Control of a Multivariable Lighting System

Axel Halldin
Abstract

This master’s thesis examines how a small MIMO lighting system can be identified and controlled. Two approaches are examined and compared; the first approach is a dynamic model using state space representation, where the system identification technique is Recursive Least Square, RLS, and the controller is an LQG controller; the second approach is a static model derived from the physical properties of light and a feedback feed-forward controller consisting of a PI controller coupled with a Control Allocation, CA, technique. For the studied system, the CA-PI approach significantly outperforms the LQG-RLS approach, which leads to the conclusion that the system’s static properties are predominant compared to the dynamic properties.
Acknowledgments

First of all I want to thank Syntronic AB and Andreas Forsberg for the opportunity for me to write my master’s thesis there. I also want to thank my supervisor at Syntronic, Anders Bergmark, for his support as well as Daniel Nordgren for his help with assembling the system.

I also want to thank my supervisor at ISY, Gustav Lindmark, for his support and for our discussions on approaches and implementation techniques. A special thanks to my examiner at ISY, Daniel Axehill, for walking the extra mile while providing valuable ideas and feedback.

Finally, I want to thank everyone I’ve gotten to know through different student associations during my time at Linköping University. You are the reason I’ve had the energy to finish my studies and the reason that I’ll look back on these years with fondness.

Linköping, December 2016
Axel Halldin
# Contents

Notation  ix

1 Introduction  1
   1.1 Motivation   1
   1.2 Purpose and Goal   1
   1.3 Limitations and Assumptions   2
   1.4 Outline   2

2 Method  3
   2.1 Development Environment   3
   2.2 Problem Decomposition   3
      2.2.1 System   3
      2.2.2 System Identification   4
      2.2.3 Controller Design   4
   2.3 System Identification   4
   2.4 Controller Design   4
   2.5 Performance Evaluation   5

3 System Description  7
   3.1 General Characteristics   7
   3.2 LEDs   8
   3.3 Sensors   9
   3.4 The Limited System   9

4 Preliminaries  11
   4.1 Recursive Least Squares   11
   4.2 Linear-Quadratic-Gaussian Controller   14
   4.3 Model Predictive Control   15
   4.4 Photometry   20
   4.5 Feedback and Feed-Forward   21
   4.6 PI Controller   22
   4.7 Control Allocation   24
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Recursive Least Squares</td>
<td>27</td>
</tr>
<tr>
<td>5.1.1</td>
<td>General Recursive Least Squares Identification</td>
<td>27</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Offline Identification</td>
<td>28</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Online Identification</td>
<td>28</td>
</tr>
<tr>
<td>5.2</td>
<td>LQG</td>
<td>29</td>
</tr>
<tr>
<td>5.3</td>
<td>MPC</td>
<td>30</td>
</tr>
<tr>
<td>5.4</td>
<td>Physical Identification</td>
<td>31</td>
</tr>
<tr>
<td>5.4.1</td>
<td>General Physical Identification</td>
<td>31</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Offline Identification</td>
<td>31</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Online Identification</td>
<td>32</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Implementation Specifics</td>
<td>32</td>
</tr>
<tr>
<td>5.5</td>
<td>Control Allocation PI Controller</td>
<td>33</td>
</tr>
<tr>
<td>5.6</td>
<td>Industrial Features</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>Results</td>
<td>37</td>
</tr>
<tr>
<td>6.1</td>
<td>Dynamic Approach</td>
<td>38</td>
</tr>
<tr>
<td>6.2</td>
<td>Static Approach</td>
<td>46</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>51</td>
</tr>
<tr>
<td>7.1</td>
<td>System Properties</td>
<td>51</td>
</tr>
<tr>
<td>7.2</td>
<td>Control Signal Selection</td>
<td>51</td>
</tr>
<tr>
<td>7.3</td>
<td>Conclusion Summary</td>
<td>52</td>
</tr>
<tr>
<td>7.4</td>
<td>Future Research</td>
<td>53</td>
</tr>
<tr>
<td>A</td>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>RLS System Parameters</td>
<td>55</td>
</tr>
<tr>
<td>A.2</td>
<td>LQG Parameters</td>
<td>56</td>
</tr>
<tr>
<td>A.3</td>
<td>Physical Identification Parameters</td>
<td>56</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>57</td>
</tr>
</tbody>
</table>
## Notation

### Sets of numbers

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{R} )</td>
<td>The set of all real numbers</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARX</td>
<td>Auto-Regressive eXogenous</td>
</tr>
<tr>
<td>ARMAX</td>
<td>Auto-Regressive Moving Average eXogenous</td>
</tr>
<tr>
<td>CA</td>
<td>Control Allocation</td>
</tr>
<tr>
<td>ISY</td>
<td>The Department of Electrical Engineering at</td>
</tr>
<tr>
<td></td>
<td>Linköping University</td>
</tr>
<tr>
<td>LED</td>
<td>Light-Emitting Diode</td>
</tr>
<tr>
<td>LQG</td>
<td>Linear-Quadratic-Gaussian (controller)</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time-Invariant (system)</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple Input Multiple Output</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control (controller)</td>
</tr>
<tr>
<td>PI</td>
<td>Proportional, Integral (controller)</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional, Integral, Differential (controller)</td>
</tr>
<tr>
<td>PBRS</td>
<td>Pseudo Random Binary Sequence</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Programming (Optimization problem)</td>
</tr>
<tr>
<td>RLS</td>
<td>Recursive Least Squares</td>
</tr>
<tr>
<td>SISO</td>
<td>Single Input Single output</td>
</tr>
<tr>
<td>Variable</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>Angle between the source surface normal and the sensor</td>
</tr>
<tr>
<td>( \beta )</td>
<td>Angle between the ( I_\alpha ) vector and the sensor surface normal</td>
</tr>
<tr>
<td>( \theta )</td>
<td>ARX model parameters</td>
</tr>
<tr>
<td>( \varphi )</td>
<td>ARX regressors</td>
</tr>
<tr>
<td>( a_{i,j}^{(k)} )</td>
<td>Element ((i, j)) of matrix ( A_k )</td>
</tr>
<tr>
<td>( A_k )</td>
<td>ARX model parameters for ( y(t - k) )</td>
</tr>
<tr>
<td>( b_{i,j}^{(k)} )</td>
<td>Element ((i, j)) of matrix ( B_k )</td>
</tr>
<tr>
<td>( B_k )</td>
<td>ARX model parameters for ( u(t - k) )</td>
</tr>
<tr>
<td>( e(t) )</td>
<td>Output error, ( r(t) - y(t) )</td>
</tr>
<tr>
<td>( G )</td>
<td>System</td>
</tr>
<tr>
<td>( F )</td>
<td>Feedback controller</td>
</tr>
<tr>
<td>( F_f )</td>
<td>Feed forward controller</td>
</tr>
<tr>
<td>( I )</td>
<td>Identity matrix of appropriate dimensions</td>
</tr>
<tr>
<td>( k )</td>
<td>Arbitrary index</td>
</tr>
<tr>
<td>( m )</td>
<td>Input order of system difference equation</td>
</tr>
<tr>
<td>( M )</td>
<td>Number of samples for each intensity (see ( N ))</td>
</tr>
<tr>
<td>( n )</td>
<td>Output order of system difference equation</td>
</tr>
<tr>
<td>( N )</td>
<td>Number of intensities in physical identification</td>
</tr>
<tr>
<td>( p )</td>
<td>Number of outputs</td>
</tr>
<tr>
<td>( Q_i )</td>
<td>Weight matrix</td>
</tr>
<tr>
<td>( r )</td>
<td>Number of inputs, number of control signals, number of LEDs</td>
</tr>
<tr>
<td>( r(t) )</td>
<td>Reference signal</td>
</tr>
<tr>
<td>( R )</td>
<td>Distance between a LED and a sensor</td>
</tr>
<tr>
<td>( T_s )</td>
<td>System sampling time</td>
</tr>
<tr>
<td>( u(t) )</td>
<td>Control signal</td>
</tr>
<tr>
<td>( x(t) )</td>
<td>State variable</td>
</tr>
<tr>
<td>( y(t) )</td>
<td>Measurement signal</td>
</tr>
<tr>
<td>( z(t) )</td>
<td>Controlled variable</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter is an introduction of this master’s thesis. In the following sections, the motivation, purpose and goal, limitations and assumptions, and the outline of the report are described.

1.1 Motivation

Assuring adequate lighting levels is pertinent in many situations, from homes to office spaces and warehouses, and is often part of work environment legislation. By controlling every light source individually and only providing light where light is needed, redundant light sources can be turned off and thus saving energy and money.

Syntronic AB does, to some extent, see commercial possibilities in an end-user smart lighting product but are primarily interested in techniques to control multi-variable over-actuated systems. Syntronic AB’s motivation is to take the findings of this thesis on other over-actuated physical systems, such as heat distribution and industrial ovens, where redundant input is more expensive.

1.2 Purpose and Goal

The system described in Chapter 3 needs to be controlled so that the sensors each receive a desired amount of light, which is achieved by choosing light intensity on each LED. However, the control system needs to be able to handle system changes such as

- Sensors being moved in the room.
• LED:s malfunctioning, being covered, or turned off.
• External changes such as sun shining in (parts of) the room.

These changes can be summarized as changes in system properties or, for a model of the system, system parameters and outputs. The goal of this thesis is to find and implement a solution that can identify system parameters and satisfactorily control the system, with no previous system data available.

1.3 Limitations and Assumptions

The system described in Chapter 3 is a small scale model of a room, or in other words: a box. The sensors are assumed to be positioned within the box at all times, and this thesis will not consider the eventuality of removing sensors from the box during execution. The system will have different characteristics and, therefore, solutions will have different possible output ranges depending of e.g. actuator and sensor placement. Therefore, there are limitations on what is possible to achieve in different experimental setups.

