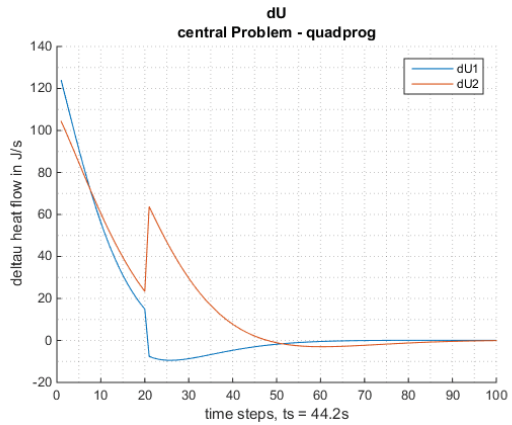

Figure 6.12: Delta Input Formulation - central Problem solved with quadprog, change of heat flow

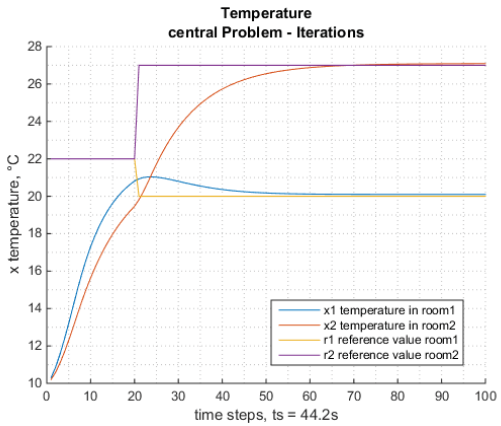

Figure 6.13: Delta Input Formulation - central Problem solved with accelerated gradient method, Temperature profile

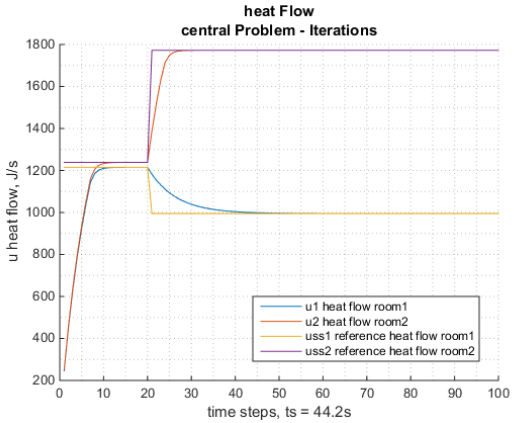

Figure 6.14: Delta Input Formulation - central Problem solved with accelerated gradient method, heat flow

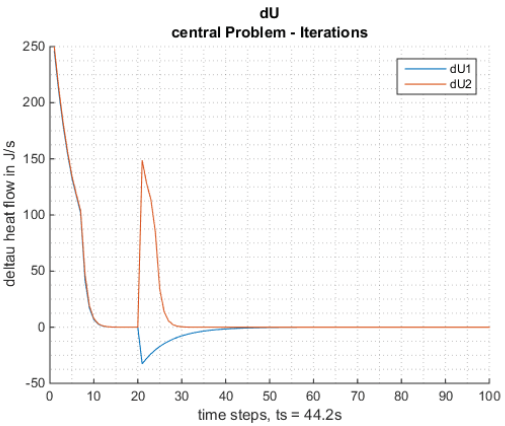

Figure 6.15: Delta Input Formulation - central Problem solved with accelerated gradient method, change of heat flow

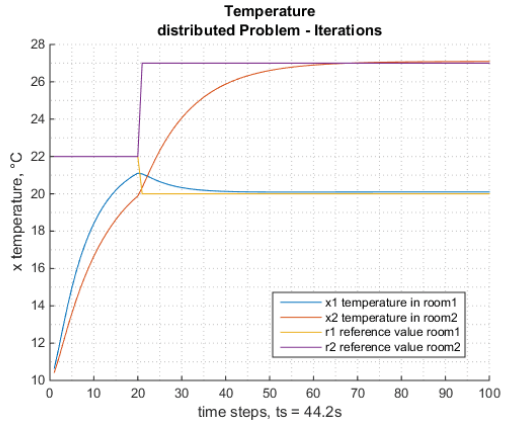

Figure 6.16: Delta Input Formulation - distributed Problem solved with accelerated gradient method, Temperature profile

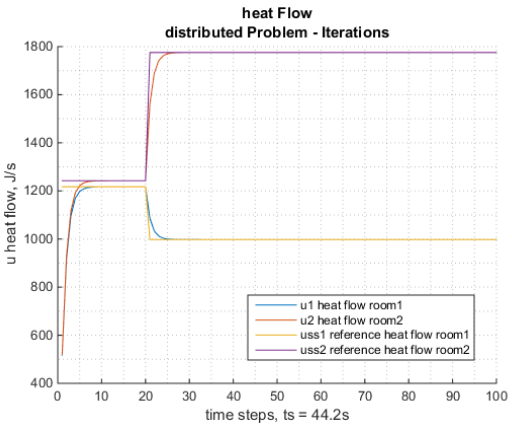

Figure 6.17: Delta Input Formulation - distributed Problem solved with accelerated gradient method, heat flow

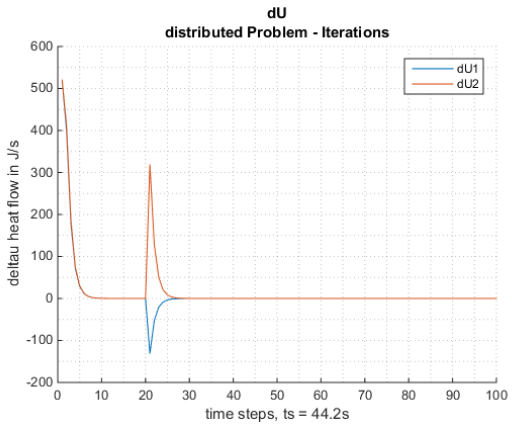

Figure 6.18: Delta Input Formulation - distributed Problem solved with accelerated gradient method, change of heat flow

