SLOVAK UNIVERSITY OF TECHNOLOGY
IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-5414-91950

# Distributed Model Predictive Control Design for Plants of Chemical Industry

**MASTER THESIS**

**2022** **Bc. Erika Pavlovičová**

**SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA**

**FACULTY OF CHEMICAL AND FOOD TECHNOLOGY**

Reg. No.: FCHPT-5414-91950

# Distributed Model Predictive Control Design for Plants of Chemical Industry

**MASTER THESIS**

| | |
|---|---|
| Study programme: | Process Control |
| Study field: | Cybernetics |
| Training workspace: | Institute of Information Engineering, Automation, and Mathematics |
| Thesis supervisor: | doc. Ing. Juraj Oravec, PhD. |

**2022** **Bc. Erika Pavlovičová**

Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology
Academic year: 2021/2022
Reg. No.: FCHPT-5414-91950

STU FCHPT

# MASTER THESIS TOPIC

Student: **Bc. Erika Pavlovičová**
Student's ID: 91950
Study programme: Automation and Information Engineering in Chemistry and Food Industry
Study field: Cybernetics
Thesis supervisor: doc. Ing. Juraj Oravec, PhD.
Head of department: prof. Ing. Michal Kvasnica, PhD.
Workplace: Institute of Information Engineering, Automation, and Mathematics, FCHPT STU

Topic: **Distributed Model Predictive Control Design for Plants of Chemical Industry**

Language of thesis: English

Specification of Assignment:

The aim of this master thesis is to design a model predictive control using a distributed optimization for a plant in the chemical industry. Particular aims are: (i) analysis of the model structure suitable for distribution, (ii) design decoupled optimization problems, (iii) design coupled optimization problem, (iv) build algorithm for iterative enumeration of control actions, and (v) analyse the control performance.

Length of thesis: 50

Selected bibliography:

1. B. Houska – J. Frasch – M. Diehl: An augmented Lagrangian based algorithm for distributed non-convex optimization, SIAM J. Optim., 2, 26, 1101–1127, 2016.
2. F. Borrelli: Constrained Optimal Control of Linear and Hybrid Systems. Springer Berlin Heidelberg, 2017.
3. Y. Jiang – J. Oravec – B. Houska – M. Kvasnica: Parallel MPC for Linear Systems with Input Constraints. IEEE Transactions on Automatic Control, 7, 66, 3401–3408, 2021.

Deadline for submission of Master thesis: 15. 05. 2022

Approval of assignment of Master thesis: 28. 02. 2022

Assignment of Master thesis approved by: prof. Ing. Miroslav Fikar, DrSc. – Study programme supervisor

# Honour Declaration

I declare that the submitted diploma thesis was completed on my own, in cooperation with my supervisor, with the help of professional literature and other information sources, which are cited in my thesis in the reference section. As the author of my diploma thesis, I declare that I did not break any third party copyrights.

.....
Signature

# Acknowledgment

*I want to thank my supervisor, doc. Ing. Juraj Oravec, PhD. for professional guidance, valuable suggestions and comments on creating the final work, willingness and time he devoted to me, especially for all the knowledge I have gained thanks to him. Thanks to my family and friend, who stood by me during my studies and constantly supported me.*

# Abstract

The main aim of this master thesis is to design a model predictive control (MPC) using a numerical method of distributed optimization – the Augmented Lagrangian based Alternating Direction Inexact Newton method (ALADIN). For the controller design purpose, the system of two tanks with interaction is considered as the controlled plant. The thesis is divided into two parts. In the first part, the theoretical aspects of the model predictive control are introduced and the way to reduce the computational complexity using explicit MPC is also described. Subsequently, this part offers a brief overview of the methods of distributed optimization methods with an emphasis on the ALADIN method. The second, experimental part focuses on the implementation aspects of the ALADIN algorithm in the MPC. This chapter discusses the controller design using two approaches at the distributed optimization level - implicit and explicit MPC. The analysis of the results obtained using this numerical method is compared to the optimal MPC solution. Subsequently, the computational effort of both, implicit and explicit MPCs, are analysed. Obtained results are then used in the closed-loop simulation to analyse the control performance with the use of the proposed algorithm.

**Keywords:** distributed optimization; ALADIN; model predictive control; explicit model predictive control; computational complexity

# Abstrakt

Hlavným cieľom predkladanej diplomovej práce je navrhnúť prediktívne riadenie založené na modeli (MPC) pomocou numerickej metódy distribuovanej optimalizácie - angl.: Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN). Na tento účel je ako riadený proces zvolený systém dvoch nádrží s interakciou. Práca je rozdelená na dve časti. V prvej časti sú predstavené teoretické aspekty návrhu prediktívneho riadenia a taktiež je tu popísaný spôsob zníženia výpočtovej náročnosti distribuovaného optimalizačného problému pomocou explicitného MPC. Následne táto časť ponúka stručný prehľad metód distribuovanej optimalizácie s dôrazom na metódu ALADIN. Druhá, experimentálna časť je zameraná na implementáciu algoritmu ALADIN v rámci návrhu MPC. Táto kapitola sa zaoberá riadením pomocou dvoch prístupov – implicitného a explicitného MPC. Analýza výsledkov získaných pomocou tejto numerickej metódy je porovnaná s optimálnym riešením MPC. Následne je analyzovaná výpočtová náročnosť implicitného aj explicitného MPC. Získané výsledky sú potom použité v simulácii s uzavretou slučkou, aby sa vyhodnotila kvalita riadenia pomocou navrhovaného algoritmu.

**Kľúčové slová:** distribuovaná optimalizácia; ALADIN; prediktívne riadenie; explicitné prediktívne riadenie; výpočtová náročnosť

# Contents

# List of Abbreviations

MPC Model Predictive Control

ALADIN Augmented Lagrangian based Alternating Direction Inexact Newton method

PID Proportional-integral-derivative controller

LQR Linear-quadratic regulator

SISO Single-input, single-output

MIMO Multiple-input, multiple-output

MPHC Model predictive heuristic control

DMC Dynamic matrix control

ADMM Alternating Direction Method of Multipliers

SQP Sequential quadratic programming

QP Quadratic programming

# List of Figures

# List of Tables

Chapter 1

# Introduction

The model predictive control (MPC) origins can be traced back to the 1970s. Since then, it has become one of the popular control methods for advanced process control in many industrial applications [2]. The majority of these applications are found in refineries, which is where MPC got its start. However, it should be noted that there are a significant number of applications in other areas as well. Its widespread acceptance can be explained by the ability of the MPC design to deliver high-performance control systems that can operate for extended time periods without the need for expert assistance [12].

Among other advantages belong a possibility to include the constraints on both manipulated and controlled variables and the ability to operate closer to the constraints [11]. Moreover, the variables can be finely tuned, for example, to get the output faster to the reference or get cheaper process control. Finally, all the advantages mentioned above can be used to minimise the operating costs, minimise the impact on the environment, and, on the other hand, maximise the quality of the obtained product.

Chemical companies need to be able to react immediately to changes in requirements or possible disturbances in a plant. For this reason, it is necessary to make decisions quickly enough to respond to a given situation with the appropriate action. The time required for decision-making is called the sampling time. Within this sampling time, the measurement, calculation of the optimal action, and its implementation take place. Since the MPC calculations are performed iteratively at each time step, depending on the complexity of the process model, it is often computationally and also time-consuming to solve these systems as a whole within the sampling time.

Fortunately, many of these plants are made up of several symmetrical structures periodically repeated in space or time domain. Based on these findings, the controlled system can be broken down into many simpler sub-systems, which are interconnected. This fact allows solving these partial problems in parallel with distributed optimisation

methods. As a result, the research of well-established theoretical concepts in the field of distributed optimisation is fundamental.

There are several methods that can be used to solve the problem of distributed optimisation. One of them is the Alternating Direction Method of Multipliers which is a variant of augmented Lagrangian and method of multipliers or Augmented Lagrangian based Alternating Direction Inexact Newton method (ALADIN) is one of the most perspective methods for solving the problems of the distributed optimisation. This approach can also be used for large-scale problems in a distributed way. Compared to other methods of distributed optimisation, its primary advantage is its high convergence rates, which lead to the decreased number of iterations needed to obtain the local optimum [6]. However, even when the number of iterations approaches infinity, these methods of distributed optimisation often converge to a solution that only within a limit comes to the optimal one. The way to guarantee the asymptotic stability and bounds on suboptimality of the results were discussed in [9].

The main aim of this thesis is to utilise this advantage of the ALADIN algorithm for the purposes of model predictive control. The thesis is divided into chapters. The first chapter introduces the model predictive control, its history, the theoretical aspects, and basic mathematical formulation. Also, there is briefly described the way to reduce its computational complexity using the explicit approach. Then, the basics of distributed optimisation methods are introduced and the critical methods used within the algorithms of distributed optimisation. Finally, the end of the first chapter offers a brief overview of the two methods of distributed optimisation mentioned above - the Alternating Direction Method of Multipliers and the Augmented Lagrangian based Inexact Newton method.

The second chapter discusses the practical implementation aspects of this thesis. First, the controlled process is introduced with its specifications and the model used for the model predictive control. Then the theoretical concepts presented in the first chapter are used to obtain the control of the system. For this purpose, there are two approaches implemented and analysed. The first one uses the principles of implicit MPC, and in the second one, the explicit MPC is used with the ALADIN optimisation algorithm. Results generated by these approaches are then analysed from the point of view of the open-loop simulation. The conclusion of this part is focused on the closed-loop simulation, where the results of the previous part are analysed.

Chapter 2

# Theoretical part

## 2.1 Model predictive control

Model predictive control (MPC) is a series of advanced methods of control that use a model of a plant to predict its behaviour in the future. The concept of optimal control, considering constraints, and the intuitive formulation of the control law as an optimization problem has made MPC interesting for many different branches of industry [11]. Since its introduction, it has found application not only in the process or chemical industry, but it is gradually spreading wide throughout all fields of engineering, such as power electronics, manufacturing, building climate and energy control and others.

One of the significant advantages of the MPC offer is the already mentioned ability to predict future states of the system while considering possible technological limitations of quantities, therefore it can propose appropriate input actions unlike proportional–integral–derivative controllers (PID) or even linear–quadratic regulator (LQR), which do not have such a combination of features. It is one of few control approaches that take constraints into account directly [11]. Another advantage is the possibility to control not only a single-input and single-output systems (SISO), as with PID controllers, but also allows to control of multiple-input and multiple-output systems (MIMO).

Since the MPC algorithm requires solving the optimization problem in each sampling period, its main disadvantage is the relatively long time required to calculate the optimal input actions. Therefore, it is necessary to use powerful hardware to implement MPC. That is why research is still underway to reduce the response time, which is directly proportional to the complexity of the optimization problem and the number of optimized variables.

Therefore, it is necessary to first consider well the specifics of the application and based

on them the decision to choose the appropriate approach to the effective management of the plant. For example, for a simple SISO system with fast dynamics, without technological limitations, it may be more advantageous to use a PID controller to control it. However, if we consider a complex multidimensional MIMO system with slow dynamics and significant interactions between process variables, which are additionally limited, then MPC is almost always a winner that can achieve a high level of control quality when set correctly.

### 2.1.1 History

The foundations of the MPC were laid in the late 1970s independently through [10] and [5]. Model predictive heuristic control (MPHC), proposed by [10], had already included all the characteristic features of MPC, but achieving optimal control was not guaranteed. Future control actions were determined iteratively until they satisfied the constraints. This technique was invented for MIMO systems with longer processing times [10].

Around the same time-frame, Shell Oil Company came with [5] introducing dynamic matrix control (DMC). They predicted the future behaviour of a catalytic cracking unit using a partially linear model of a system, so the controller was able to recognise the dynamics of the plant. Moreover, the model uses a receding horizon to update the coefficients of the model according to the difference between the output predicted in the previous step and the actual value of the measured output. The main advantage of DMC over MPHC was that DMC calculates the optimal input actions [11]. On the other hand, DMC was limited only to linear process models due to the matrix formulation of the control problem.