1.4 Outline

Chapter 2 describes the methodology of this thesis, including problem decomposition, developing environment and algorithm selection processes. Chapter 3 contains a system overview and description of system components. Chapter 4 summarizes the theoretical foundation that the thesis is built on. Chapter 5 describes how the solutions were derived from theory and implemented. Chapter 6 contains the performance results of these solutions during experimental setup, as well as some origins of some characteristics. Chapter 7 contains conclusions of system properties, a discussion of similarities and differences of the solutions, and suggested further research.
2 Method

2.1 Development Environment

MATLAB’s Simulink is used to implement controllers and system identification routines, as well as communication to the system. In particular, MATLAB S-functions are used to implement algorithms ill-suited for regular Simulink Block implementation.

2.2 Problem Decomposition

In order to achieve a sound development methodology, a minimum viable product is developed in the primary stages in order to allow a core on which additional functionality could be expanded. This minimum viable product will only solve a simplified version of the problem. However, the approaches for the least viable product are selected with these functionality expansions in mind and implemented in a manner that allows these expansions.

2.2.1 System

For the minimum viable product, a limited system is considered. Instead of studying the 58-input-5-output system described in Chapter 3, an 8-input-2-output stationery system is studied. The LED:s (inputs) are symmetrically distributed on each LED stripe and the sensors (outputs) are arbitrarily placed in the box. These simplifications allow for a more manageable problem.
2.2.2 System Identification

For the minimum viable product, the system is identified prior to execution (off-line) and system parameters are viewed as time-invariant during execution. These assumptions have the implication that system controllability and observability is known prior to controller design. See Section 2.3 for a more thorough description of the system identification methodology.

2.2.3 Controller Design

For the minimum viable product, the system is assumed to be an LTI system not needing adaptive control. Hence, a feedback controller is assumed sufficient to control the simplified system described in Section 2.2.1. See Section 2.4 for a more thorough description of the controller design methodology.

2.3 System Identification

The system identification process consists of an algorithm selection process, implementation and testing, and evaluation. Because the evaluation of the system identification algorithm is closely linked with the controller, these are both described in Section 2.5.

Potential identification algorithms are found in [10] as well as from personal communications with Martin Enqvist and Lennart Ljung, both at ISY. These discussions are followed up by examining algorithms in primarily [9, 11].

In order to qualify, algorithms must first and foremost be MIMO compatible in order to be considered. Algorithms should also work (with slight modifications) for both online and offline identification. Finally, whether algorithms require little or much information about the dynamics or physicality of the system is considered in the algorithm selection process.

The algorithms are implemented in Simulink as S-functions and its functionality was tested on a dummy system, in order to verify successful implementation. This dummy system is a Simulink block where all parameters are known, thus enabling testing in a controlled environment. After successful testing, the implementation is modified to fit the system.

2.4 Controller Design

The controller design process consists of an algorithm selection process, implementation and testing, and evaluation. Because the evaluation of the controller algorithm is closely linked with the system identification approach, these are both described in Section 2.5.
2.5 Performance Evaluation

Potential controller algorithms are found in [5, 6] as well as through personal communication with Gustav Lindmark and Daniel Axehill, both at ISY, and Anders Bergmark at Syntronic AB. These discussions are followed up by examining algorithms in primarily [6, 8, 12, 16].

In order to qualify, algorithms must first and foremost be MIMO compatible in order to be considered. Algorithms are coupled with a suitable identification algorithm, because controller selection is dependent on which system parameters that can be obtained.

The algorithms are implemented in Simulink as either S-functions or a series of Simulink blocks, depending on the selected algorithm. Its functionality is tested on a dummy system, in order to verify successful implementation. After successful testing, the implementation is modified to fit the system.

2.5 Performance Evaluation

The implementations of the system identification algorithms and controllers are separately tested and adjusted to fit the system before its combined performance on the system is tested and evaluated. The performance is closely tied to both identification algorithm and controller, and it is not always straightforward to single out whether non-optimal performance primarily depends on selected identification algorithm or controller. This is particularly the case when the controller is explicitly using the identified system parameters to calculate the appropriate control signal.

By studying how output and control signals react on different step responses on the reference signal, it is possible to make assumptions of the origin of the non-optimal performance.
3 System Description

3.1 General Characteristics

The system is a small scale model of a room with multiple arbitrary placed controllable light sources and multiple arbitrary placed light sensors. Both light sources and sensors have a Simulink interface provided by Syntronic. For a schematic overview of the system, see Figure 3.1.

The physical aspects of the system is a cardboard box with 58 LEDs and two sensors, where the LEDs are stationary and the sensors are mobile within the box. For a sketch of the physical system, see Figure 3.2.

The Simulink model has a sampling time $T_S = 0.2$ seconds, meaning that the system is run in 5Hz. The delay for a change in $u(t)$ to be seen in $y(t)$ is $2T_S$.

Figure 3.1: Schematic view of the system with control signal $u(t)$, controlled variable $z(t)$ and measurement signal $y(t)$. 
3.2 LEDs

The LEDs have a "wide scattering angle" [15], implying that the LEDs might have lambertian properties. See Section 4.4 for details. The LEDs can be controlled individually through the Simulink interface, allowing each LED to be assigned an individual 8 bit RGB value. While the LED have RGB capacity, only the illumination is considered in this thesis. Therefore, the R, G and B parts are all assigned the same 8 bit value, allowing the interface to assign each LED a 8 bit value. The control signal \( u_k(t) \in [0, 1], k \in \{1, 2, \ldots, r\} \), which means that each LED receives the 8 bit value

\[
u_{\text{LED}_k} = \lfloor u_k \times 255 \rfloor.
\]

The intensity of the LEDs are controlled through the use of pulse width modulation, PWM, meaning that the LED \( k \) is fed a pulse train that is equal to \( \frac{\lfloor u_k \times 255 \rfloor}{255} \) of the period of time, and equal to 0 otherwise. Thus, the average emitted value for this period will be the perceived analogue value.
3.3 Sensors

The sensors used were Texas Instruments TSL250 and TSL252, which are light-to-voltage optical sensors [7]. In order for PWM to work, the frequency of the LED pulse train shift must be substantially faster than the frequency of the sensor. Therefore, the sensors were equipped with a capacitance in order to form an analogue low-pass filter; this was deemed necessary because the sensors sampled at a frequency that the measured value was at an arbitrarily position, and not as an average value, of the PWM pulse train, which defeats the purpose of the PWM concept. The sensor delivers an 8-bit value, \( y_{\text{sensor}} \), \( l \in \{1, \cdots, p\} \), which is translated to \( y_l \in [0, 1], l \in \{1, \cdots, p\} \), through

\[
y_k = \frac{y_{\text{sensor}_k}}{255}.
\]

The system sampling time \( T_S = 0.2s \) is not primarily limited by the choice of sensor, but by the updating frequency, and PWM pulse period, of the LED. Therefore, in order to upgrade the speed of the system, the first course of action should be to upgrade the LEDs. Higher LED updating frequencies and shorter PWM periods would allow for higher effective sampling frequencies.

3.4 The Limited System

For the purpose of this thesis, the system described in 3.1 was limited in order to reduce the complexity and computational power. The limited system used during the experiments was an 8-input-2-output system, with LED \( u_k(t), k \in \{1, \cdots, 8\} \) and sensor \( y_l(t), l \in \{1, 2\} \) placement according to Figure 3.3. Note that \( y_l(t), l \in \{1, 2\} \), are in the floor plane and \( u_k(t), k \in \{1, \cdots, 8\} \) are in the ceiling plane of the box.
Figure 3.3: Layout of the limited system.
4 Preliminaries

4.1 Recursive Least Squares

Recursive Least Squares estimates the parameter of a linear regression method recursively. Consider the SISO model

\[ y(t) = \varphi^T(t)\theta + v(t), \]  

(4.1)

where

\[ \theta = \begin{bmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_m \end{bmatrix}^T, \]

and

\[ \varphi^T(t) = \begin{bmatrix} -y(t-1) & \cdots & -y(t-n) & u(t-1) & \cdots & u(t-m) \end{bmatrix}. \]

The components of \( \varphi(t) \) are called regressors and the components of \( \theta^T \) are the model parameters. This is merely a different way to express the ARX model presented in [9]:

\[ A(q^{-1})y(t) = B(q^{-1})u(t) + v(t), \]  

(4.2)

where \( A(q^{-1}) \) and \( B(q^{-1}) \) are polynomials of the delay operator \( q^{-1} \) [11]:

\[ A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}, \]
\[ B(q^{-1}) = b_1 q^{-1} + \cdots + b_m q^{-m}. \]
In the MIMO case, where

$$y(t) = \begin{bmatrix} y_1(t) & y_2(t) & \cdots & y_p(t) \end{bmatrix}^T$$

and

$$u(t) = \begin{bmatrix} u_1(t) & u_2(t) & \cdots & u_r(t) \end{bmatrix}^T,$$

the notation in (4.1) can also be used for the MIMO case, but with

$$\theta = \begin{bmatrix} a^{(1)}_{\text{tot}} & a^{(2)}_{\text{tot}} & \cdots & a^{(n)}_{\text{tot}} & b^{(1)}_{\text{tot}} & b^{(2)}_{\text{tot}} & \cdots & b^{(m)}_{\text{tot}} \end{bmatrix}^T$$

(4.3)

where

$$a^i_{\text{tot}} = \begin{bmatrix} a^{(i)}_{1,1} & \cdots & a^{(i)}_{1,p} & a^{(i)}_{2,1} & \cdots & a^{(i)}_{p,p} \end{bmatrix},$$

$$b^i_{\text{tot}} = \begin{bmatrix} b^{(i)}_{1,1} & \cdots & b^{(i)}_{1,r} & b^{(i)}_{2,1} & \cdots & b^{(i)}_{p,r} \end{bmatrix},$$

and

$$\varphi^T(t) = \begin{bmatrix} Y(t-1) & Y(t-2) & \cdots & Y(t-n) & U(t-1) & U(t-2) & \cdots & U(t-m) \end{bmatrix}$$