7 Conclusion and Future Work

7.1 Conlusion

The control of the house temperature to a specific temperature using central and distributed target calculation and delta Input formulation was implemented. The reference temperature is reached in all cases. The Target Calculation is not useful to control the heaters of the house model, since the MPC algorithm build this way cannot consider the constraints on the change of heat flow a heater can produce. This way it is like turning the heater on or of. Also the distributed algorithm needs a lot of iterations to converge. The delta Input Formulation combined with Target Calculation gives the possibilty to set constraints on the change of heat flow. This way the result gets more realistic. Here the distributed accelerated algorithm is faster then the central accelerated algorithm.

7.2 Future Work

The following points are useful additions to this thesis:

- a stopping criterion for the Iterations
- more rooms
- more specific room and heater model
- delta Input without Target Calculation

A stopping criterion would reduce the number of Iterations needed to get a sub-optimal result, that fulfils specified performance and stability demands. More rooms would give more realistic results regarding the computational effort and the time consumption through the Iterations in comparison to the central solution. The more specific room and heater model has the same purpose. The more specific the model becomes, the more realistic the result will be. In this thesis we used the delta Input formulation and on top of it target Calculation. It would be interesting to implement the delta Input formulation without the additional target Calculation and see the results.

Bibliography

[Bol06] Bolton, William: *Bausteine mechatronischer Systeme*. 3rd ed. 2006

[BV09] Boyd, Stephen ; Vandenberghe, Lieven: *Convex Optimization*. 7th ed. Cambridge university Press, 2009

[BXMM08] Boyd, Stephen ; Xiao, Lin ; Mutapcic, Almir ; Mattingley, Jacob: *Notes on Decomposition Methods*. notes for EE364B, Stanford University, Winter 2006-07, 2008

[FPEN14] Franklin, Gene F. ; Powell, J. D. ; Emami-Naeini, Abbas: *Feedback Control of Dynamic Systems*. 7th ed. 2014

[GDK+13] Giselsson, Pontus ; Doan, Minh D. ; Keviczky, Tamás ; Schutter, Bart D. ; Rantzer, Anders: Accelerated gradient methods and dual decomposition in distributed model predictive control. (2013)

[Gis12] Giselsson, Pontus: *Gradient-Based Distributed Model Predictive Control*. doctoral dissertation, Department of Automatic Control, Lund University, 2012

[GR13] Giselsson, Pontus ; Rantzer, Anders: On feasibility, stability and performance in distributed model predictive control. In: *IEEE Control Systems Society* (2013)

[Gör15a] Görges, Daniel: *Model Predictive Control*. lecture notes 2015, Lehrstuhl für Regelungssysteme, Fachbereich Elektrotechnik und Informationstechnik, Technische Universit?t Kaiserslautern, 2015

[Gör15b] Görges, Daniel: *Model Predictive Control, Chapter 2*. lecture notes 2015, Lehrstuhl für Regelungssysteme, Fachbereich Elektrotechnik und Informationstechnik, Technische Universit?t Kaiserslautern, 2015

[Gör15c] Görges, Daniel: *Model Predictive Control, Chapter 4 and Chapter 5*. lecture notes 2015, Lehrstuhl für Regelungssysteme, Fachbereich Elektrotechnik und Informationstechnik, Technische Universit?t Kaiserslautern, 2015

[Gör15d] Görges, Daniel: *Model Predictive Control, Chapter 7*. lecture notes

2015, Lehrstuhl für Regelungssysteme, Fachbereich Elektrotechnik und Informationstechnik, Technische Universitaet Kaiserslautern, 2015

[www] `http://www.engineeringtoolbox.com/` (last view: 18.03.2015)

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER´S THESIS

Date of issue
February 2016

Document Number
ISRN LUTFD2/TFRT--5998--SE

Author(s)

Kirsten Carstaedt

Supervisor
Daniel Görges, Department of Electrical and Computer Engineering, University of Kaiserslautern, Germany
Felix Berkel, Department of Electrical and Computer Engineering, University of Kaiserslautern, Germany
Meike Stemmann, Dept. of Automatic Control, Lund University, Sweden
Anders Rantzer, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle

Distributed Model Predictive Control for Building Temperature Control

Abstract

This thesis and technical report concentrates on distributed control using a distributed model predictive scheme. The model of a two room house and three room houses is build and a distributed model predictive control (MPC) algorithm is implemented in order to reach specified room temperatures with minimized energy effort in each room. For reference tracking Target Calculation and the delta input scheme are used. The MPC optimization problem is solved at each time step through an iterative method, where the number of iterations is reduced through a stopping criterion guaranteeing stability and a prespecified amount of performance and feasibility. The optimization problem is divided up into subproblems, where each subproblem takes less computational effort than the central optimization problem. Due to the possibility of coupling between subsystems, communication between the subsystems is needed. The reference values are reached and iterations needed to solve the optimization are reduced with the stopping condition. This method saves computing time and gives privacy to each subsystem, since only required information is communicated. Also the subsystems get less susceptible to the failure of one coupled subsystem, since if one subsystem fails, the others could go on. But, due to the needed communication, this method is more suitable for large systems with sparse coupling. For a small system, or too much coupling the communication effort will get to high.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-62

Recipient's notes

Security classification