Both works provided the groundwork for the mass acceptance of MPC in the petrochemical industry. As the sampling periods were many hours even with linear systems, the focus was initially on reducing the complexity of the controller design and developing a thorough theory so that the approach could be applied in the industry. Since most chemical engineering plants were open-loop stable, the pioneering methods simply ignored uncertainties of a model and plant instabilities. From the late 1980s, the focus of researchers switched to the stability and robustness of MPC.

Using a finite horizon, the linear estimation problems could be stated as quadratic programming problems, which have proven to be computationally efficient [11]. In addition, with the new millennium and the increasing computing performance, the research switched from huge problems with large calculation periods towards problems with significantly higher requirements for reducing the computational time.

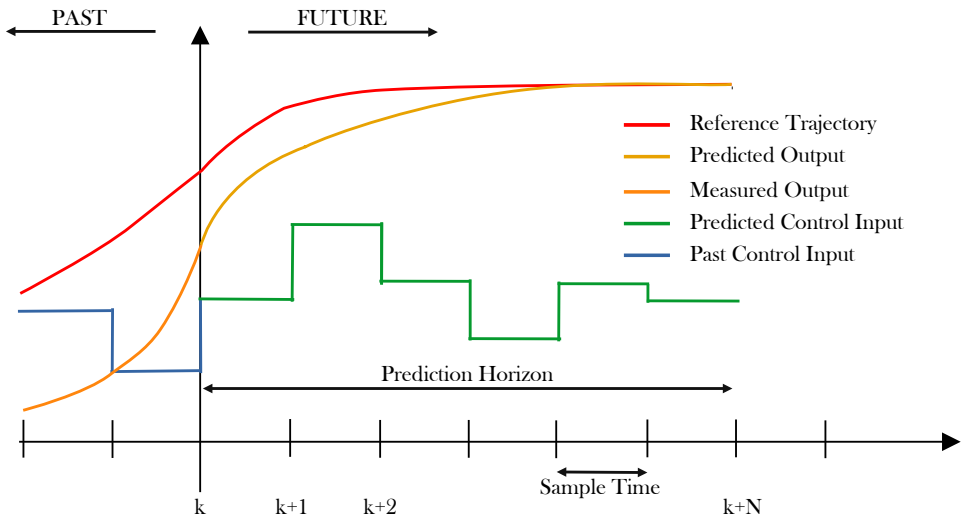



**Figure 2.1:** A basic principle of Model Predictive Control.

### 2.1.2 Formulation of MPC

MPC in basics can be described as a solution to the optimization problem with constraints in each sampling period, throughout the prediction horizon (2.1) to obtain the optimal value of the control action in the current sampling period. The resulting optimal value will be then applied to a controlled system whose mathematical model is the basis for predicting future states [12]. Measured values of state variables are then used as initial conditions for the following iteration of optimization.

The structure of MPC formulation can be divided into the cost function that in the basic form penalizes the size of state and control actions. The second part of structure are constraints that include the plant model, the initial condition and in some cases also technological limitations of control, state and output variables.

When predicting the future values of the system states, model predictive control relies on its mathematical model, thus it is important to choose the right form. There are several models and representations in control theory that more or less accurately explain the behaviour of the real plant and hence influence the accuracy of the prediction. It is obvious that by using a precise model a higher quality of control is achieved, but at the same time, the consequence can be an enormous increase in computational complexity. The standard formulation of MPC considers discrete-time linear state-space model

that quite well approximates many real plants and after its discretization, it can be used to solve the optimization problem, where it enters as constraints in the form:

$$x_{k+1} = Ax_k + Bu_k \tag{2.1a}$$
$$y_k = Cx_k + Du_k, \tag{2.1b}$$

where matrix $A \in \mathbb{R}^{n \times m}$ is the system matrix, $B$ is the input matrix, $C$ is the output matrix and $D$ is the feed forward matrix. Since only state control will be considered, the output model equation (2.1b) defining the outputs of the system can be neglected.

The basic formulation of MPC can be written as follows:

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left( \|x_k\|_Q^2 + \|u_k\|_R^2 \right) \tag{2.2a}$$
$$s.t. \quad x_{k+1} = Ax_k + Bu_k, \tag{2.2b}$$
$$x(0) = x_0, \tag{2.2c}$$
$$x_{\min} \leq x_k \leq x_{\max}, \tag{2.2d}$$
$$u_{\min} \leq u_k \leq u_{\max}, \tag{2.2e}$$

where $k = 0, \ldots, N-1$, matrix $Q$ is an weighting matrix reflecting the relative importance $x_k$ and $R$ weighting matrix penalizing relative significant changes in $u_k$. Due to the flexibility of MPC formulation, it is relatively simple to modify the standard formulation and consider the specifics of the controlled plant and thus achieve increased control quality.

### 2.1.3 Complexity Reduction

The MPC strategy described in the previous section require performing on-line optimization to solve the optimization problem (2.1b) based on the current state. As a result, MPC has long been regarded as a technique for processes with slow dynamic. Even while advances in microcontroller and computer technology are redefining the definition of "slow processes", the inability to solve on-line prevents MPC from being used in many situations, even in the most basic cases. This leads to the formation of explicit MPC.

Explicit MPC approaches combine an off-line solved optimization problem with online control implementation. Optimization problem is solved off-line for every state variable within a set defined by the constraints using the technique of multiparametric programming and stored in a form of the look-up table [2]. The solution to this problem is the state space divided into a specific number of polyhedral regions, called polytopic

partition [14]. Over these regions, the control law is defined in the form of piecewise affine function. The subsequent real-time implementation consists of finding the region in which the current state of the system is located and quantifying the appropriate affine function in the form

$$u(x) = \begin{cases} F_1 x(t) + g_1 & \text{if } x(t) \in R_1, \\ \quad \vdots & \\ F_m x(t) + g_m & \text{else if } x(t) \in R_m, \end{cases} \tag{2.3}$$

where $F_i$ and $g_i$ represent slope and the affine part of the control law, $R_i$ is the critical region and $m$ indicates the number regions. The result is an optimal input applied to the controlled system.

Obviously, the implementation of an explicit MPC is significantly easier than a computationally demanding solution to the implicit MPC problem. However, a simple state search within the partition can be time consuming in the case of higher order systems and partition with thousands of polyhedral region and can also have large memory requirements for storing the solution in the form of look-up table [14]. The solution to this issue can be approximation of the polytopic partition by a neural network.

## 2.2 Distributed Optimization

Optimization is a mathematical field that finds the best feasible solution at which the optimal performance of the problem is achieved. Automatic control systems, estimation and signal processing, communications and networks, data analysis, statistics, and finance are just a few of the fields where optimization theory is necessary. Despite their differences, all these areas face the same problems – optimized datasets are frequently enormous and due to the large scale, the data are often stored distributively [15]. Finding analytic solutions of these optimization problems using centralized solution methods may be therefore difficult or even impossible. As a result, it is necessary to use distributed optimization approaches, in which number of smaller sub-systems cooperate to identify a solution of the original problem.

Distributed optimization is an optimization method that is often used in large-scale networked systems. Even if the central controller is not part of the system, this technique allows the system to solve a global problem cooperatively [1]. When compared to centralized methods, distributed optimization offers several significant befits. In distributed algorithms, network nodes or users only share information with those parts

of the system that need it. This improves cyber security and decreases communication costs [1]. Furthermore, distributed approaches may tackle also issues of any scale, including the large ones. There is also the potential that these methods can speed up the solution process. Large-scale and data-intensive problems are solved using distributed optimization algorithms in many applications such as communications, energy grids, smart grids, and statistical learning [1].

In many distributed optimization algorithms, there is decomposition approach used. Decomposition is a method for breaking down a large global problem into smaller sub-problems that are solved independently [3].

On the other hand, standard optimization techniques are not designed for parallel computing. For that reason, it is necessary to use methods applying decentralization of the optimization [15]. Such methods include the *Alternating Direction Method of Multipliers* (ADMM) and the *Augmented Lagrangian Alternating Direction Inexact Newton* (ALADIN) method, which will be discussed in detail later in this work.

### 2.2.1 The Dual Problem

Consider the following convex optimization problem with equality constraints:

$$\begin{aligned} \min_{z} \quad & f(u) \\ \text{s.t.} \quad & Eu = b. \end{aligned} \tag{2.4}$$

This is a primal problem for a primal function $f : \mathbb{R}^n \to \mathbb{R}$ with a primal variable $u \in \mathbb{R}^n$, where $E \in \mathbb{R}^{m \times n}$ [4]. The dual problem for (2.4) is $\max_v \quad g(v)$, where $g(v)$ is the dual function with $v$ called dual variable or Lagrange multiplier

$$g(v) = \inf_u L(u, v),$$

and $L(u, v)$ is the Lagrangian defined as

$$L(u, v) = f(u) + v^\top (Eu - b). \tag{2.5}$$

The optimal values of the primal and dual problems are the same if strong duality holds. From the dual optimum $v^\star$, the primal optimal point $u^\star$ can be recovered as follows

$$u^\star = \arg\min_u L(u, v^\star).$$

One of the approach solving this problem is called *Dual Ascent* method and it is based on gradient ascent. The algorithm of the Dual Ascent method strictly converges and

seeks out for the gradient $\nabla g(v)$, if $g$ is a differentiable function [4]. The gradient can be then calculated as $\nabla g(v) = Eu^\star + b$, where $u^\star$ is defined using the relation above. The algorithm of the dual ascent method can be used in some cases even if $g$ is not differentiable, the convergence is not monotone and the algorithm seeks out the negative of a sub-gradient of $-g$[4]. The dual ascent method is then performed by iterating the updates:

$$u_{k+1} := \arg\min_u(L(u, v_k)), \tag{2.6}$$

$$v_{k+1} := v_k + \alpha_k(Eu_{k+1} - b), \tag{2.7}$$

where $\alpha_k$ symbolizes the step size for iteration $k$, equation (2.6) represents the minimization of the function $f(u)$ with respect to $u$, and in equation (2.7), the dual variable $v$ is updated for the next step of iterations.

### 2.2.2 Dual Decomposition

In the case of larger systems, it may be appropriate to parallelize the *Dual Ascent* method to achieve better performance. This section describes how to accomplish this using the dual decomposition. Assume that the objective is separable, and that $f(u) = f_1(u_1) + \cdots + f_n(u_N)$, where $u = (u_1, \ldots, u_N)^\top$ and the variables $u_i \in \mathbb{R}_i^n$ are sub-vectors of $u$. Then the same may be applied to the Lagrangian described in (2.5) as follows

$$L(u, v) = L_1(u_1, v) + \cdots + L_N(u_N, v), \tag{2.8}$$

where $L_i = f(u_i) + v^\top E_i u_i$ [4]. As a result, the dual $u$-minimization step in dual ascent method is separated into $N$ independent minimization problems that can be solved in parallel:

$$u_{i,k+1} := \arg\min_{u_i} \left(L_i\left(u_i, v_k\right)\right).$$

This gives a suitable parallelization strategy – distribute $v(k)$ and update $u_i$ in parallel for each $i = 1, \ldots, N$. This is known as Dual Decomposition and it first appeared in optimization in works from 1960s [1]. Thus, the method computes the above $u$-minimization step in parallel, and the coordinates to update the dual variable:

$$v_{k+1} := v_k + \alpha_k \left(\sum_{i=1}^{N} E_i u_{i;k+1} - b\right). \tag{2.9}$$

Although this method is very convenient at first glance, but it involves some major assumptions (a sufficiently smooth and separable $f(u)$), also it may be slower in some cases [3].