(4.4)

where

$$Y(t-j) = \begin{bmatrix} -y^T(t-j) & 0 & \cdots & 0 \\ 0 & -y^T(t-j) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & -y^T(t-j) \end{bmatrix},$$

$$U(t-j) = \begin{bmatrix} u^T(t-j) & 0 & \cdots & 0 \\ 0 & u^T(t-j) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & u^T(t-j) \end{bmatrix}.$$
4.1 Recursive Least Squares

\[
A_i = \begin{bmatrix}
(a^{(i)}_{1,1}) & \cdots & (a^{(i)}_{1,p}) \\
\vdots & \ddots & \vdots \\
(a^{(i)}_{p,1}) & \cdots & (a^{(i)}_{p,p})
\end{bmatrix}
\]

\[
B_i = \begin{bmatrix}
(b^{(i)}_{1,1}) & \cdots & (b^{(i)}_{1,r}) \\
\vdots & \ddots & \vdots \\
(b^{(i)}_{r,1}) & \cdots & (b^{(i)}_{r,p})
\end{bmatrix}
\]

combined with the MIMO ARX model

\[
y(t) + A_1 y(t-1) + \cdots + A_n y(t-n) = B_1 u(t-1) + \cdots + B_m u(t-m) + v(t), \quad (4.5)
\]

the linear regression-ARX connection is trivial. However, it is the updating of \(\hat{\theta}(t)\) (the estimator of \(\theta(t)\)) that is the key feature of RLS which is done recursively, i.e. the computational burden does not increase with time. Variations of RLS can be found in [9, 11], but here the weighted version of the MIMO RLS will be presented in (4.6) - (4.8), originally found in [9]. The weighted version allows different weights to be assigned to different time instances, hence modifying the minimization criteria presented in [9], which is done by setting \(\lambda(t)\). \(\lambda(t)\) is combined with the diagonal matrix \(\Lambda_t\), which allows for different weights on different channels. A special case of the weighted version is when \(\lambda(t)\Lambda_t\) equals the identity matrix, which gives the unweighted version of RLS.

\[
\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)[y(t) - \varphi^T(t)\hat{\theta}(t-1)] \quad (4.6)
\]

\[
L(t) = P(t-1)\varphi(t)[\lambda(t)\Lambda_t + \varphi^T(t)P(t-1)\varphi(t)]^{-1} \quad (4.7)
\]

\[
P(t) = \frac{P(t-1) - P(t-1)\varphi(t)[\lambda(t)\Lambda_t + \varphi^T(t)P(t-1)\varphi(t)]^{-1}\varphi^T(t)P(t-1)}{\lambda(t)} \quad (4.8)
\]
4.2 Linear-Quadratic-Gaussian Controller

In this thesis, we consider a discrete-time dynamic system $G$ (see Figure 4.1), described in state space representation as

\[
x(t + 1) = A x(t) + B u(t) + N v_1(t) \quad t \in 0, 1, 2, \cdots, \\
z(t) = M x(t) \quad t \in 0, 1, 2, \cdots, \\
y(t) = C x(t) + v_2(t) \quad t \in 0, 1, 2, \cdots,
\]

with $x(0) = [0 \cdots 0]^T$ [6].

The objective function in the LQG problem is

\[
\text{minimize} \quad \sum z(t)^T Q_1 z(t) + u(t)^T Q_2 u(t) \quad (4.10)
\]

where $\begin{bmatrix} v_1(t) & v_2(t) \end{bmatrix}^T$ is white noise with intensity $\begin{bmatrix} R_1 & R_{12} \\ R_{21} & R_2 \end{bmatrix}$. The objective is to find the feedback $u(t) = -F_y y(t)$ that minimizes (4.10) using the weighting matrices $Q_1$ and $Q_2$ to prioritize between the minimization of $u(t)$ and $z(t)$. Assuming that the controller is causal and contains a delay, the optimal linear controller is

\[
u(t) = -L \hat{x}(t) \\
\hat{x}(t + 1) = A \hat{x}(t) + B u(t) + K \left( y(t) - C \hat{x}(t) \right) \quad (4.11)
\]

where $\hat{x}(0) = x(0)$ and $K$ is defined according to the Kalman filter, that is

\[
P = APA^T + NR_1 N^T - (APC^T + NR_{12})(CPC^T + R_2)^{-1} (APC^T + NR_{12})^T \\
K = (APC^T + NR_{12})(CPC^T + R_2)^{-1} \\
\]

and $L$ is

\[
L = (B^T SB + Q_2)^{-1} B^T SA \\
\]

\[
S = A^T SA + M^T Q_1 M - A^T SB (B^T SB + Q_2)^{-1} B^T SA, \quad (4.13)
\]
according to [6, 16]. In order to expand (4.11) to include a solution to the servo problem, the optimization objective function in (4.10) must be modified [16]. Assume that \(\dim(u(t)) \geq \dim(y(t))\) and that \(u'(r)\) is a defined constant that yield \(z(t) = r(t)\) in a stationary and noise free environment. Then, the modified objective function is

\[
\text{minimize} \quad \sum [r(t) - z(t)]^T Q_1 [r(t) - z(t)] + [u(t) - u'(r)]^T Q_2 [u(t) - u'(r)],
\]

(4.14)

with \(e(t) = r(t) - z(t)\), and the modified controller is

\[
\begin{align*}
up(t) &= -L \hat{x}(t) + L_r r(t) \\
\hat{x}(t+1) &= A \hat{x}(t) + B u(t) + K(y(t) - C \hat{x}(t))
\end{align*}
\]

(4.15)

with

\[
L_r = [M(I + BL - A)^\dagger B]^\dagger
\]

(4.16)

according to [6]. In (4.16), \((\cdot)^\dagger\) denotes the Moore-Penrose pseudoinverse.

### 4.3 Model Predictive Control

Model Predictive Control, MPC, is an expansion of the LQG controller methodology, introducing one major feature: allowing constraints. The controller allows for a limited prediction horizon \(L\), making the basic MPC expansion of (4.10)

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{L-1} x(t + j)^T Q_1 x(t + j) + u(t + j)^T Q_2 u(t + j) \\
\text{subject to} & \quad \text{some constraints.}
\end{align*}
\]

(4.17)

In our case, the system constraint \(u_k(t) \in [0, 1], \forall t\), (4.17) is included in

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{L-1} x(t + j)^T Q_1 x(t + j) + u(t + j)^T Q_2 u(t + j) \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.
\end{align*}
\]

(4.18)

Using the control signal \(u(t)\) on system \(G\) (see Figure 4.1), the state will be updated according to

\[
x(k + 1) = A x(k) + B u(k).
\]

(4.19)
Using (4.19) recursively, (4.20) is obtained.

\[ x(k + 2) = A x(k + 1) + B u(k + 1) \]
\[ = A^2 x(k) + AB u(k) + B u(k + 1). \] \tag{4.20}

Introducing the control signal vector \( U \) and the state vector \( X \), where

\[
U = \begin{bmatrix} u(k) \\ u(k + 1) \\ \vdots \\ u(k + L - 1) \end{bmatrix}
\]

and

\[
X = \begin{bmatrix} x(k) \\ x(k + 1) \\ \vdots \\ x(k + L - 1) \end{bmatrix},
\]

and repeated recursion, the future states can be described as

\[ X = A x(k) + B U, \] \tag{4.21}

with

\[
A = \begin{bmatrix} I \\ A \\ \vdots \\ A^{L-1} \end{bmatrix}
\]

and

\[
B = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ B & 0 & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{L-2}B & \cdots & AB & B & 0 \end{bmatrix}.
\]

Using (4.21) as well as

\[
Q_1 = \begin{bmatrix} Q_1 \\ \vdots \\ Q_1 \end{bmatrix}
\]
4.3 Model Predictive Control

and

\[
Q_2 = \begin{bmatrix}
Q_2 & & \\
& \ddots & \\
& & Q_2
\end{bmatrix},
\]

the objective function of (4.18) is

\[
\sum_{j=0}^{L-1} x(t+j)^T Q_1 x(t+j) + u(t+j)^T Q_2 u(t+j)
\]

\[
= X^T Q_1 X + U^T Q_2 U
\]

\[
= [Ax(t) + BU]^T Q_1 [Ax(t) + BU] + U^T Q_2 U,
\]

and (4.18) can be rewritten as

\[
\begin{align*}
\text{minimize} & \quad [Ax(t) + BU]^T Q_1 [Ax(t) + BU] + U^T Q_2 U \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq U \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad (4.23)
\end{align*}
\]

Similarly to LQG, MPC can be expanded to solve the servo problem. The servo version of (4.18) is

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{L-1} [z(t+j) - r(t+j)]^T Q_1 [z(t+j) - r(t+j)] + \\
& \quad [u(t+j) - u^*(r)]^T Q_2 [u(t+j) - u^*(r)] \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad (4.24)
\end{align*}
\]

which, with the reference signal vector

\[
R = \begin{bmatrix}
r(t) \\
r(t+1) \\
\vdots \\
r(t+L-1)
\end{bmatrix}, \quad U^* = \begin{bmatrix}
\quad u^*(r(t)) \\
\quad u^*(r(t+1)) \\
\quad \vdots \\
\quad u^*(r(t+L-1))
\end{bmatrix}
\]

and

\[
Z = \begin{bmatrix}
Mx(t) \\
Mx(t+1) \\
\vdots \\
Mx(t+L-1)
\end{bmatrix} = \begin{bmatrix}
M \\
\quad M \\
\quad \ddots \\
\quad M
\end{bmatrix} X = MX,
\]
can be rewritten as

\[
\begin{align*}
\text{minimize} & \quad [M(Ax(t) + BU) - R]^T Q_1 [M(Ax(t) + BU) - R] + [U - U^*]^T Q_2 [U - U^*] \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq U \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.
\end{align*}
\]

(4.25)

Note that similarly to LQG, \( u^*(r(t)) \) is a defined constant that yield \( z(t) = r(t) \) in a stationary and noise free environment, that is

\[ u^*(r(t)) = L_r r(t), \]

where \( L_r \) is calculated according to (4.16) [6]. (4.25) can be rewritten as

\[
\begin{align*}
\text{minimize} & \quad [Ax(t) + BU - M^T R]^T \tilde{Q}_1 [Ax(t) + BU - M^T R] + [U - U^*]^T Q_2 [U - U^*] \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq U \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},
\end{align*}
\]

(4.26)

where

\[ \tilde{Q}_1 = M^T Q_1 M. \]

In order to obtain the same optimization as in LQG, an end-weight must be applied to the state \( x(t+N) \), that is (4.26) must be appended with

\[ x(t+N)^T S x(t+N), \]

(4.27)

where \( S \) is the solution to the discrete Riccati equation. The manipulation of (4.25) into (4.26) [2] allows for the state weight matrix \( \tilde{Q}_1 \) instead of the measurement error weight matrix \( Q_1 \). This manipulation is necessary in order to incorporate (4.27) in (4.26). By appending \( A \) and \( B \) by one additional row and appending \( \tilde{Q}_1 \) with a diagonal element \( S \), the optimization formulation in (4.26) can still be used [1, 4]. That means that

\[
A = \begin{bmatrix}
I \\
A \\
\vdots \\
A^L
\end{bmatrix},
\]

\[
B = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
B & 0 & \cdots & 0 \\
AB & B & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
A^{L-1}B & \cdots & AB & B
\end{bmatrix}.
\]
and

\[
\tilde{Q}_1 = \begin{bmatrix}
M^T Q_1 M \\
\vdots \\
M^T Q_1 M \\
S
\end{bmatrix}.
\]

(4.26) can be written on its quadratic form as

\[
\min_U \quad U^T [B^T \tilde{Q}_1 B + Q_2] U + [B \tilde{Q}_1 (Ax(t) - M^\dagger R) - Q_2 U^*]^T U
\]

subject to

\[
\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq U \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},
\]

(4.28)

where all terms non-dependant on \( U \) has been removed as they are not affected by the selection of \( U \).

Contrary to LQG, the MPC controller must solve the optimization problem in real time, giving it a higher computational burden compared to LQG. [3, 4, 14].
4.4 Photometry

Consider Figure 4.2. The source A, with distance $R$ from a surface S (e.g. a sensor), causes the illuminance $E_S$ on S. $E_S$ is dependant on both the intensity of light A radiates in its normal direction as well as the angle $\alpha$ between this normal and surface S. $E_S$ also depends on the angle of incidence, $\beta$, of the light ray on the sensor’s surface. In Figure 4.2, $n_S$ is the normal of surface S.

The light intensity $I_a$ radiated in this angle $\alpha$ is given by Lambert’s cosine law

$$I_a = I_0 \cos \alpha \quad \text{[candela].}$$

(4.29)

Using $I_a$ from (4.29), the expression for $E_S$ is obtained:

$$E_S = \frac{I_a \cos \beta}{R^2} = \frac{I_0 \cos \alpha \cos \beta}{R^2} \quad \text{[lumen/m}^2\text{].}$$

(4.30)

The relationship between the illuminance $E_S$ and the luminous flux $\phi$ from source A on a lambertian surface A is

$$E_S = \frac{d\phi}{dS} \quad \text{[lumen/m}^2\text{].}$$

(4.31)

By combining (4.30) and (4.31), the luminous flux from source A on surface S for a lambertian surface is given:

$$\phi = \int_S \frac{I_0 \cos \alpha \cos \beta}{R^2} dS = I_0 \cos \alpha S \cos \beta \frac{1}{R^2} \quad \text{[lumen],}$$

(4.32)

(4.30)-(4.32) can all be found in [13].
4.5 Feedback and Feed-Forward

The traditional feedback controller $F$, see Figure 4.3, adjusts the control signal $u(t)$ for system $G$ based on the error value $e(t) = r(t) - y(t)$, where $r(t)$ is the reference signal and $y(t)$ is the measured output [5, 6, 16].

![Figure 4.3: Schematic view of a feedback controller. Based on Figure 3.15 in [5].](image)

However, the feedback controller is primarily suited for control around a static $r(t)$. For a servo problem, where $y(t)$ should follow a changing $r(t)$, the feedback controller can be combined with feed-forward from a reference controller [5, 16], see Figure 4.4.

![Figure 4.4: Schematic view of a feedback and feed-forward controller. Based on Figure 7.16 in [5].](image)

Using Figure 4.4, an expression for the control signal $u(t)$ can be derived as

\[
    u(t) = F_f r(t) + F e(t) \\
    = u_{F_f}(t) + u_F(t),
\]

\[ (4.33) \]

\[
    u_{F_f}(t) = F_f r(t), \\
    u_F(t) = F e(t),
\]

a separation also occurring in [16].
4.6 PI Controller

The PI controller drives the feedback part of the control signal \( u(t) \), i.e. \( u_F(t) \) (see (4.33)), based on the error value \( e(t) = r(t) - y(t) \). For a coupled MIMO system, the discrete time PI controller is

\[
\begin{align*}
  u_F(t) &= K \left[ e(t) + \frac{T_S}{T_i} \sum_{j=0}^{t} e(j) \right] = \tilde{K} K_p \left[ e(t) + \frac{T_S}{T_i} I_t \right],
\end{align*}
\]

where \( \tilde{K} \) is a \( \mathbb{R}^{r \times p} \) matrix and \( K_p \) is a diagonal \( \mathbb{R}^{p \times p} \) matrix. This separation is made in order to separate the proportional constant \( K_p \) in the PI controller from the control signal distribution matrix \( \tilde{K} \). An ideal control signal distribution matrix is a pseudo-inverse of system \( G \), i.e. \( G^\dagger \). This is based on the definition of the closed-loop system \( G_C \) [6, 16] for the system in Figure 4.3 as well as (4.35).

\[
\begin{align*}
  u_F(t) &= F e(t) \\
  \text{where} \quad F &= \tilde{K} K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right] \\
\end{align*}
\]

Using (4.35), with \( \tilde{K} = G^\dagger \), and the definition of \( G_C \), (4.36) is obtained:

\[
\begin{align*}
  G_C &= (I + GF)^{-1} GF = \\
  &\begin{cases}
    I \
    \text{with} \quad F = G^\dagger K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right] \\
  \end{cases} \\
  &\begin{cases}
    = (I + G^\dagger K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right])^{-1} G C^\dagger K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right] \\
    = (I + K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right])^{-1} K_p \left[ I + \frac{T_S}{T_i} I_t e^\dagger(t) \right],
  \end{cases}
\end{align*}
\]

which is independent from the system parameters. Note that \( I \) is the identity matrix of appropriate size. \( I_t \) is updated iteratively, which at time \( k \) is

\[
\begin{align*}
  I_k &= I_{k-1} + K_p \frac{T_S}{T_i} e(k) \\
  v_F(k) &= K_p e(k) + I_k \\
  u_F(k) &= \begin{cases}
    u_{F_{\text{max}}}, & \text{if} \quad v_F(k) > u_{F_{\text{max}}} \\
    v_F(k), & \text{if} \quad u_{F_{\text{min}}} \leq v(k) \leq u_{F_{\text{max}}} \\
    u_{F_{\text{min}}}, & \text{if} \quad v(k) < u_{F_{\text{min}}}
  \end{cases} \\
  I_k &:= I_k + \frac{T_S}{T_t} (u(k) - v(k))
\end{align*}
\]
with \( I_0 = 0 \). \( T_i, K_p \) and \( T_i \) are tuning parameters for the PI controller. (4.37) is a PI version of the anti-windup PID controller algorithm presented in [16]. For further details on the PI controller, see [5, 16]. (4.34) can be implemented as \( F \) in Figure 4.3 or 4.4. For the feedback system in Figure 4.3, the limitations of \( u_F(t) \) are the same as the limitations of \( u(t) \), namely

\[
\begin{align*}
    u_{F_{\text{min}}} &= u_{\text{min}} = 0, \\
    u_{F_{\text{max}}} &= u_{\text{max}} = 1.
\end{align*}
\]

For the feedback feed forward system in Figure 4.4, the selection of \( u_{F_{\text{min}}} \) and \( u_{F_{\text{max}}} \) is more complicated, because it depends on the value of \( u_{F_f}(t) \). Therefore, the limitations in this scenario are

\[
\begin{align*}
    u_{F_{\text{min}}} + u_{F_f}(t) &= u_{\text{min}} = 0, \\
    u_{F_{\text{max}}} + u_{F_f}(t) &= u_{\text{max}} = 1.
\end{align*}
\]
4.7 Control Allocation

For a MIMO system, control allocation can be used to distribute control signals in order to reach an output objective. This is done by selecting an \( u(t) \) that minimizes the affine transformation \( f(u(t)) \) in the QP problem

\[
\begin{aligned}
\text{minimize} & \quad f(u(t))^T f(u(t)) \\
\text{subject to} & \quad \min \leq u(t) \leq \max
\end{aligned}
\]  
(4.38)

where \( f(u(t)) \) is some function of \( u(t) \) and with some constraint on \( u(t) \). Now regard Figure 4.5.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{ca_pi_controller.png}
\caption{Schematic view of a Control Allocation PI controller.}
\end{figure}

Because of (4.33) and that \( u_F(t) \) is input to the control allocation, \( u(t) \) can be substituted to \( u_F(t) \) in the objective function in (4.38), and thus making \( f(u_F(t)) \) some affine transformation of \( u_F(t) \). When the error \( e(t) = 0 \), the optimization of \( f(u_F(t)) \) will provide perfect control. The constraint, however, is derived from the saturation of the system input, i.e. \( u(t) \), and therefore \( u(t) \) remains in the criteria as \( u_k(t) \in [0, 1], k \in \{1, 2, \cdots, r\} \).