### 2.2.3 Method of Multipliers

In order to make the dual ascent method more robust and speed the convergence, it is convenient to replace the Lagrangian with an Augmented Lagrangian:

$$L_\rho(u, v) = f(u) + v^\top (Eu - b) + \frac{\rho}{2} \|Eu - b\|_2^2, \tag{2.10}$$

where $\rho > 0$, and $\frac{\rho}{2} \|Eu - b\|_2^2$ is another penalty term added to penalize straying from the constraints [13]. The algorithm now proceeds in the following steps until convergence is achieved:

$$u_{k+1} := \arg\min_u (L_\rho(u, v_k)), \tag{2.11}$$

$$v_{k+1} := v_k + \rho(Eu_{k+1} - b), \tag{2.12}$$

where in (2.11) the Lagrangian is minimized with respect to $u$ and in (2.12) the dual variable $v$ is updated for next iteration. The dual update step length $\rho$ is set to the same value as the penalty coefficient in (2.10).

The Augmented Lagrangian is differentiable under mild conditions for the primal problem [13]. For the problem defined in (2.4), the optimality conditions for a differentiable $f$ are:

| | |
|---|---|
| Primal Feasibility: | $Eu^\star - b = 0$ |
| Dual Feasibility: | $\nabla f(u^\star) + E^\top v^\star = 0.$ |

Since $u_{k+1}$ minimizes $L_\rho$ in each iteration, leading to

$$\begin{aligned} \nabla_u L_\rho \left(u_{k+1}, v_k\right) &= \nabla_u f \left(u_{k+1}\right) + E^\top \left(v_k + \rho \left(Eu_{k+1} - b\right)\right) \\ &= \nabla_u f \left(u_{k+1}\right) + E^\top v_{k+1} = 0. \end{aligned}$$

From the above it is obvious that the iterate $(u_{k+1}, v_{k+1})$ is dual feasible when $\rho$ is used as the step size in the dual update. The primal residual $(Eu_{k+1} - b)$ converges to zero as the method of multipliers continues, achieving optimality. [13]

On the one hand, the method of multipliers has far better convergence features that the dual ascent. On the other hand, the augmented Lagrangian is not separable when $f$ is separable so the step (2.11) cannot performed in parallel for each $u_i$. As a result, the basic method of multiplier cannot be used in combination with the decomposition.

### 2.2.4 Alternating Direction Method of Multipliers

For convex optimization problems, *Alternating Direction Method of Multipliers* (ADMM) has shown to be an efficient distributed approach combining the advantages of the dual ascent method and the method of multipliers.

Consider the optimization problem written in form:

$$\begin{aligned} \min_{u} \quad & \sum_{i=1}^{N} f_i(u_i) \\ \text{s.t.} \quad & \sum_{i=1}^{N} E_i u_i = b \\ & h_i(u_i) \leq 0, \end{aligned} \tag{2.13}$$

where $i = 1, \ldots, N$. The structure optimization problem can be written in the equivalent form

$$\begin{aligned} \min_{u,z} \quad & \sum_{i=1}^{N} g_i(z_i) + I_0 \left( \sum_{i=1}^{N} E_i u_i - b \right) \\ s.t. \quad & E_i(z_i - u_i) = 0. \end{aligned} \tag{2.14}$$

The term $g_i(z_i)$ represents the extended objective functions given by

$$\forall i \in \{1, \ldots, N\}, \qquad g_i(z_i) = \begin{cases} f_i(z_i) & \text{if } h_i(z_i) \leq 0, \\ \infty & \text{otherwise,} \end{cases}$$

$I_0 : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ denotes the indicator function

$$I_0(r) = \begin{cases} 0 & \text{if } r = 0 \\ \infty & \text{otherwise,} \end{cases}$$

and the matrices $E_i : \mathbb{R}^{m \times n}$ and the vectors $b \in \mathbb{R}^m$ contain the coefficients of the given constraints. The equivalence of forms (2.13) and (2.14) can be explained by the fact that the variables $x_i$ enter the coupled constraints only through the terms $E_i u_i$. That is why it is sufficient to apply the constraints $E_i(z_i - u_i) = 0$ instead of requiring $u_i = z_i$ [6]. The main idea of ADMM is to construct an augmented Lagrangian function of the form

$$\begin{aligned} L_\rho(u, z, \lambda) = & I_0 \left( \sum_{i=1}^{N} E_i u_i - b \right) \\ & + \sum_{i=1}^{N} \left( g_i(z_i) + \lambda_i^\top E_i(z_i - e_i) + \frac{\rho}{2} \left|| E_i(z_i - z_i) \right||^2 \right), \end{aligned} \tag{2.15}$$

where $\rho$ is the penalty parameter. Starting with an initial guess $u$ for the primal optimization variable and an initial guess $\lambda$ for the dual vector that is associated with the equality constraints [6].

Optimization problem (2.16) for variable $z$ in the first step of Algorithm 1 is decoupled and can be solved in parallel. The second optimization problem (2.17) for the variable $u^+$ in the fourth step can be solved explicitly since this corresponds to solving a quadratic program with linear equality constraints.

**Algorithm 1** Alternating Direction Method of Multipliers [6].

**Input:** Initial guesses $ui \in \mathbb{R}^n$ and $\lambda_i \in \mathbb{R}^m$, a penalty parameter $\rho > 0$, and a tolerance $\epsilon > 0$.

**Repeat:**

1: Solve for all $i \in \{1, \ldots, N\}$ the decoupled nonlinear optimization problems (NLPs)

$$\begin{aligned} \min_{z_i} \quad & f_i(z_i) + \lambda_i^\top E_i z_i + \frac{\rho}{2} ||E_i(z_i - u_i)||_2^2 \\ s.t. \quad & h_i(z_i) \leq 0. \end{aligned} \tag{2.16}$$

2: If $\left|\left|\sum_{i=1}^N E_i z_i - b\right|\right|_1 \leq \epsilon$, terminate and return $u^\star = z$ as a numerical solution.

3: Implement the dual gradient steps $\lambda_i^+ = \lambda_i + \rho E_i(z_i - u_i)$.

4: Solve the coupled equality constrained quadratic programming problem

$$\begin{aligned} \min_{u^+} \quad & \sum_{i=1}^N \left( \frac{\rho}{2} \left|\left|E_i(z_i - u_i^+)\right|\right|_2^2 - \left(\lambda_i^+\right)^\top E_i u_i^+ \right) \\ s.t. \quad & \sum_{i=1}^N E_i u_i^+ = b. \end{aligned} \tag{2.17}$$

5: Update the iterates $u \leftarrow u^+$ and $\lambda \leftarrow \lambda^+$ and continue with Step 1.

The convergence of Algorithm 1 may be shown with minimal assumptions if the functions $f_i$ and $h_i$ are convex. This conclusion holds regardless of how far the starting $(u, \lambda)$ is from the optimal solution or how the penalty value $\rho > 0$ is selected [6]. Another benefit of Algorithm 1 is that it is possible to solve it in parallel.

### 2.2.5 Sequential Quadratic Programming

One of the advanced and widely used approaches for handling nonlinear constrained optimization problem is sequential quadratic programming (SQP). The basic idea behind the SQP is to use a quasi-Newton updating method to approximate the computationally extensive full Hessian matrix. As a result, at each iteration, this creates a quadratic programming sub-problem and its solution may be used to define the

search direction and the next trial solution. Consider a general nonlinear optimization problem

$$\begin{aligned} \min_{u} \quad & f(u) \\ \text{s.t.} \quad & e(u) = 0, \\ & h(u) \leq 0. \end{aligned} \tag{2.18}$$

Let $u_k$ denote the current iteration and $u_{k+1} = u_k + p_k$ denote the next iteration, with $p_k$ denoting the search direction. The search direction in unconstrained optimization algorithms can be found either by going the steepest descent of the cost function ($u_{k+1} - u_k = p_k = -\nabla_u f(u_k)$), the Newton direction of a quasi Newton direction if inexact Hessians are present. SQP algorithms, on the other hand, determine the search direction by solving a local quadratic program.

First, define the Lagrangian $L(u, \lambda, \mu) = f(u) - \nabla^\top e(u) - \mu^\top h(u)$. Let $t_L(p_k)$ be a quadratic model of $L(u_{k+1}) = L(u_k + p_k)$ derived from the second order Taylor expansion

$$t_L(p_k) = L(u_k) + \nabla_u L(u_k)^\top p_k + \frac{1}{2} p_k^\top \nabla_{uu}^2 L(u_k) p_k,$$

and the first order Taylor expansion will provide the linearized constraints

$$\begin{aligned} e_i(u_k) + \nabla_u e_i(u_k)^\top p_k = 0, \\ h_j(u_k) + \nabla_u h_j(u_k)^\top p_k \leq 0. \end{aligned} \tag{2.19}$$

SQP approaches iteratively solve the quadratic problem

$$\begin{aligned} \min_{p_k} \quad & t_L(p_k) \\ \text{s.t.} \quad & e_i(u_k) + \nabla_u e_i(u_k)^\top p_k = 0, \\ & h_j(u_k) + \nabla_u h_j(u_k)^\top p_k \leq 0, \end{aligned} \tag{2.20}$$

until convergence is reached. In fact, this QP may be understood as Newton's method applied to the Karrush-Kuhn-Tucker (KKT) optimality conditions.

## 2.3 Augmented Langrangian based Alternating Direction Inexact Newton Method

*Augmented Langrangian based Alternating Direction Inexact Newton Method* (ALADIN) is a new approach of distributed optimization introduced in 2016 through an [7]. One of its advantages is that ALADIN is locally equivalent to a Newton-type approach,

which means that using correct setups, superlinear or quadratic convergence rates may be obtained [7]. If no non-linear constraints are present and the augmented Lagrangian parameters go to infinity, it can be also proved that ALADIN leads to the sequential quadratic programming SQP techniques 2.2.5.

Another advantage is that ALADIN can be used also to find local optimum solutions to non-convex optimization problems. Although ADMM approaches have similar findings under specific assumptions on the augmented Lagrangian parameter, ALADIN has the benefit that its local convergence qualities are unaffected by the choice of this parameter [7]. Furthermore, for large-scale optimization problems, it was proved that ALADIN needs significantly less iterations to reach the same accuracy as ADMM.

For the purposes of describing the algorithm, let us consider the optimization (2.13) where function $f_i : \mathbb{R}^n \to \mathbb{R}$ and $h_i : \mathbb{R}^n \to \mathbb{R}^{n_h}$ are assumed to be twice continuously differentiable for all $i = 1, \ldots, N$. Problem (2.13) is also considered to be feasible, with all local minimizers being regular Karush-Kuhn-Tucker points [6].

The main idea of the Algorithm 2 is based on the assumption that tools for solving the coupled and potentially distributed equality constrained quadratic programming problems in (2.22) as well as a centralized non-linear programming solver for solving problems in (2.21) are already available [6]. In the algorithm 2 $\kappa_i \in \mathbb{R}_+^{n_h}$ represent the dual variable of the inequality constraints $h_i(u_i)$ and the multipliers of the coupling layer equality constraints are denoted as $\lambda$. When the original problem (2.13) is feasible, also the decoupling optimization problems (2.21) are feasible. The quadratic programming sub-problems (2.22) is always feasible, since the point

$$(\Delta, s) = \left(0, \sum_{i=1}^{N} A_i v_i - b\right)$$

is an universal solution of the problem (2.22).

The size of the parameters $\lambda$ and $\rho$ used in (2.21) directly influence the convergence rate of decoupled optimization problem, as if they are large enough they push the solution to its optimum in the case when the optimized variables $u_i$ and the help variables $v_i$ are not equal. In addition, the scaling matrices $\Omega_i$ can help the convergence too and there is also a possibility to choose different weight on various optimized variables. From the second step it is obvious that the optimization iterations of the algorithm are terminated in the case when the difference between the optimized and the help variables is sufficiently small - smaller that the set tolerance and equality constraints are fulfilled.

**Algorithm 2** Augmented Lagrangian based Alternating Direction Inexact Newton Method [6].

**Input:** Initial guesses $u_i \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^m$ and a tolerance $\epsilon > 0$.

**Repeat:**

1: Choose a sufficiently large penalty parameter $\rho \geq 0$ and positive semi-definite scaling matrices $\Omega_i$ and solve for all $i \in \{1, \ldots, N\}$ the decoupled problems

$$\begin{aligned} \min_{z_i} \quad & f_i(z_i) + \lambda^\top E_i z_i + \frac{\rho}{2} ||z_i - u_i||^2_{\Sigma_i} \\ s.t. \quad & h_i(z_i) \leq 0 \quad | \quad \kappa_i \end{aligned} \tag{2.21}$$

to either local or global optimality.

2: If $\left|\left|\sum_{i=1}^N E_i z_i - b\right|\right|_1 \leq \epsilon$ and $\rho \left\|\Sigma_i(z_i - u_i)\right\|_1 \leq \epsilon$, terminate with $u^\star = z$ as a numerical solution.

3: Choose constraint Jacobian approximations $J_i \approx J_i^\star$ of the matrices $J_i^\star$ defined by

$$J^\star_{i,j} = \begin{cases} \frac{\partial}{\partial u}(h_i(u))_j\big|_{u=z_i} & \text{if } (h_i(z_i))_j = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for} \quad i \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, n_h\}.$$

Compute the modified gradient $g_i = \nabla f_i(z_i) + (C_i^\star - C_i)^\top \kappa_i$ and choose symmetric Hessian approximations $H_i \approx \nabla^2 \{f_i(z_i) + \kappa_i^\top h_i(z_i)\}$.

4: Choose a sufficiently large penalty parameter $\mu > 0$ and solve the coupled QP

$$\begin{aligned} \min_{\Delta z, s} \quad & \sum_{i=1}^N \left( \frac{1}{2} \Delta z_i^\top H_i \Delta z_i + g_i^\top \Delta z_i \right) + \lambda^\top s + \frac{\mu}{2} \|s\|_2^2 \\ s.t. \quad & \sum_{i=1}^N A_i(z_i + \Delta z_i) = b + s \quad \Big| \lambda_{QP} \\ & C_i \Delta z_i = 0, i \in \{1, \ldots, N\}. \end{aligned} \tag{2.22}$$

5: Set $\alpha_1 = \alpha_2 = \alpha_3 = 1$ and define

$$\begin{aligned} u^+ &= u + \alpha_1(z - u) + \alpha_2 \Delta v, \\ \lambda^+ &= \lambda + \alpha_3(\lambda_{QP} - \lambda). \end{aligned}$$

6: Update the iterates $u \leftarrow u^+$ and $\lambda \leftarrow \lambda^+$ and continue with Step 1.

In the fourth step of the algorithm quadratic optimization problem with linear constraints (2.22) can be transformed to the analytical solution and that is why the convergence of this problem can be superlinear. Within this step the sizes of the changes of the optimized variables, that have to be made so that the every single sub-problem of the decoupled layer will follow each other, are minimized. Similarly to other gradient based method, in this algorithm too, the Jacobian approximation in the third step helps to decide which direction will helps to convergence of the coupled quadratic problem. In the last two steps of the algorithm, the changes of optimized variables are made and these modified values are used in the next iteration of the algorithm in the decoupled layer.

Chapter 3

# Experimental Part

## 3.1 Model of Controlled System

The experimental part of the thesis involved the application of a control algorithm to a controlled system consisting of two liquid tanks with straight vertical walls that each hold a liquid and are connected in series. Figure 3.1 depicts a schematic representation of the controlled system. The liquid enters a tank on its top. There is no considered constraint on the flow rate of the stream $q_0(t)$. With the flow rate caused by gravity, an out-flowing stream exits the $i$th tank at its bottom and flows away. The flow rates are determined by Torricelli's law, which is $q_i(t) = k_{ii}h_i(t)$, where $k_{ii}$ is the valve constant and $h_i(t)$ is the height of liquid in the tank. Two streams, $q_0(t)$ and $q_1(t)$, flow into the second tank. The area of the $i$th tank's cross-section is indicated by $F_i$. The density $\rho$ is assumed to be constant everywhere in order to create the model, and the only level that can be measured is the level in the second tank.

As we mentioned in Section 2.1.2, an appropriate mathematical model is necessary to predict behaviour of the controlled system. For this reason, the state-space model of the two-tanks system was derived in the form of (2.1):

$$
\begin{aligned}
A_{\mathrm{c}} &= \begin{bmatrix} -0.63 & 0.00 \\ 0.25 & -0.21 \end{bmatrix}, \\
B_{\mathrm{c}} &= \begin{bmatrix} 1.25 \\ 0.50 \end{bmatrix}, \\
C_{\mathrm{c}} &= \begin{bmatrix} 0.00 & 1.00 \end{bmatrix}, \\
D_{\mathrm{c}} &= \begin{bmatrix} 0.00 \end{bmatrix}.
\end{aligned}
$$

After discretization considering the sampling time $T_s = 1.50$ seconds, we get the

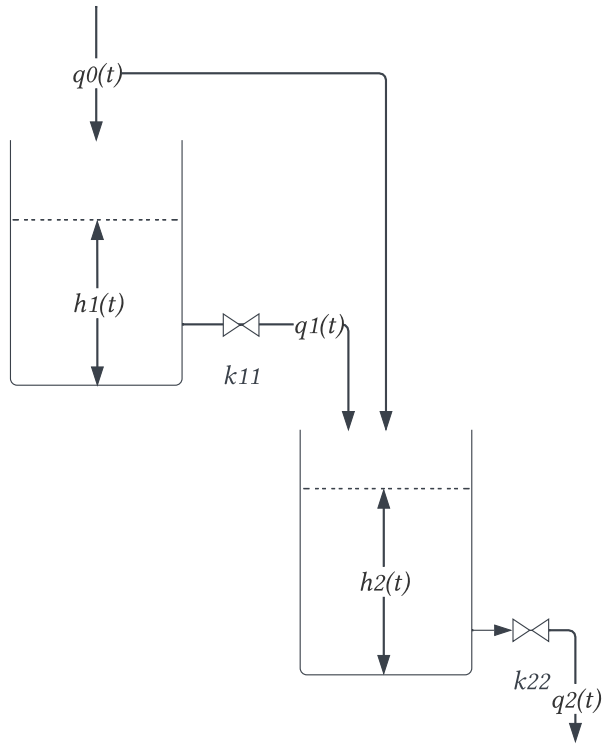



**Figure 3.1:** Schematic diagram of the two-tank system.

discrete-time state-space model

$$
\begin{aligned}
A &= \begin{bmatrix} 0.39 & 0.00 \\ 0.20 & 0.73 \end{bmatrix}, \\
B &= \begin{bmatrix} 1.22 \\ 0.88 \end{bmatrix}, \\
C &= \begin{bmatrix} 0.00 & 1.00 \end{bmatrix}, \\
D &= \begin{bmatrix} 0.00 \end{bmatrix},
\end{aligned}
$$

which will be used as the (equality) constraints in the optimization problem of MPC design. One of the most significant advantages of MPC is that it incorporates physical limitations into the optimization process. We use two forms of this type of restriction to compute the input actions. The first is a limitation on the amount of flow that can enter the tops of both liquid tanks. This flow-rate is represented by input deviation variable $u$. The second is a constraint on the level of liquid that can be stored in the tanks, which are represented by the state deviation variables $x_1$ and $x_2$:

$$
\begin{aligned}
u_{\min} &= -0.1 \text{ m}^3/\text{s}, \\
u_{\max} &= 0.1 \text{ m}^3/\text{s}, \\
x_{\min} &= \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \text{m}, \\
x_{\max} &= \begin{bmatrix} 0.5 \\ 2.5 \end{bmatrix} \text{m}.
\end{aligned}
$$

The aim of the process control is to get liquid levels in both tanks to their steady-state values from the non-zero initial state

$$
x_0 = \begin{bmatrix} -0.3 \\ 2 \end{bmatrix} \text{m}.
$$

The deviation variables $u$, $x_1$, and $x_2$ were determined as the differences between the current value of the input flow-rate and its steady-state value, respectively the current values of the liquid levels in both tanks and their steady-state values.The steady-state values that correspond to these are $q_0^s = 1 \text{ m}^3/\text{s}$, $h_1^s = 1 \text{ m}$, and $h_2^s = 2.37 \text{ m}$.

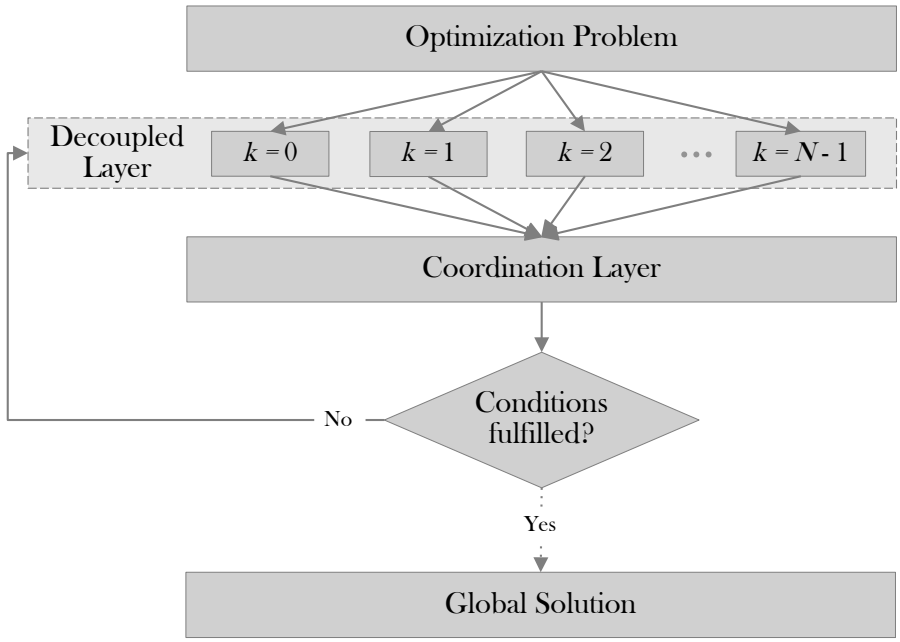



**Figure 3.2:** Schematic algorithm of the implicit MPC.

## 3.2 Implementation of ALADIN Algorithm for MPC

To obtain a control inputs that will bring both state variables to their steady state values, as described in the previous Section, it is necessary to solve the optimization problem (2.2). Solution of this problem might be computational extensive in case of larger controlled systems with high number of optimization variables and constraints that are taken into consideration. The application of traditional optimization techniques, both analytical and numerical, is frequently insufficient, particularly when it comes to the control of systems with fast dynamics for which the sampling time is limited. For this reason we decided to use one of the method of distributed optimization - ALADIN 2.3. In the following Sections, there are discussed ways of its implementation for the design of model predictive control of the two-tanks system.

### 3.2.1 Implementation of Implicit MPC

Analyzing the controlled system, we may observe that it is composed of a number of symmetrical structures that are periodically repeated in either space or time. Based

on these findings, we can break down the investigated system into a number of simpler sub-systems that are interconnected with one another. This operation is what allows us to apply the ALADIN algorithm to the system control optimization problem.