For an over-actuated system, where \( dim(u(t)) > dim(y(t)) \) (there are more LEDs than sensors), there are several solutions to the problem, allowing for secondary optimization objectives. One example of this is to add preferences on which \( u_k(t) \) to use, which is done by modifying (4.38) to

\[
\begin{aligned}
\text{minimize} & \quad f(u_F(t))^T Q_1 f(u_F(t)) + u_F(t)^T Q_2 u_F(t) \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\end{aligned}
\]  
(4.39)

Here, the relation between \( Q_1 \) and \( Q_2 \) decides which objective is primary and which is secondary, can be done for each \( u_k(t) \). If \( Q_1 \) is larger than \( Q_2 \), focus will be to minimize \( Q_1 \) making it the primary objective, and vice versa [8].
In this chapter, the implementation of two discrete Controller-Identification approaches is derived from the theory in Chapter 4. The first is the dynamic approach, where (4.9) is assumed an adequate system description, and the second is the static approach, where

\[ y(t) = c u(t) + d + v(t) \]  \hspace{1cm} (5.1)

is assumed an adequate description. \( c \) and \( d \) are constants and \( v(t) \) is noise with zero mean, containing system dynamics and other components inexplicable by the estimate

\[ \hat{y}(t) = c u(t) + d. \]  \hspace{1cm} (5.2)

The dynamic approach is a RLS-LQG implementation based on Sections 4.1-4.2 and the implementation is described in Sections 5.1-5.2. The static approach is a Control Allocation-PI, CA-PI, controller coupled with a physical identification scheme, based on Sections 4.4-4.6, and its implementation is described in Sections 5.4-5.5. Some industrial features were also developed, these are presented in Section 5.6.

Note that in the following chapters, the terms Online Identification and Offline Identification occur, which for the dynamic approach also occurs as Online and Offline RLS and for the static approach also occurs as Online and Offline physical identification. These are specific experiments with specific input sequences, please note that these experiments are not the only way to achieve online and offline identification. The main differences between the online and offline schemes used in this thesis are that the offline schemes are free to select \( u(t) \) in any way they
want to achieve more accurate system identification, while the online schemes are designed to work around a system working point. This means that during online identification, the identification is done while a controller tries to maintain a desired output while during offline identification, the controller is disregarded during the entire identification process. The reason that the offline approaches are more or less the same as the online approaches - with the difference of free selection of $u(t)$ - is partly covered in Chapter 2, but also because it provides the user with the option to "reboot" the model parameters without terminating the execution. This does also mean that the use of offline does not mean "previously done" or "done in another execution", it does merely affect how $u(t)$ is selected.
5.1 Recursive Least Squares

5.1.1 General Recursive Least Squares Identification

The MIMO-RLS is implemented in three different MATLAB Level 2 S-function blocks for abstraction purposes, see Figure 5.1. The first block updates the $\varphi^T(t)$ matrix described in (4.4). The updated $\varphi^T(t)$ is forwarded to the second block, where (4.6)-(4.8) are implemented.

The updated $\hat{\theta}(t)$ is forwarded to the third block, where decision on whether to update the system parameters is made. The calculated $\hat{\theta}(t)$ can be anticipated to be volatile, particularly when a sensor is moved or covered, but also because there is noise that $\hat{\theta}(t)$ is unable to estimate. The system state space matrices should only be updated, and hence a new controller calculated, when the system characteristics have changed and not when the difference in $\hat{\theta}(t)$ can be attributed to noise. Therefore, the third block implements a moving $\hat{\theta}(t)$ average of $N$ samples, $AVG_N[\hat{\theta}(t)]$, and only updates the system parameters if both the following conditions are fulfilled

1. $|\hat{\theta}(t) - AVG_N[\hat{\theta}(t)]| < \delta_{\text{avg}}$
2. $|\hat{\theta}(t) - \hat{\theta}_{\text{trans}}| > \delta_{\text{trans}}$

where $\hat{\theta}_{\text{trans}}$ is the model parameters last used to update the system parameters, and $\delta_{\text{avg}}$ and $\delta_{\text{trans}}$ are design parameters. The first condition correlates to that the change must have stabilized around some steady state, meaning that the difference between $AVG_N[\hat{\theta}(t)]$ and $\hat{\theta}(t)$ cannot be too large. The second condition correlates to that the new parameters $\hat{\theta}(t)$ must be significantly different from the current system parameters $\hat{\theta}_{\text{trans}}$, therefore the difference between $\hat{\theta}(t)$ and $\hat{\theta}_{\text{trans}}$ must be sufficiently large. If both conditions are fulfilled, the third block updates the $A,B$ and $C$ parameters in (4.9).

Figure 5.1: RLS block abstraction.
5.1.2 Offline Identification

Offline identification allows for complete control over each $u_k(t)$. According to [9, 10], an ideal $u_k(t)$ for system identification is using a PBRS, i.e. a pseudo-random binary sequence, in order to obtain as much of the dynamic of the system as possible. Therefore, an individual Bernoulli PBRS generator was connected to each $u_k(t)$ during offline identification.

5.1.3 Online Identification

Because RLS recursively uses a set of current and old $u(t)$ and $y(t)$ to obtain and update the model parameters, the key difference between offline and online identification is how $u(t)$ is determined. Instead of manually setting each individual $u_k(t)$, the online implementation uses whatever $u(t)$ the controller selects and estimates model parameters based on these $u(t)$ combined with the resulting $y(t)$. With this implementation, a true online identification occurs because the control signal is not modified. The trade-off is that some system dynamics are likely to be omitted in the model parameters.
5.2 LQG

The controller described in (4.15) was implemented as a LTI system block (Linear Time-Invariant) for the state update and matrix multiplication blocks for the creation of \( u(t) \). \( u(t) \) is then saturated to \( u(t) \in [0, 1] \), in accordance with the limitations described in Chapter 3, before being sent to the system and fed back to the LTI system block. See Figure 5.2 for Simulink implementation. Note that the Mux block in Figure 5.2 concatenates \( u(t) \) and \( y(t) \) to \( [u(t)^T \ y(t)^T]^T \), because how the LTI system block is implemented. This is merely a implementation detail, as the theory is still intact.

\[
\begin{align*}
K, L \text{ and } L_r & \text{ are computed at start-up and when system parameters are updated (see 5.1). } K \text{ is computed using } \text{kalman, } L \text{ using } \text{dlqr and } \\
L_r & = [C(I + BL - A)^\dagger B]^\dagger, \\
\text{in accordance with [6]. } R_1 \text{ and } R_2 & \text{ are estimated prior to execution and set at start-up.}
\end{align*}
\]

Note that the LQG controller requires initial state space matrices in order to compute the controller at start-up.
5.3 MPC

The purpose of the MPC controller implementation is to assert that the performance of the LQG controller is not flawed because of its inability to account for control signal constraints. Therefore, the MPC controller is implemented using the same parameters as the LQG controller, and is used only for the offline cases, meaning that it is not coupled with online RLS. The controller described in (4.28) is implemented with MATLAB’s quadprog in a MATLAB Level 2 S-function in Simulink. The future reference vector $R$ is assumed to be constant for all elements, i.e. the reference signals are assumed constant. The prediction horizon $L$ is selected as long enough to contain all system dynamics, during the simulation $L$ was selected to 50 samples because of the aim that the MPC controller should act like the LQG controller with the control signal constraint. In order to compare the LQG and MPC controller, the MPC observer is chosen to the same observer as for the LQG controller, i.e. according to (4.15).
5.4 Physical Identification

5.4.1 General Physical Identification

Using photometry, described in Section 4.4, one realizes that for a stationary system, where \( \alpha, \beta, R \) and \( S \) are constant, (4.32) can be written as (5.3)

\[
\phi_i = I_j \frac{k_{i,j}}{R_{i,j}^2} = I_j K_{i,j} \quad [\text{lumen}],
\]

for a sensor \( \phi_i, i \in \{1 \cdots p\} \), a source \( I_j, j \in \{1, \cdots, r\} \), with \( K_{i,j} \) as a constant when LEDs and sensors are stationary. Because illuminance is additive, the illuminance level on sensor \( i \) from \( r \) sources each emitting \( I_j \) candela is

\[
\phi_i = \sum_{j=1}^{r} K_{i,j} I_j,
\]

where \( v(t) \) is background light and noise. With \( y_i(t) = \phi_i(t) \) and \( u_j(t) = I_j(t) \), (5.1) and (5.4) can be rewritten as

\[
y(t) = \begin{bmatrix} K_{1,1} & \cdots & K_{1,r} \\ \vdots & \ddots & \vdots \\ K_{p,1} & \cdots & K_{p,r} \end{bmatrix} u(t) + d + v(t) \\
= c u(t) + d + v(t),
\]

where

\[
c = \begin{bmatrix} K_{1,1} & \cdots & K_{1,r} \\ \vdots & \ddots & \vdots \\ K_{p,1} & \cdots & K_{p,r} \end{bmatrix}
\]

The estimate of (5.5) is therefore

\[
\hat{y}(t) = \begin{bmatrix} K_{1,1} & \cdots & K_{1,r} \\ \vdots & \ddots & \vdots \\ K_{p,1} & \cdots & K_{p,r} \end{bmatrix} u(t) + d = c u(t) + d,
\]

which is on the form (5.2) and is a first order polynomial in \( u(t) \).

5.4.2 Offline Identification

The offline identification uses (5.4) to sequentially identify each \( K_{i,j} \). This is done by collecting \( N \) samples from each of \( M \) intensities of \( u(t) \) for each LED one at
the time, while remaining LEDs are set to zero. $y(t)$ is measured simultaneously, resulting in an identification time of $N \times M \times r \times T_S$ seconds. For each intensity, the $N$ corresponding samples are averaged, resulting in $M$ data points for each LED for polynomial fit. The reason for this procedure is to isolate each channel between each $u_k(t)$, $k \in \{1, \cdots , 8\}$ and $y_l(t)$, $l \in \{1, \cdots , 2\}$ so all contributions from remaining $u_k(t)$, are eliminated.

When data from all sensors has been gathered, all polynomials are fitted, resulting in the $c$ matrix in (5.5). The $d$ matrix is calculated as a mean of all samples when all LEDs are turned off.

### 5.4.3 Online Identification

Instead of shutting all other LEDs down during identification as in Section 5.4.2, the online identification aims to identify the system around a operating point. Analogously to the methodology described in Section 5.4.2, each $u_j(t)$, $j \in \{1, \cdots , r\}$ is gradually changed while remaining $u_i(t)$, $j \in \{1, \cdots , r\}, i \neq j$ are constant. However, during online identification, $u_i(t)$ are not necessarily equal to zero.

Before LED $j$ is evaluated, the $u(t)$ chosen by the controller is stored. Using (5.4), the contribution from all other LEDs is determined by selecting $u_j(t) = 0$ and averaging over $N$ samples. $u_i(t + l)$, $i \neq j$, $l \in \{1, \cdots , N \times M\}$, is constant for the entire evaluation of the sensor, and the contribution from $u_i(t + l)$ value is then subtracted from every $y(t + l)$ when evaluating LED $j$. After each LED, the $c$ matrix is updated and the controller controls the system for a, by the user selected number, of samples before the next LED is evaluated, in order to let the controller find a new operating point. The $d$ matrix is updated after all LEDs are updated, analogously with Section 5.4.2.

### 5.4.4 Implementation Specifics

Measurements of $u(t)$ and $y(t)$ gathered according to the online or offline methods described in Sections 5.4.2 and 5.4.3 are used to fit a first order polynomial in order to estimate $c$ and $d$. In MATLAB, this was done using \texttt{polyfit}.

Both the online and offline methods described in Sections 5.4.2- 5.4.3 were implemented in Simulink as Level 2 S-functions, because these S-functions allow the allocation of the memory needed to store a large number of variables from different sample times as well as the possibility to write the code needed to implement these identification algorithms.
5.5 Control Allocation PI Controller

Inserting (4.33) in (5.6), gives

\[
\hat{y}(t) = c u(t) + d = c u_{F_f}(t) + c u_{F}(t) + d \\
= \hat{y}_{F_f}(t) + \hat{y}_{F}(t),
\]

\[(5.7)\]

\[
\hat{y}_{F_f}(t) = c u_{F_f}(t) + d, \\
\hat{y}_{F}(t) = c u_{F}(t).
\]

Using (4.39) and selecting \(f(u_{F_f}(t))\) as the difference between the reference signal \(r(t)\) and \(\hat{y}_{F_f}(t)\) in (5.7), (5.8) is derived.

\[
\text{minimize} \quad \left[ r(t) - (c u_{F_f}(t) + d) \right]^T \left[ r(t) - (c u_{F_f}(t) + d) \right] \\
\text{subject to} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u(t) = u_{F_f}(t) + u_{F}(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\]

\[(5.8)\]

Non-contributing LEDs should not be used during optimization. Therefore, \(K_{i,j}\) (see (5.6)) virtually zero shouldn’t be used during optimization. This was achieved by modifying the \(c\) matrix in (5.5) and introducing \(\tilde{c}\), where almost-zero elements are set to zero. The exact threshold is a design parameter, but was set to 0.01 during these experiments.

\[
\text{minimize} \quad \left[ r(t) - (\tilde{c} u_{F_f}(t) + d) \right]^T \left[ r(t) - (\tilde{c} u_{F_f}(t) + d) \right] \\
\text{subject to} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u_{F_f}(t) + u_{F}(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\]

\[(5.9)\]

For rows \(k\) in \(\tilde{c}\) where all elements are zero, the corresponding \(u_k(t)\) can be considered redundant and should therefore be forced to \(u_k(t) = 0\). This is achieved by introducing a secondary optimization objective according to (4.39):

\[
\text{minimize} \quad \left[ r(t) - (\tilde{c} u_{F_f}(t) + d) \right]^T Q_1 \left[ r(t) - (\tilde{c} u_{F_f}(t) + d) \right] + u_{F_f}(t)^T Q_2 u_{F_f}(t) \\
\text{subject to} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u_{F_f}(t) + u_{F}(t) \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\]
where

\[ Q_1 = \begin{bmatrix}
  g_1 & 0 & \cdots & \cdots & 0 \\
  0 & g_2 & 0 & \cdots & 0 \\
  \vdots & \ddots & \ddots & \ddots & \vdots \\
  0 & \cdots & \cdots & \cdots & g_p \\
\end{bmatrix}, \]

\[ Q_2 = \begin{bmatrix}
  h_1 & 0 & \cdots & \cdots & 0 \\
  0 & h_2 & 0 & \cdots & 0 \\
  \vdots & \ddots & \ddots & \ddots & \vdots \\
  0 & \cdots & \cdots & \cdots & h_r \\
\end{bmatrix}. \]

with

\[ g_i = g_j, \ \forall i, j \in \{1, \cdots, p\} \]

\[ h_k = H, \ H >> g_1, \cdots, g_p \]

\[ h_i = 0, \ \forall i \neq k. \]

The reason for \( h_k = H, \ H >> g_1, \cdots, g_p \) is to make the use of only non-redundant \( u_i(t) \) the primary optimization objective. The redundant \( u_k(t) \) contribute to noise and unmodeled behaviour and have no cost in the other optimization objective, and must therefore be suppressed. Note that the use of the \( \tilde{c} \) matrix instead of the \( c \) matrix avoids the strong bias that otherwise would be obtained through implementing (5.11).

The objective function is now optimizing the control allocation block in Figure 4.5 using only non-redundant \( u_i(t) \). Thus far, the constraint has used \( u(t) \), which is the sum of the control signal contributions from both the control allocation and feedback (F) blocks. By combining (4.33) and (4.34), (5.14) is obtained. By selecting \( \tilde{K} \) as the Moore-Penrose pseudoinverse of \( \tilde{c} \), i.e. \( \tilde{c}^\dagger \), appropriate dimensions and characteristics are obtained. This choice is motivated through an expansion of (5.6):

\[ \hat{y}(t) = c \ u(t) + d \]

\[ = c \ u_{f_f}(t) + c \ u_{f}(t) \]

\[ = c \ u_{f_f}(t) + c \ \tilde{K} K_p [I_t + e(t)] \]

\[ = c \ u_{f_f}(t) + c \ \tilde{c}^\dagger K_p [I_t + e(t)] \] (5.12)

thus virtually removing the system parameters from the PI controller (see (4.36) for the general case). The reason why \( \tilde{c}^\dagger \) is chosen instead of \( c^\dagger \) (that would cancel the impact of system parameters altogether) is because \( c^\dagger \) would allow the PI
controller to use redundant \( u^k(t) \) whereas \( \tilde{c}^\dagger \) does not. By modifying (5.10)-(5.11) accordingly, (5.13) is obtained.

\[
\begin{align*}
\text{minimize} & \quad [r(t) - (\tilde{c} u^{F_f}(t) + d)]^T Q_1 [r(t) - (\tilde{c} u^{F_f}(t) + d)] + u^{F_f}(t)^T Q_2 u^{F_f}(t) \\
\text{subject to} & \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leq u^{F_f}(t) + \tilde{c}^\dagger K_p [I_t + e(t)] \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
& \quad (5.13)
\end{align*}
\]

The control signal to the system is thereafter selected as

\[
u(t) = u^{F_f}(t) + \tilde{c}^\dagger K_p [I_t + e(t)]
\]

(5.14)

In other words, this implementation optimizes the control signal distribution using the PI controller as a constraint. Both the control allocation and the feedback controller was implemented using MATLAB Level 2 S-function together with an anti-windup solution for the PI controller. The optimization problem was set up and calculated using YALMIP, optimization toolbox for Matlab [12].
5.6 Industrial Features

A number of industrial features were implemented, the most important ones are presented here.

Reference limits

For the physical identification implementation, a theoretical higher illuminance level for each sensor was calculated and presented to the user after each update of $\tilde{c}$ (see Section 5.2). The maximum reference for sensor $k$ is calculated as

$$r_{k_{\text{max}}} (t) = \sum_{i=1}^{r} \tilde{c}_{k,i},$$  \hspace{1cm} (5.15)

which equals that non-redundant LEDs $i$ are all given the control signal $u_i(t) = 1$, while redundant LEDs $j$ are given $u_j(t) = 0$.

Reference toggle

The reference signal $r(t)$ can either be set manually or generated through a reference generator, which is a sum of pulse generators of different phase and period for each $r_k(t)$. The user can switch between these modes seamlessly during execution.

Identification toggle

The user can during simulation toggle whether or not system identification should be done during simulation. The user can also toggle whether online or offline identification should be used (see Sections 5.1 and 5.4). The user can select an online identification frequency, is informed of a maximum frequency based on chosen parameter values, and is also alerted when identification is in progress.
In this chapter, the performance of the implementations of Chapter 5 are tested. The results of the dynamic approach, implemented in Sections 5.1 and 5.2, are found in Section 6.1; the results of the static approach, implemented in Sections 5.4 and 5.5, are found in Section 6.2.