In the Figure 3.2 is the schematic structure of the algorithm used for the open-loop process control.In the first step, there is solved the decoupled layer. Powerful benefit of this part of ALADIN algorithm is in the possibility to solve these sub-problems in parallel, so that the computational time might be significantly reduced. It consists of several sub-problems representing individual steps of the prediction horizon $N$. From this it is clear that in our case complexity of the decoupled layer depends on the length of prediction horizon, so it is necessary to choose it wisely. In this thesis, the prediction horizon $N$ was set to 20 steps. Based on the equation (2.21), the objective function of these sub-problems is implemented in three forms depending on the current step of the prediction horizon – the first step, the last step, and the inner step of the prediction horizon.

As these sub-problems are designed to minimize the cost function (2.2), it is necessary to choose also the weight matrix for state variables $Q$ and for the weighting matrix for control input $R$. Within this work these weighting matrices were set as follows:

$$Q = \begin{bmatrix} 15.00 & 0.00 \\ 0.00 & 2.68 \end{bmatrix},$$
$$R = \begin{bmatrix} 10.00 \end{bmatrix}.$$

Besides these matrices, there are also few extra tuning parameters that are needed to be set to solve the optimization problem of the decoupled layer. In this project Lagrange multipliers $\lambda$ are initialized with zero matrices, the penalty parameter $\rho = 5$ and the scaling matrix $\Omega$ is set to the solution of the discrete-time Riccati equation for the studied system. These parameters were introduced in the equation (2.21).

These optimization problems are formulated using the YALMIP toolbox in MATLAB and then they are evaluated using the GUROBI solver, considering the technological constraints introduced in the previous Section. Result of the decoupled layer is prediction of the control input and state variables for every step of the prediction horizon.

Since the sub-systems are solved independently on each other, it is necessary to introduce the coordination layer that pushes the individual solutions into a continuous trajectory. At the coordination layer, there is only a simple quadratic optimization

problem solved:

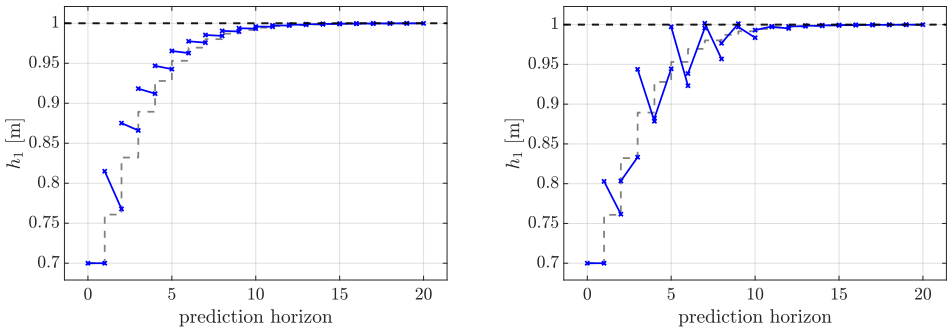
$$\begin{aligned} \min_{\Delta u, \Delta z} \quad & \sum_{k=0}^{N-1} \left( \|\Delta z_k\|_Q^2 + \|\Delta u_k\|_R^2 \right) \\ s.t. \quad & x_{k+1} + \Delta z_{k+1} = A(x_k + \Delta z_k) + B(u_k + \Delta u_k), \\ & \Delta z_0 = 0, \end{aligned}$$

where $k = 0, \ldots, N-1$, $\Delta u$ represents the changes of inputs in successive steps of the prediction horizon, respectively $\Delta z$ represents the size of the gaps between the state solutions of the distributed problems. As a result, we get the information about the changes that are needed to take and the optimal values of Lagrange multipliers, that are used during the following iteration to solve the decoupled layer.

These iterations continue until the stopping criterion is reached. In this project we use two types of terminal criteria – the first one determines the maximum number of iterations that we are willing to take within optimization process, and the second one is stopping criterion for the minimal size of the change that should be applied in the following iteration:

$$\rho \|\Delta z\|_1 > \epsilon,$$

where $\epsilon$ is the set tolerance. Within this project, the maximum number of iterations is 15 and the tolerance is set to 0.01 m. Using appropriately chosen tuning parameters, we should get a result approaching the global optimum after several iterations.



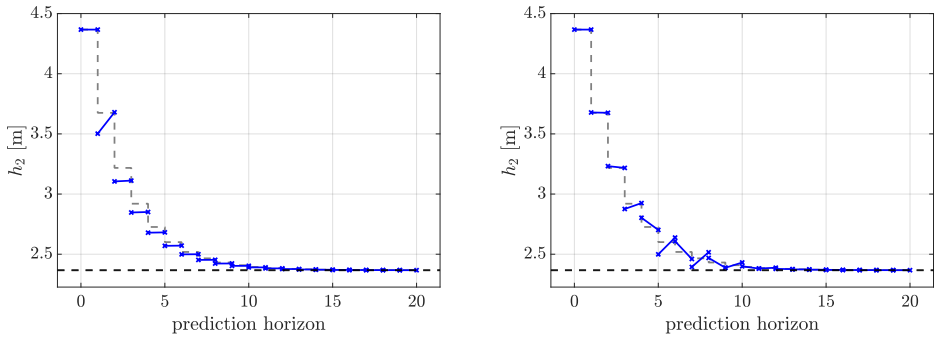

**(a)** Trajectory after the 1st iteration

**(b)** Trajectory after the 15th iteration

**Figure 3.3:** Trajectory of the liquid level in the first tank over the prediction horizon - results of the ALADIN algorithm (blue solid), non-distributed MPC (grey dashed), and the steady-state value (black dashed).

Figures 3.3–3.5 show trajectories of the state variables and control input, obtained by the algorithm described above. In the Figure 3.3 there are displayed resulting trajectories of the level in the first tank. For the comparison there is also visible the optimal solution of MPC obtained using same weight matrices as in the ALADIN algorithm. This solution was calculated using YALMIP toolbox and GUROBI solver.

Grey dashed trajectory of MPC solution is also included in the Figures 3.3–3.5 to verify the correctness of the obtained results from numerical ALADIN method. As we can see, both trajectories that we got after the first and also the 15th iteration of ALADIN algorithm differ quite distinctly from the reference optimal solution. This may be the consequence of the fact, that when deriving the model of the controlled system, we considered that only the level in the second tank can be measured, and thus the first-state behaviour may not be correctly captured in the numerical method.



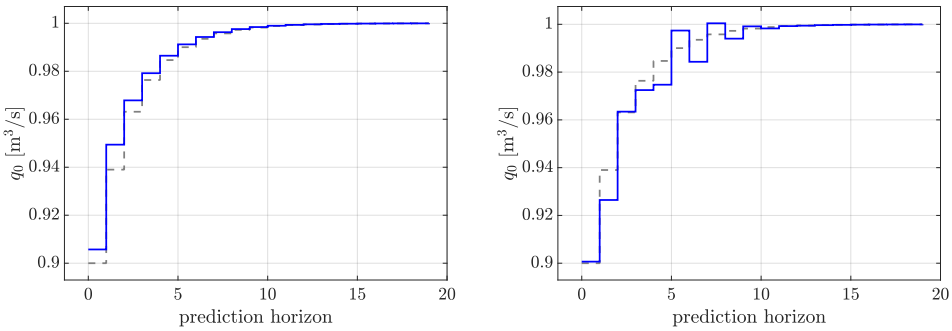

**(a)** Trajectory after the 1st iteration

**(b)** Trajectory after the 15th iteration

**Figure 3.4:** Trajectory of the liquid level in the second tank over the prediction horizon - results of the ALADIN algorithm (blue solid), non-distributed MPC (grey dashed), and the steady-state value (black dashed).

In the Figure 3.4, there are trajectories for the liquid level in the second tank. Similarly to the previous graphs, the reference optimal solution is also shown here to check the correctness of the obtained solution. As we can see both trajectories of the ALADIN solution after one iteration and after fifteen iterations are approaching the reference MPC solution. There is also displayed the reference line of the steady-state value, that we want to reach applying the obtained control input to the system. Since the trajectory obtained after the first iteration does not significantly differ from the reference trajectory, there is potential to shorten the time needed for the calculation of the ALADIN method by considering the first solution as enough satisfactory.

Trajectory of the control input – the flow-rate $q_0$ is shown in the Figure 3.5. Comparing these trajectories, we can see that the first input action in the trajectory obtained after the $15^{\text{th}}$ iteration of ALADIN algorithm is almost identical with the result of the reference MPC solution in the first step of the prediction horizon. Given that this solution of the first step of the prediction horizon will be used in the control loop, we can assume that the control quality will be higher in the case of solution obtained after the the $15^{\text{th}}$ iteration.




**(a)** Trajectory after the $1^{\text{st}}$ iteration **(b)** Trajectory after the $15^{\text{th}}$ iteration

**Figure 3.5:** Trajectory of the control input over the prediction horizon - flow-rate obtained using the ALADIN algorithm (blue solid), non-distributed MPC (grey dashed).

On the other hand, in the Figure 3.5a containing the result of the first iteration of ALADIN algorithm, we can see that the obtained trajectory is also sufficiently approaching the optimal trajectory of the MPC. This can confirm us in our assumption that in the future research on this method could be developed in the way of accepting the result from the first iteration, even though the control quality would be lower.

Considering all the shown figures, we can assume that our algorithm for numerical solution of MPC is correct. However, the computational time of the implicit solution is too long to be used for control of the systems with fast dynamics, see Table 3.1.

### 3.2.2 Implementation of Explicit MPC

One of the major advantages of the explicit model predictive control design is a possibility to pre-optimize the control problem within the offline phase. This leads to significant shortening of time needed for the calculations as the online phase is reduced to the solution of a point location problem (see Section 2.1.3). This is one of

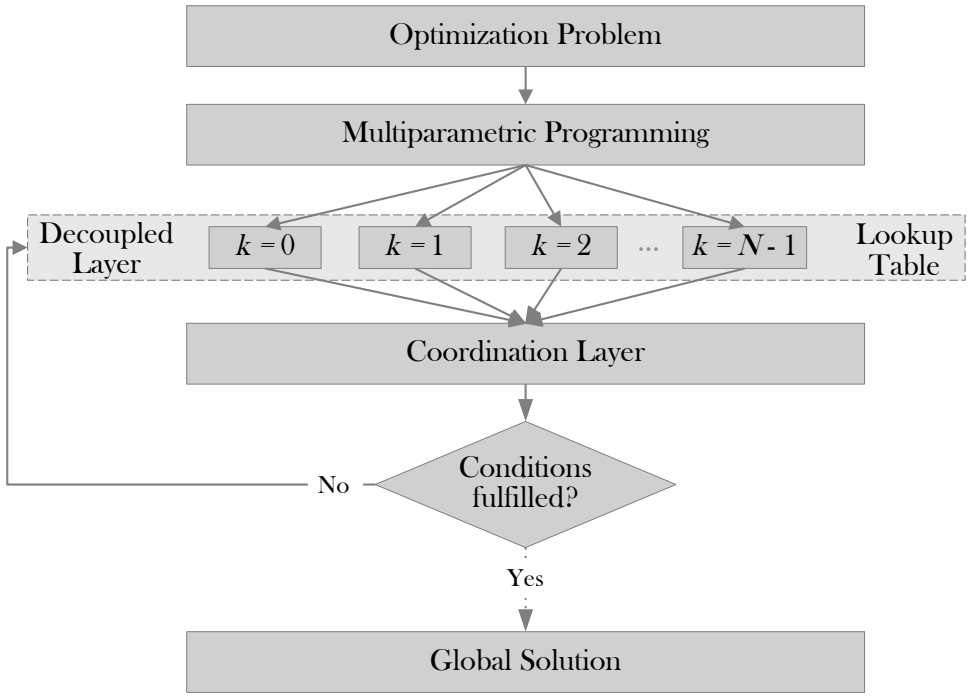



**Figure 3.6:** Schematic algorithm of the explicit MPC.

the options how to decrease the evaluation time of the ALADIN algorithm introduced in the Section 3.2.1. However, in case of more complex optimization problem, the memory requirements for storing of all the solutions in the look-up table are increasing. In order to implement this method to microcontrollers used in practice, it might be more suitable to apply this approach rather to smaller problems.