In order to adequately evaluate how well different controllers perform in the task of following a reference signal, the experimental setup of the system must be identical. Therefore, the sensors in the system were identically placed and did not change during the experiments, see Figure 3.3 for LED and sensor placement. Also, a step function reference signal $r_{\text{eval}}(t)$ was generated for each $y_k(t)$ in order to achieve identical reference signals. Note that $r(t) = [r_1(t) \; r_2(t)]^T$ is the reference signal of $y(t) = [y_1(t) \; y_2(t)]^T$. 
6.1 Dynamic Approach

The offline RLS, implemented as described in Section 5.1.2, uses a PBRS $u(t)$. The exact values for each $u_k(t), k \in \{1, \cdots, 8\}$ are not the focus of this experiment, which is why they are not shown in a separate figure. Because of the PBRS properties of $u(t)$, the measured signal, $y(t)$, should have pseudo-random properties. Random input should yield random output. Studying Figure 6.1, one can see that $y_1(t)$ and $y_2(t)$ are focused around two levels each. For $y_1(t)$, it is approximately 0.02 and 0.08, and for $y_2(t)$ it is approximately 0.02 and 0.06. This behaviour indicates the binary properties of PBRS. Furthermore, it indicates that one or a few LEDs have the majority of the impact on the sensor, because if all LEDs had similar impact, $y(t)$ would likely be more evenly distributed in respective range.

![Offline RLS - y(t)](image)

*Figure 6.1: y(t) of offline RLS*

The identified model parameters are converted into a state space model, and from this the LQG parameters are calculated, see Appendix A.1 and A.2 for numerical values. The LQG controller, implemented according to Section 5.2, was then performance tested by letting the controller follow $r_{\text{eval}}(t)$. The unsaturated control signals for this experiment are shown in Figure 6.2. Note that the term LQG on Offline RLS refers to the fact that the LQG controller is calculated from model parameters obtained through a prior Offline RLS experiment, i.e. a RLS identification experiment where the input can be chosen freely (see introduction to Chapter 5).

Remembering that $u(t) \in [0, 1]$, one can notice that several $u_k(t) < 0$ will be saturated as well as some $u_k(t)$ will occasionally be larger than 1. This might implicate suboptimal performance, which is shown in Figure 6.3. As expected, one can see that the LQG controller on offline RLS data was unable to follow $r_{\text{eval}}(t)$ satisfactory. This can partly be attributed the saturation of $u(t)$, because it is physically impossible to produce negative light and negative control signals are therefore truncated to 0. However, by studying $y(t)$ and $u(t)$ one can assume that...
6.1 Dynamic Approach

**Figure 6.2:** $u(t)$ of LQG on offline RLS

**Figure 6.3:** $r(t)$ and $y(t)$ of LQG on offline RLS

the inability to satisfactory $r_{\text{eval}}(t)$ following can also partly be contributed an impreciseness in the model.

By using the offline estimated parameters and also allowing the RLS to be executed online, implemented as in Section 5.1.3, the model parameters (and, hence, the controller parameters) can be tuned during execution. By setting up the same test environment and letting the RLS-LQG controller follow $r_{\text{eval}}(t)$, one can see in Figure 6.4 that the parameters are updated six times during the execution. Each spike represents a parameter update. Note that the RLS and system update algorithms are active during the entire experiment.

However, both $u(t)$ (Figure 6.5) and $y(t)$ (Figure 6.6) are virtually the same in the LQG on offline RLS (see Figure 6.2 and Figure 6.3, respectively); if anything, the
online LQG RLS have a slightly more noisy $y(t)$ than its offline counterpart. This means that these parameter updates merely causes minor changes compared to the error $e(t) = r(t) - y(t)$, which suggests that the RLS is not a very effective way to describe the system. It might also suggests that the RLS converges quickly, giving future updates only a marginal effect.

The LQG controller suffered from saturated control signals (see Figures 6.2 and 6.5), potentially harming the reference following performance. The implementation of the MPC controller, allows the addition of control signal constraints to a controller of LQG type. First, an MPC controller without constraints is implemented to verify the implementation. Both the control signals (see Figure 6.7) and the measurement signals (see Figure 6.8) are the same (except for the system
6.1 Dynamic Approach

Figure 6.6: $r(t)$ and $y(t)$ of LQG with online RLS

Introducing the MPC controller with control signal constraints (see Figures 6.9 and 6.10), one notices major similarities with the measurement signals of the LQG and the unconstrained MPC controllers (Figures 6.3 and 6.8). However, the control signals are within the bounds specified in Section 3.2.
Apparently, the optimization objective formulation is ill-suited for the application of reference following for this system, both for the LQG and for the MPC controller. Including the control signal constraints to avoid saturation did not improve the controller performance, therefore the conclusion is that the optimization objective is ill-suited for the application. The optimization objectives are formulated through system parameters for both controllers, so the cause is likely to be inexact or volatile system parameters.

The ARX A- and B-matrices show how the $y(t)$ depends on previous $y(t)$ and $u(t)$. One can say that the A-matrix describes the "inertia" of the system and the B-matrix describes how control signals from different time instants propagates to $y(t)$. By studying Figure 6.11, one can see that the confidence interval is big compared to the mean of the parameter, and also that the sign of the parameter
is non-uniform within many confidence intervals. These are clear signs of uncertainty in the parameter estimation. By studying Figure 6.12, one can see the same pattern of uncertain parameter estimation as for the A-matrix concerning sign non-uniformity, except for order 1. However, because of the system delay of $2T_S$ (see Chapter 3), any direct-term (order 0) or order 1 parameter can be considered obsolete.

However, it still remains unclear whether it is the ARX that fails or if it is the RLS that fails to properly obtain all the information. In order to assert which of these are true, the control signals and measurement output of the offline RLS experiment (see Figure 6.1) is used to find an offline ARX model. This is done through MATLAB’s System Identification Toolbox using the first half of the experiment as model data and the second half as verification data. This data-set is suitable because of the pseudo-random binary sequence, PBRS, of the control sig-
Figure 6.12: Mean and 95% confidence interval for transposed ARX B-matrix

A good model fit in System Identification Toolbox approaches the value 100, whereas a poor model fit is small positive or negative. The results for different offline ARX models are presented in Table 6.1.
Table 6.1: Order of A-matrix (na) order of B-matrix (ba) and delay (nk)

<table>
<thead>
<tr>
<th>ARX model name</th>
<th>na</th>
<th>nb</th>
<th>nk</th>
<th>$y_1(t)$ fit</th>
<th>$y_2(t)$ fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>arx110</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-38.06</td>
<td>-9.542</td>
</tr>
<tr>
<td>arx112</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-31.90</td>
<td>-1.916</td>
</tr>
<tr>
<td>arx113</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-13.55</td>
<td>-0.347</td>
</tr>
<tr>
<td>arx114</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>-3.966</td>
<td>-8.487 · 10^{-3}</td>
</tr>
<tr>
<td>arx115</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>-1.495</td>
<td>-1.552</td>
</tr>
<tr>
<td>arx116</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>-1.677</td>
<td>-1.346</td>
</tr>
<tr>
<td>arx220</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>-32.57</td>
<td>-43.28</td>
</tr>
<tr>
<td>arx222</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-30.15</td>
<td>-18.40</td>
</tr>
<tr>
<td>arx226</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>-13.01</td>
<td>-17.138</td>
</tr>
<tr>
<td>arx330</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>arx332</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>-1.036 · 10^{198}</td>
<td>-1.63 · 10^{198}</td>
</tr>
<tr>
<td>arx336</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>-106.5</td>
<td>-397.1</td>
</tr>
<tr>
<td>arx440</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>arx442</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>arx446</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>-262.3</td>
<td>-820.7</td>
</tr>
</tbody>
</table>

As one can observe in Table 6.1, all examined models have very poor model fit and models of higher order even provides a Not-a-Number (NaN). The linear auto-regressive ARX is clearly insufficient in describing the system. Because the performance is so extremely weak, a reasonable conclusion is that the system has very weak linear dynamic properties, which explains the poor performance of the LQG and MPC controllers. Note that this conclusion does not necessarily imply that all lighting systems are non-dynamic, it merely implies that the system described in Chapter 3, which also includes the sample time $T_S = 0.2s$, appears linearly non-dynamic. Lighting systems with significantly higher sampling frequency might respond differently to dynamic models.
6.2 Static Approach

The offline physical identification, implemented as in Section 5.4 and Section 5.4.2, steps through each $u_k(t)$ while the remaining control signals are zero. In the experiment, the Control Allocation, CA, PI controller is allowed to control a previously identified system during $t \in [0, 20]$ and at $t = 20$ the offline identification starts, see Figure 6.13. The reason for this is to ensure that the identification can be successfully started mid-execution and to ensure it can resume reference following once the experiment is over.

![Offline physical Identification - u(t)](image)

**Figure 6.13:** $u(t)$ of offline physical identification

The resulting $y(t)$ is shown in Figure 6.14, where one can clearly see how each sensor reacts to each control signal, ignoring $r(t)$ while identification is executed. One can also see that the controller manages to follow the reference after the identification experiment is finished.