Since the decoupled layer of the ALADIN method optimizes multiple smaller sub-problems of the process control repeatedly, there is a space to utilize the advantage of explicit MPC. We are also confirmed about the correctness of this decision by the fact that the decoupled layer solution currently makes up more than 95% of the time required for the complete calculation of the ALADIN algorithm.

The schematic algorithm of implementation of the explicit MPC into the ALADIN method described in the Section 3.2.1 is shown in the Figure 3.6. As part of this approach we included the offline phase of explicit MPC before solving the decoupled layer. Within this step there are three controllers constructed for the decoupled quadratic programming. As we mentioned before, the optimization problem of these sub-problems is implemented in three forms depending on the current step of the prediction horizon – the first step, the last step or any other the inner step of the prediction horizon. This means that independently on the length of the prediction horizon, there are always only three explicit MPC problems stored.

In order to obtain these three explicit controllers, we solve the problem of multiparametric optimization with additional parameters that are the Lagrange multipliers in the current and following step of the prediction horizon, and the auxiliary variables $z$ in the current and following step of the prediction horizon, that are used to move the state variables towards their optimal values. These parameters are used in the formulations of decoupled layer's sub-problems. To design these controllers we used the Multi-Parametric Toolbox 3 in MATLAB. As a result, we get the polytopic partitions with 4 regions for the first step, 6 regions for the final step and a polytopic partition with 29 regions for the inner steps of the prediction horizon.

Generated explicit controllers are then used to solve the decoupled layer online. In this step, the point location problem is solved for the current values of the parameters listed above. This means that we look for the region which contains these parameters. The found region is then used to evaluate the explicit control law in the form (2.3). This procedure is repeated over the whole prediction horizon. Results of this procedure are then the predictions of the control action and state variables that are further used in the following steps of the algorithm. These evaluations – coordination layer solution and evaluation of the stopping criteria remain the same as in the implicit form of the

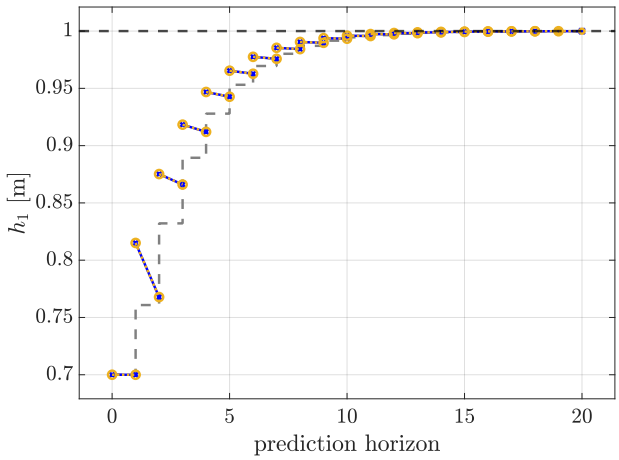



**Figure 3.7:** Trajectory of the liquid level in the first tank over the prediction horizon - results of implicit ALADIN algorithm (blue solid), ALADIN algorithm with implementation of explicit MPC (yellow dotted), non-distributed MPC (grey dashed), and the corresponding steady-state value (black dashed).

MPC formulation.

In the Figures 3.7 - 3.9 we can see the comparison of the results obtained using the implicit MPC algorithm and the explicit MPC described in this Section. Since in the Section 3.2.1 we have found out that the solution obtained after the the $1^{st}$ iteration is sufficiently approaching the reference trajectory (grey dashed line), we decided to show these results, as this may also lead to a reduction in computing time. These figures proved that explicit MPC was implemented correctly because all the trajectories obtained from implicit MPC (blue solid line) and the explicit MPC (yellow dotted line) are identical.

### 3.2.3 Analysis of Obtained Results

Results shown in the Figures 3.7 - 3.9 demonstrate that both algorithms, described in the Section 3.2.1 and Section 3.2.2, give identical trajectories. That is why it makes no sense to compare these two approaches from the point of view of the control quality. However, calculation of process control using ALADIN algorithm with the implicit form of MPC, we noticed, that the time needed to obtain the results might not be

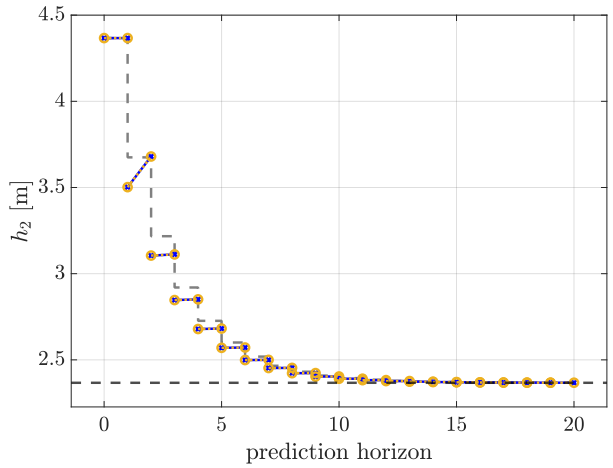



**Figure 3.8:** Trajectory of the liquid level in the second tank over the prediction horizon - results of implicit ALADIN algorithm (blue solid), ALADIN algorithm with implementation of explicit MPC (yellow dotted) non-distributed MPC (grey dashed), and the corresponding steady-state value (black dashed).

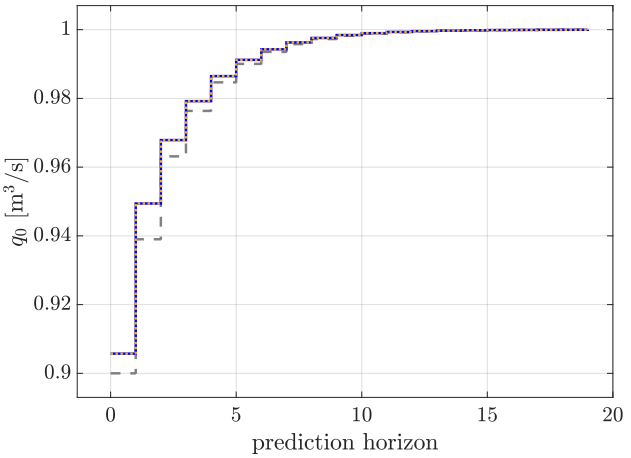



**Figure 3.9:** Trajectory of the control input over the prediction horizon - flow-rate after one iteration of implicit ALADIN algorithm (blue solid), non-distributed MPC (grey dashed), and ALADIN algorithm with implementation of explicit MPC (yellow dotted) after one iteration.

**Table 3.1:** Computational time for 15 iterations of algorithm.

| | Offline Phase [s] | Decoupled Layer [ms] | Coordination Layer [ms] | Total Online Time [ms] |
|---|---|---|---|---|
| Implicit MPC | - | 619 | 30 | 650 |
| Explicit MPC | 3.49 | 38 | 30 | 69 |

short enough to use this method for the systems with fast dynamics. Based on this we decided to take the benefits of explicit MPC and use them to shorten the optimization time in the decoupled layer.

In Table 3.1, there are mean values of times needed to calculate fifteen iterations of ALADIN algorithm for implicit and explicit MPC. As we can see the decoupled layer contributes the most to the overall computational time of the implicit form - 95.3 %, whilst solving the optimization problem of the coordination layer is only 4.7 %. Despite this, the total time of this algorithm is still short enough for the input action to be applied to the controlled system within its sampling time.

The computational time of the explicit MPC is split into the offline phase and the online phase. Within the offline phase, there are three controllers constructed for the given MPC settings. In the online phase, the solution of the decoupled layer is reduced to looking for the region, in which the current values of states are located and evaluating the control law.

Since the construction of the controllers is performed only once, at the beginning of computation, we do not really mind the fact that this phase takes nearly 3.5 s, as this step helps us to spare more time in the solution of the decoupled layer. Using the principles of explicit MPC, the online phase makes up little more than half of the overall time spent online. The solution of the decoupled layer is more than 16-times faster than it is using implicit MPC. Since, in this thesis, we do not try to modify also the coordination layer, so the time needed for its evaluation remains the same and the overall only time is almost 10-times shorter.

Since the results of the previous sections point to the possibility of implementing the optimisation result after the first iteration, it is also worth mentioning the comparison of computation times needed in this case. As we can see in the Table 3.2, if we consider the sub-optimal solution obtained from the first iteration sufficient enough, the time needed to solve the optimization problem is approaching the time of non-distributed MPC. It is important to note that these times are obtained for a small-scaled system of two fluid reservoirs, which we can easily control even using a non-distributed

MPC. However, in the case of larger systems, a numerical algorithm of distributed optimisation using explicit MPC will be a more effective approach. Moreover, the code of the current implementation of ALADIN algorithm was not optimized to high performance, yet.

**Table 3.2:** Computational time for 1 iteration of algorithm.

| | Offline Phase [s] | Decoupled Layer [ms] | Coordination Layer [ms] | Total Online Time [ms] |
|---|---|---|---|---|
| Implicit MPC | - | 41 | 2 | 43 |
| Explicit MPC | 3.49 | 3 | 2 | 5 |
| Non-distributed MPC | - | - | - | 3 |

Even though shortening the evaluation time is undeniably a great benefit of explicit MPC, in the practical implementations, it is necessary to take into consideration also the memory footprint. Table 3.3 shows the memory footprint of the optimization problems of the decoupled layer and also of the coordination layer. Since the default data type for numeric values in MATLAB is double, all the numeric variables are stored as 64-bit (8-byte) double-precision floating-point values. MATLAB constructs the double data type according to IEEE Standard 754 for double precision [8]. As we mentioned above, the coordination layer is the same for the implicit and explicit MPC, so we will first discuss this part.

In the coordination layer, we optimize the quadratic optimization problem with linear constraints. The controlled plant has two state variables and one input variable. As the prediction horizon is set to 20, this means we have to solve the objective function for 60 optimized variables. In addition, we need to take into consideration one extra step of the prediction horizon for state variables in order to evaluate the state equation of the model. Therefore, the objective has the dimension $62 \times 1$. The linear constraints are initial conditions and also the state equations of the system model, both for two state variables over the whole prediction horizon. This means we have $(2 \times 20 + 2)$ linear constraints for 62 optimized variables. We also need to take into consideration

**Table 3.3:** Memory footprint.

| | Decoupled Layer | | Coordination Layer | |
|---|---|---|---|---|
| | Number of Doubles [-] | Memory [kB] | Number of Doubles [-] | Memory [kB] |
| Implicit MPC | 1 552 | 12.42 | 2 708 | 21.66 |
| Explicit MPC | 6 261 | 50.09 | 2 708 | 21.66 |

the right side of each constraint, so the final number of doubles is defined by ($62 \times 1 + 42 \times 62 + 42 \times 1$).

Decoupled layer contains 20 optimization problems of 5 optimized variables (one input variable, two current system states and two following state variables). The size of objective functions is the same for all steps of prediction horizon, so there are ($20 \times (5 \times 1)$) doubles. To evaluate the number of doubles in the constraints, we need to consider two cases. In the first step of the prediction horizon, there are two state equations, two initial conditions, and eight constraints for minimal and maximal values of the current and future state values. There are also two constraints for the limits of the input variable. Then the matrix of left-hand side constraints is of the dimension $14 \times 5$ and the right-hand side vector has 14 elements. In the following steps of the prediction horizon, there are no initial conditions for the state variables. This means that the left-hand side matrix has a dimension of $12 \times 5$ and the right-hand side vector $12 \times 1$. The final number of doubles defining the constraints is ($14 \times 5 + 14 + 19 \times (12 \times 5 + 12)$).

In the explicit MPC, it is necessary to store both, the matrices for domains of particular solutions and the matrices defining the control law. These have to be evaluated for all three explicit controllers constructed in the offline phase. As we mentioned before, in the first controller - for the first step of the prediction horizon, 4 regions were created within the polytopic partition. Since in this step we solve the problem for 3 parameters - the auxiliary variable $z$ in the current and in the following control step and the Lagrange multiplier $\lambda$ in the next step of the prediction horizon, all these regions are defined by 6 columns. The controller for the inner steps of the prediction horizon is defined for 29 regions, each with 8 columns, as we consider 4 two-dimensional optimized parameters - the help variable $z$ in the current and in the following step and the Lagrange multiplier $\lambda$ in the current step and in the following step of the prediction horizon. The final step of the prediction horizon is defined by 6 regions with 4 columns. The multiparametric optimization problem in this step is solved subject to 2 parameters - the help variable $z$ in the current step and the Lagrange multiplier $\lambda$ in the current step of the prediction horizon. As a result, we can see in the second line of the Table 3.3, that to control the system using explicit MPC, we need to store 37.67 kB more than in the case of the implicit MPC. This increase of memory footprint is caused by the fact that, we solve the multi-parametric optimization problem with four additional parameters.

We also need to take care of the size of the optimization problem. In this thesis, the controlled plant is a benchmark system of two liquid tanks connected in series. However, if the system would be more complex the memory footprint of the explicit

MPC will be rising rapidly, whilst the increase in the implicit MPC will not be so significant. To implement explicit MPC to the microcontroller, we need to consider additional reduction of complexity, for instance using the approximation with the neural network.

### 3.2.4 Closed-loop Simulation

As we mentioned in the Section 3.2.1, the trajectories of the input variables obtained after one iteration and after at most fifteen iterations of the ALADIN algorithm, are both sufficiently approximating the optimal solution. To better analyse the quality of the control performance using these inputs, we decided to run the closed-loop simulations. For both mentioned cases, were performed simulation with the simulation time set to 30 s. In the first case, we took the input action after the first iteration and apply this to the state-space system of the controlled plant. The generated simulation results can be seen in the Figures 3.10 and 3.12.

In the second case, we took the input of the last performed iteration. Within this approach, we observed that after a few iterations over the simulation horizon, the ALADIN algorithm stopped by activating the stopping criterion of the minimal size of the change of the state variables. Results of control using the input action of the last iteration are displayed in the Figure 3.11 for the control input - the external flow-rate $q_0$, and in the Figure 3.13 for the controlled variable - liquid level in the second tank $h_2$.

As we can see in the pairs of figures referring to the input variable (Figures 3.10 and 3.11) and the output variable (Figures 3.12 and 3.13), respectively, the results of these approaches do not differ significantly. The values of the integral square error criteria also confirm this visual verification, since the value of the ISE criterion for the second approach is only 0.02 % smaller than in the first approach. However, we have to consider also the computational time. In the case if apply to the controlled system the input action from the last iteration of the ALADIN algorithm solved using the implicit MPC, the needed time can be over the sampling time, so we will not be able to apply it to the plant.

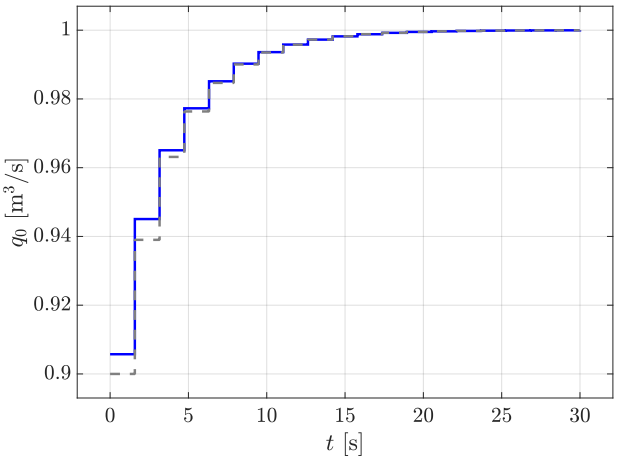



**Figure 3.10:** Closed-loop trajectory of the input after the 1$^{\text{st}}$ iteration of ALADIN algorithm (blue solid) and non-distributed MPC (grey dashed).

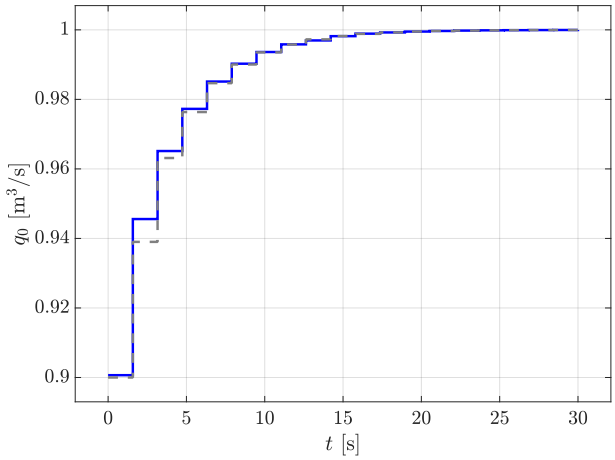



**Figure 3.11:** Closed-loop trajectory of the input after at most the 15$^{\text{th}}$ iteration of ALADIN algorithm (blue solid) and non-distributed MPC (grey dashed).

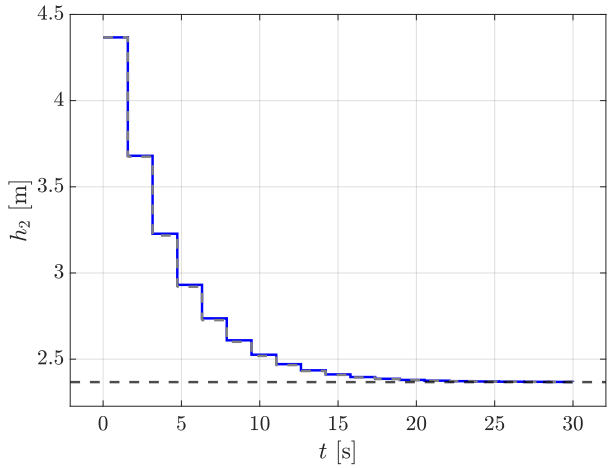



**Figure 3.12:** Closed-loop trajectory of the output after the 1$^{\text{st}}$ iteration of ALADIN algorithm (blue solid), non-distributed MPC(grey dashed), and the reference (black dashed).

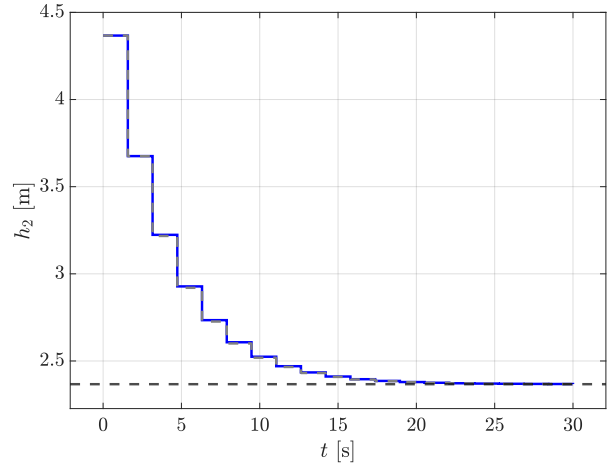



**Figure 3.13:** Closed-loop trajectory of the output after at most the 15$^{\text{th}}$ iteration of ALADIN algorithm (blue solid), non-distributed MPC (grey dashed), and the reference (black dashed).

Chapter 4

# Conclusions

The main aim of the thesis was to design a model predictive control using a method of distributed optimization for a plant in the chemical industry. For these purposes, we decided to use a method that is currently becoming more popular in the scientific community - the *Augmented Lagrangian based Alternating Direction Inexact Newton method*. This approach has been demonstrated to be capable of effective controller design in a distributed fashion, which can aid in implementing the MPC on systems with fast dynamics.

Since we decided to pay increased attention, especially to the development of a calculation algorithm, to verify the accuracy of our method, we decided to use a benchmark system of two tanks with an interaction. The task of the proposed control was to solve the regulatory problem of the controller design. In other words, we wanted to achieve steady-state liquid levels in both tanks in the presence of non-zero initial conditions. Since MPC belongs to a group of control methods based on a mathematical model of a controlled system, the first task was to derive the model of the controlled plant. Once we obtained it, we were able to incorporate it into the optimization problem, solved by the ALADIN method.

The optimization process is performed iteratively in two phases. In the first phase, so-called the decoupled layer, small sub-problems were solved, potentially in parallel fashion, along the entire prediction horizon. In order to obtain a continuous solution, it was necessary to subsequently optimize the obtained results in the second phase, so-called the coordination layer. Here, a quadratic optimization problem with linear constraints is solved, which ensures the minimization of gaps between state values following after each other within the prediction horizon. The use of our proposed algorithm required, in addition to the traditional weight matrices $Q$ and $R$, used in the design of the MPC, also setting of some other tuning parameters. These parameters - the penalty parameter $\rho$ and the scaling matrix $\Omega$, allow us to additionally tune the convergence of the optimization algorithm.

The control of the investigated controlled system was designed using two MPC approaches. First, we use the method of implicit MPC. This included solving the optimization algorithm online, in the real-time control. To verify the obtained results from the numeric optimization method, we decided to compare the prediction trajectories to the optimal one obtained using the YALMIP toolbox with the GUROBI solver in MATLAB programming environment. We analyzed these results in two controller design scenarios. In the first, the results were obtained after one iteration of the ALADIN algorithm, and in the second scenario, we decided to run the iteration at the most for fifteen times. Comparing these results, we found out that our algorithm was correct, as trajectories in both cases converged close to the reference MPC solution. Moreover, we noticed that even the control action designed from the first algorithm might be sufficient enough. It means that this might be a way to spare more time in real-time optimization.

Subsequently, we decided to utilize the advantages of the explicit MPC. As this method can be highly demanding on computational memory in the case of larger optimization problems, we chose to apply it to the decoupled layer as it solves small simple sub-problems. The major benefit of explicit MPC implementation was, that the optimization problems are not solved in real-time. There is an offline phase, in which the explicit controllers are constructed. This means the control law over the polytopic partition was obtained before the real-time control. The solution is stored in the form of the look-up table. We used the Multi-Parametric Toolbox 3 in MATLAB to solve the resulting multi-parametric optimization problems. In the online phase was solved only the point location problem and the obtained solution was used in the coordination layer.

Results of the open-loop simulation of both, implicit and explicit MPCs, were analysed and as we expected the generated results were identical. To see the advantages and disadvantages of the approaches, we additionally analysed the results from the viewpoint of computational complexity, as this is the key factor in practical implementation. We analyzed two aspects - the solver-time needed in the online phase to solve the optimization problem and also the memory footprint necessary to store the explicit solution. In terms of evaluation time, it is obvious that the favorite for practical implementation would be the approach with the use of explicit MPC to solve the problem at the decoupled layer, as this helped us to decrease the overall time of the online phase almost 10-times. However, the microcontrollers have limited memory capacity and usage of the external flash memory significantly increases the calculation time. Based on our simulation results, we evaluated that for explicit MPC, almost 40 kB of extra memory is needed to store. We need to consider the fact that these results are obtained for a simple controlled process with few constraints. For the larger

MIMO system with large number of state and input variables, the memory footprint of the explicit MPC will rise exponentially.