![Offline physical Identification - r(t) & y(t)](image)

**Figure 6.14:** $r(t)$ and $y(t)$ of offline physical identification
Similarly, for the online physical identification experiment, the CA-PI controller controlled a previously identified system during \( t \in [0, 20] \). At \( t = 20 \), the online identification alters \( u_k(t) \) while remaining control signals are stationary, see Figure 6.15.

\[
\text{Online physical Identification - } u(t)
\]

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{online_physical_identification_u_t.png}
\caption{\( u(t) \) of online physical identification}
\end{figure}

Here, the identification algorithm aims to keep the system at a constant reference signal and the resulting relationship between \( y(t) \) and \( r(t) \) can be seen in Figure 6.16. The identification can satisfactory be performed around a working point. For numerical values on \( c, \hat{c} \) and \( d \) for both online and offline identification experiments, see Appendix A.3.

\[
\text{Online physical Identification - } r(t) \text{ & } y(t)
\]

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{online_physical_identification_r_y.png}
\caption{\( r(t) \) and \( y(t) \) of online physical identification}
\end{figure}

To evaluate the CA-PI controller, the controller, with precisely offline estimated system parameters, is following the reference signal \( r_{\text{eval}}(t) \). One can see in Fig-
ure 6.17 that solely \( u_1(t), u_3(t), u_5(t) \) and \( u_6(t) \) were used to control the system, which corresponds to the non-zero columns in \( \tilde{c} \) (see (5.9) and Appendix A.3); control signals \( u_3(t), u_4(t), u_7(t) \) and \( u_8(t) \) are therefore redundant.

![Control Allocation reference following - u(t)](image)

**Figure 6.17:** \( u(t) \) of control allocation reference following

The resulting relationship between \( r(t) \) and \( y(t) \), can be seen in Figure 6.18 and one can see that the CA-PI controller satisfactorily controls the system. The feed forward (CA) link is static in itself; this means that in the absence of a PI controller (i.e. an open-loop CA controller), \( u(t) = u_{F} \) is constant for a selected \( r(t) \) when the system parameters are constant, which would yield step-like \( u_k(t), k \in \{1, \cdots, 8\} \), and \( y_l(t), l \in \{1, 2\} \). Thus, the smoothness of \( u_k(t), k \in \{1, \cdots, 8\} \) is attributed to the I-part of the PI controller as well as the perceived dynamic of \( y_l(t), l \in \{1, 2\} \). The P-part of the PI controller is one of the causes of the noisy behavior of \( y_l(t), l \in \{1, 2\} \) around its steady state, however the disturbances (noise) of the system should not be neglected.
Figure 6.18: $r(t)$ and $y(t)$ of control allocation reference following

Control Allocation reference following - $r(t) \& y(t)$
Conclusions

7.1 System Properties

The system described in Section 3 cannot be accurately modeled using a linear auto-regression ARX model, neither offline, using System Identification Toolbox, nor online, using the Recursive Least Square, RLS, algorithm. The system can, however, be accurately described using a static affine transformation. One can therefore draw the conclusion that the system has predominantly static properties and inferior dynamic properties. This is the case for the current system, which includes the possible sampling time $T_S$. Note that the system is limited by the LED specifications, and upgrading the LED would allow for faster dynamics to be investigated. If the dynamics of the system is much faster than $T_S$, the system will appear static. Therefore, the conclusion of the system’s inferior dynamic properties cannot be transmitted to all lighting systems; in order to do so, thorough research of the high speed dynamics of lighting systems must be done. However, given the physical properties of light, the lighting-dynamic null hypothesis will likely be difficult to reject.

7.2 Control Signal Selection

The beforehand promising LQG controller turned out to be ill-suited to control the system. The hypothesis that the major cause of the poor performance was the controller’s inability to account for control signal constraints was rejected as the MPC controller, implemented using the same parameters as the LQG controller, also was ill-suited to control the system. However, the low quality parameter estimation (see Section 7.1), remains as the most probable factor for the LQG and MPC controller failure. If a significantly faster lighting system would have stronger dynamic properties, both controllers would be expected to perform better. How-
ever, for this hypothetical high-speed dynamic system, the MPC controller might be too computational expensive for this application because the MPC optimization problem must be solved for each sample. The LQG controller, however, have low computational burden but may still have problem with the saturation of the control signals. If the results of this thesis are to be used on a different system, with more dynamic properties (e.g. temperature distribution), the LQG and MPC controllers are likely to perform better than in this thesis.

The proposed static Control Allocation, CA, PI controller is, on the other hand, well suited to control the system. It also successfully manages to suppress redundant control signals, which is beneficial both for lowering disturbances and for lowering energy consumption. The use of the PI controller as a constraint in the CA optimization problem is more computationally expensive than having disintegrated serial CA and PI sections, because then the CA would only be recalculated at change of reference signals or system parameters. However, the proposed integrated CA-PI is a more holistically and theoretically sound approach if computational resources are available. If computational resources are scarce, a disintegrated CA-PI controller might be well suited. Another interesting future adaptation would be to implement the CA-PI controller on a more dynamic system, however, in this scenario an expansion to a CA-PID controller might be beneficial to decrease overshoot.

At first glance, the LQG controller share some similarities the CA-PI controller. Both approaches tries to minimize sum of the difference of a reference and a measurement signal estimate and the control signal, using quadratic programming. Aside from the differences in dynamicity, a key difference is where the feedback is considered. In the LQG controller, the error is part of the optimization function, see (4.15), meaning that the selection of weight matrices defines whether small measurement error or small control signals should be prioritized during optimization. The CA-PI controller, on the other hand, uses the measurement error in the constraint, meaning that there is no trade-off and that the optimization result is very much dependant on the value of I-part in each instant of time. The user instead have to modify the PI controller parameters, and thus indirectly adjust whether the feedback or feed-forward part of the controller should be prioritized. In other words, the two approaches contain similar theoretical optimization components, but are assembled differently and thus obtaining different tuning characteristics. Note again that the different controllers have very different performances for systems that are predominantly static and predominantly dynamic characteristics.

7.3 Conclusion Summary

This thesis has conducted the following conclusions:

1. The CA-PI controller is well suited to control the system, while the LQG and MPC controllers are not.
2. The static properties of the system are predominant, and a purely dynamic model is therefore insufficient to describe the system.

3. In case of system high-speed dynamics, the dynamic approaches might perform better on faster systems.

4. Even though both approaches contain similar components, the differences in structure give very different properties and tuning possibilities.

5. For a more dynamic system, an LQG or MPC controller would likely be better suited to solve the problem.

7.4 Future Research

The following future research is proposed:

1. Upgrading to faster LEDs would allow for research whether high-dynamics are prevalent in the system.

2. Adaptation of the CA-PI on other static systems.

3. Investigate how the CA-PI controller performs on dynamic systems.

4. Evaluation of different static identification algorithms on the system.

5. Evaluation of non-linear dynamic identification algorithms on the system.
## A.1 RLS System Parameters

\[
A = \begin{bmatrix}
0.2597 & 0.3207 & 1.0000 & 0 \\
0.2852 & 0.5036 & 0 & 1.0000 \\
-0.2766 & -0.3233 & 0 & 0 \\
-0.1798 & -0.6189 & 0 & 0 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0.0032 & 0.0111 & 0.0294 & 0.0003 & -0.0052 & 0.0300 & -0.0034 & -0.0074 \\
0.0016 & 0.0048 & 0.0404 & 0.0122 & 0.0014 & 0.0407 & -0.0068 & -0.0069 \\
-0.0031 & -0.0001 & -0.0215 & 0.0029 & -0.0077 & -0.0161 & -0.0005 & 0.0046 \\
-0.0014 & -0.0007 & -0.0144 & 0.0079 & -0.0105 & -0.0317 & -0.0030 & 0.0069 \\
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
0.0130 & -0.0013 & 0.0765 & 0.0066 & 0.0118 & -0.0028 & -0.0056 & -0.0054 \\
-0.0014 & 0.0015 & 0.0011 & -0.0148 & 0.0136 & 0.0520 & 0.0064 & -0.0096 \\
\end{bmatrix}
\]
A.2 LQG Parameters

\[ K = 10^{-3} \times \begin{bmatrix} 0.1033 & 0.1273 \\ 0.1366 & 0.1374 \\ -0.1393 & -0.2123 \\ -0.1878 & -0.3231 \end{bmatrix} \]

\[ L = \begin{bmatrix} 0.0059 & 0.0101 & 0.0043 & 0.0061 \\ 0.0102 & 0.0117 & 0.0234 & 0.0285 \\ 0.0748 & 0.1180 & 0.0719 & 0.1468 \\ 0.0056 & -0.0059 & 0.0236 & 0.0533 \\ 0.0098 & 0.0322 & -0.0206 & -0.0170 \\ 0.0804 & 0.1498 & 0.0593 & 0.1295 \\ -0.0061 & -0.0023 & -0.0166 & -0.0310 \\ -0.0168 & -0.0319 & -0.0116 & -0.0221 \end{bmatrix} \]


A.3 Physical Identification Parameters

Offline Identification

\[ c = \begin{bmatrix} 0.0112 & 0.0037 & 0.0677 & 0.0013 & 0.0050 & -0.0000 & -0.0001 & -0.0000 \\ -0.0019 & 0.0001 & 0.0054 & 0.0046 & 0.0120 & 0.0497 & 0.0021 & -0.0004 \end{bmatrix} \]

\[ \tilde{c} = \begin{bmatrix} 0.0112 & 0 & 0.0677 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0120 & 0.0497 & 0 & 0 \end{bmatrix} \]

\[ d = 10^{-3} \times \begin{bmatrix} 0.6401 \\ 0.3921 \end{bmatrix} \]

Online Identification (end of identification scheme)

\[ c = \begin{bmatrix} 0.0143 & 0.0036 & 0.0592 & 0.0066 & 0.0031 & 0.0005 & 0.0005 & 0.0003 \\ -0.0001 & -0.0001 & 0.0005 & 0.0006 & 0.0095 & 0.0455 & 0.0053 & 0.0000 \end{bmatrix} \]

\[ \tilde{c} = \begin{bmatrix} 0.0143 & 0 & 0.0592 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0455 & 0 & 0 \end{bmatrix} \]

\[ d = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]