With these results kept in mind, we performed also the closed-loop simulation, so we could analyse how the offsets from the reference control trajectory would affect the overall control. Within the simulation, we investigated the results of control using the input action from the first iteration and the final iteration of the ALADIN algorithm. The simulations aimed to get the liquid levels in both tanks to their steady-state within the simulation time set to 30 s. Obtained close-loop control results were analysed and investigated also by using the ISE criterion. We observed that these control trajectories did not differ significantly. This leads us to the conclusion that it is sufficient to consider just a single iteration of ALADIN algorithm.

In conclusion, we consider the designed distributed optimization algorithm for MPC as a suitable starting point for further research in which we could focus to adapt the algorithm so that it would be guaranteed that the result of one iteration of the ALADIN algorithm would be sufficient for the good performance of the controlled system.Based on the generated simulation results, the use of explicit MPC can help to shorten the computational time significantly. However, we would like to explore the possibilities of approximating the polytopic partition and the associated control law by a neural network in order to decrease the memory footprint. Another possible direction for future research can be a modification of the coordination layer to decrease its evaluation time and also the memory consumption so that the proposed control algorithm would be implementable in the embedded microcomputers used in practice.

Chapter 5

# Resumé

Prediktívne riadenie (z angl. *Model Predictive Control*, MPC) sa od svojho vzniku stalo jednou z populárnych metód pre pokročilé procesné riadenie vo viacerých odvetviach priemyslu [2]. Dôvodom tejto obľuby je okrem iného aj možnosť zahrnúť do návrhu riadenia aj technologické obmedzenia riadiacich, stavových, ako aj výstupných veličín. Ďalšou z výhod je možnosť ladenia tohto typu regulátora vzhľadom na konkrétne požiadavky daného riadeného procesu. Vďaka týmto možnostiam sme schopní minimalizovať náklady na riadenie alebo nežiaduci vplyv na životné prostredie, ale takisto máme možnosť maximalizovať kvalitu získaného produktu.

V spoločnostiach chemického priemyslu sa často nachádzajú veľké procesy, ktorých riadenie môže byť výpočtovo, ale aj časovo náročné. Dostupný čas na výpočet vhodného akčného zásahu je často značne limitovaný, keďže v rámci jednej periódy vzorkovania je potrebné vykonať meranie procesných veličín, samotný výpočet akčného zásahu a následne aj aplikovať vypočítané akčné zásahy do procesu. V takýchto prípadoch už tradičné optimalizačné metódy nemusia postačovať. Svoje uplatnenie tu nachádzajú metódy distribuovanej optimalizácie.

Tieto metódy využívajú skutočnosť, že v rámci spomínaných veľkorozmerných procesov je možné identifikovať symetrické štruktúry, ktoré sa periodicky opakujú v priestore alebo v čase. Na základe toho je možné riadený proces rozdeliť na viacero jednoduchších podprocesov, ktoré sú navzájom prepojené. To umožňuje riešenie takýchto úloh paralelne. V súčasnosti existuje viacero metód distribuovanej optimalizácie. V rámci tejto diplomovej práce sme sa rozhodli analyzovať ťmetódu ALADIN (z angl. *Augmented Lagranian based Alternating Direction Inexact Newton method*), ktorá v porovnaní s ďalšími podobnými metódami dosahuje vyššiu konvergenciu optimalizačných problémov [6].

V rámci našej práce sme sa rozhodli venovať väčšiu pozornosť návrhu výpočtového algoritmu, ktorý bude v sebe kombinovať výhody metód prediktívneho riadenia a dis-

tribuovanej optimalizácie pomocou metódy ALADIN. Z tohto dôvodu sme na overenie správnosti nášho prístupu zvolili systém dvoch zásobníkov kvapaliny s interakciou, ktorý chceme riadiť z nenulových začiatočných podmienok späť do jeho pôvodného ustáleného stavu. Keďže MPC patrí medzi riadiace metódy, vychádzajúce z matematického modelu riadeného systému, našou prvou úlohou bolo odvodiť tento model a zahrnúť ho vo vhodnom tvare do optimalizačného problému riešeného pomocou metódy ALADIN.

Optimalizačný proces sa vykonáva iteračne v dvoch fázach. V prvej fáze, nazývanej *distribuovaná vrstva*, boli malé optimalizačné podproblémy vyriešené v rámci celého predikčného horizontu. Na dosiahnutie spojitého riešenia, bolo potrebné využiť získané výsledky v druhej fáze, nazývanej *koordinačná vrstva*. V rámci tejto vrstvy dochádza k riešeniu kvadratického problému s lineárnymi ohraničeniami, ktorý zabezpečuje minimalizáciu medzier medzi hodnotami stavov, ktoré po sebe nasledujú v jednotlivých krokoch predikčného horizontu. Použitie nášho algoritmu vyžadovalo okrem tradičných váhových matíc $Q$ a $R$ používaných pri návrhu prediktívneho riadenia, aj nastavenie dodatočných ladiacich parametrov. Tieto parametre – penalizujúci parameter $\rho$ a škálovacia matica $\Omega$, nám umožňujú dodatočne ladiť rýchlosť konvergencie optimalizačného algoritmu.

Riadenie daného procesu bolo navrhnuté s využitím dvoch prístupov k MPC. Najskôr sme použili metódu implicitného MPC. Toto si vyžadovalo riešenie optimalizačného problému na distribuovanej vrstve online, v reálnom čase riadenia. Kvôli overeniu kvality výsledkov získaných pomocou numerickej optimalizačnej metódy, sme sa rozhodli porovnať predikované trajektórie s optimálnym riešením získaným využitím toolboxu YALMIP spolu s riešiteľom GUROBI v prostredí MATLAB. Výsledky sme analyzovali vzhľadom na dva prípady. V prvom prípade sme použili výsledky získané už po prvej iterácii algoritmu ALADIN a v druhom prípade sme sa rozhodli vykonať najviac pätnásť iterácií. Porovnaním týchto výsledkov sme zistili, že náš algoritmus je správny, keďže v obidvoch prípadoch sa trajektórie približovali k referenčnému riešeniu MPC. Naviac sme si všimli, že postačujúcim by mohlo byť už riadenie pomocou akčného zásahu vypočítaného v prvej iterácii algoritmu. To by znamenalo možnosť ušetriť viac času v rámci optimalizácie v reálnom čase.

Následne sme sa rozhodli využiť výhody, ktoré poskytuje explicitné prediktívne riadenie. Keďže táto metóda môže byť v prípade väčších optimalizačných problémov veľmi pamäťovo náročná, rozhodli sme sa aplikovať ju na riešenie distribuovanej vrstvy, ktorá je tvorená malými jednoduchými podproblémami. Hlavnou výhodou použitia explicitného MPC bol fakt, že optimalizačné problémy nie sú riešené v reálnom čase, ale v rámci offline fázy, v rámci ktorej sú skonštruované regulátory. To znamená, že

zákony riadenia pre vytvorenú polytopickú partíciu, sú získané vopred a uložené sú v prehľadávacej tabuľke. Tento krok sme uskutočnili s využitím Multiparametrického Toolboxu 3 v prostredí MATLAB. Následne bol v online fáze len vyhľadaný región, v ktorom sa nachádzali aktuálne hodnoty parametrov, potom bol vyhodnotený zákon riadenia a získaný výsledok bol použitý v rámci koordinačnej vrstvy.

Výsledky simulácie implicitného a explicitného MPC v otvorenej slučke boli analyzované a ako sme predpokladali, boli identické. Aby sme videli výhody a nevýhody oboch prístupov, výsledky sme následne porovnali z hľadiska výpočtovej náročnosti, keďže práve tá je kľúčovým faktorom pri praktickej implementácii. Analyzovali sme dva aspekty – čas potrebný na online fázu optimalizácie a tiež vygenerovanú pamäťovú stopu. Z hľadiska času vyhodnocovania je zrejmé, že vhodnejším pre praktickú implementáciu je prístup s použitím explicitného MPC na vyriešenie problému distribuovanej vrstvy, keďže nám to umožnilo skrátiť celkový čas online fázy takmer 10-krát. Avšak, mikrokontroléry majú obmedzenú kapacitu pamäte a použitie externej flash pamäte výrazne predlžuje čas výpočtu. Z nášho skúmania sme zistili, že na riadenie s explicitným MPC je potrebných o takmer 40 kB viac na uloženie. Musíme vziať do úvahy aj skutočnosť, že tieto výsledky boli získané pre jednoduchý riadený proces s niekoľkými obmedzeniami. Pre väčšie MIMO systémy s veľkým počtom stavových premenných bude pamäťová stopa explicitného MPC rásť exponenciálne.

S ohľadom na tieto výsledky sme vykonali aj simuláciu riadenia v uzavretej slučke, aby sme videli, ako odchýlky od referenčnej trajektórie riadenia ovplyvnia celkovú kvalitu riadenia. V rámci simulácie sme porovnávali výsledky riadenia použitím akčného zásahu z prvej iterácie a z najviac 15. iterácie algoritmu ALADIN. Cieľom riadenia bolo dostať hladiny kvapalín v oboch nádržiach z nenulových začiatočných podmienok do ich ustáleného stavu v rámci simulačného času nastaveného na 30 s. Získané výsledky riadenia boli analyzované graficky a tiež pomocou kritéria ISE. Zistili sme, že tieto trajektórie riadenia sa významne nelíšia. To nás vedie k záveru, že z časového hľadiska postačuje uskutočniť iba jednu iteráciu algoritmu ALADIN a použiť získané výsledky.

Na záver je možné zhrnúť, že navrhnutý distribuovaný optimalizačný algoritmus pre MPC považujeme za vhodný východiskový bod pre ďalší výskum, v ktorom by sme sa mohli zamerať na prispôsobenie algoritmu tak, aby bolo zaručené, že výsledok jednej iterácie algoritmu ALADIN bude postačovať pre kvalitné riadenie procesu. Ako sme sa presvedčili, použitie explicitného MPC môže výrazne skrátiť výpočtový čas. Chceli by sme však preskúmať možnosti aproximácie polytopickej partície a riadiaceho zákona pomocou neurónovej siete, aby sa zmenšila pamäťová stopa. Ďalším možným smerom budúceho výskumu môže byť úprava koordinačnej vrstvy, aby sa skrátil čas

jej vyhodnocovania a tiež spotreba pamäte tak, aby bol navrhnutý riadiaci algoritmus implementovateľný do bežných mikropočítačov používaných v praxi.

# Bibliography


[1] Hansi Abeynanda and G.H.J. Lanel. A study on distributed optimization over large-scale networked systems. *Journal of Mathematics*, 2021:1–19, 04 2021.

[2] Francesco Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*. Springer Berlin Heidelberg, 2017.

[3] S. Boyd. Notes on decomposition methods. *Notes for EE364B*, pages 1–36, 01 2007.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 01 2011.

[5] C Cutler and B Ramakar. Dynamic matrix control — a computer control algorithm. 01 1980.

[6] Boris Houska, Janick Frasch, and Moritz Diehl. An augmented lagrangian based algorithm for distributed nonconvex optimization. *SIAM Journal on Optimization*, 26, 04 2016.

[7] Boris Houska and Yuning Jiang. *Distributed Optimization and Control with ALADIN*, pages 135–163. 04 2021.

[8] The MathWorks Inc. Double-precision arrays. `https://www.mathworks.com/help/matlab/ref/double.html`.

[9] Yuning Jiang, Juraj Oravec, Boris Houska, and Michal Kvasnica. Parallel mpc for linear systems with input constraints. *IEEE Transactions on Automatic Control*, 66(7):3401–3408, 2021.

[10] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[11] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, Nov 2021.

[12] Daniel Simon. *Model Predictive Control in Flight Control Design - Stability and Reference Tracking*. PhD thesis, 03 2014.

[13] Peter Sutor Jr and Tom Goldstein. The alternating direction method of multipliers. 2015.

[14] Gergely Takács and Martin Gulan. *Základy prediktívneho riadenia*. Slovenská technická univerzita v Bratislave, 2018.

[15] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47, 05 2019.