SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

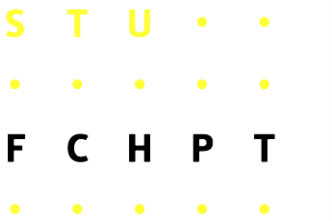
SYSTEMATIC METHOD FOR ANALYSIS OF PERFORMANCE LOSS WHEN USING SIMPLIFIED MPC FORMULATIONS

DIPLOMA THESIS

FCHPT-5414-28512

2010 **Bc. Robert Taraba**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY



SYSTEMATIC METHOD FOR ANALYSIS OF PERFORMANCE LOSS WHEN USING SIMPLIFIED MPC FORMULATIONS

DIPLOMA THESIS

FCHPT-5414-28512

Study programme:	Automation and Informatization in Chemistry and Food Industry
Study field:	5.2.14 Automation
Supervisor:	Ing. Michal Kvasnica, PhD.
Consultant:	MSc Henrik Manum
Work place:	NTNU Trondheim

Bratislava 2010 **Bc. Robert Taraba**

Slovak University of Technology in Bratislava
Department of Information Engineering and Process Control

Faculty of Chemical and Food Technology
Academic year: 2009/2010
Reg. No.: FCHPT-5414-28512

S T U
F C H P T

DIPLOMA THESIS TOPIC

Student:	**Bc. Robert Taraba**
Student's ID:	28512
Study programme:	Automation and Informatization in Chemistry and Food Industry
Study field:	5.2.14 Automation
Thesis supervisor:	Ing. Michal Kvasnica, PhD.
Consultant:	Henrik Manum
Workplace:	NTNU Trondheim

Topic: **Systematic Method for Analysis of Performance Loss when Using Simplified MPC Formulations**

Specification of Assignment:

Aim of this project is to analyze and compare system response using model predictive control (MPC) implemented on a full-order and reduced model. We would like to find the maximum (worst-case) difference between the full-order controller and low-order controller which use full and reduced model, respectively. There are several reasons why usage of the model reduction to reduce model state variables is important, e.g. the more states variables model contains, the more complex the regulator must be. This fact is very important especially for explicit MPC. One of the greatest strength of MPC is the possibility of effective involving constraints on inputs, state and output variables. As it was mentioned before, the main drawback of MPC is in increasing of the complexity in both cases (off-line and on-line) as the size of the system model grows larger as well as the control horizon and the number of constraints are increasing. There exist many algorithms for model reduction, based on different types of reducing methods. Some of them are based on balanced truncation, e.g. Balanced Truncation Algorithm, Square Root Truncation Algorithm and Balancing Free Square Root Truncation Algorithm, etc. The aim is to find the maximum difference between the full-order controller and the low-order controller. To find the maximum difference we will use bilevel programming to solve this problem. We should also compare using MPC controllers with input-blocking, different discretization time, different length of prediction horizon, etc. to make a MPC faster and focus on the connection between performance of control and computational effort.

Thesis length: 40

Assignment procedure from: 15. 02. 2010

Date of thesis submission: 22. 05. 2010

L. S.

Bc. Robert Taraba
Student

prof. Ing. Miroslav Fikar, DrSc.
Head of office

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my thesis supervisor at Institute of Information Engineering, Automation and Mathematics at the Faculty of Chemical and Food Technology of the Slovak University of Technology in Bratislava, Ing. Michal Kvasnica, PhD., for giving me the opportunity to become an exchange student and work on my diploma thesis abroad.

My big gratitude and appreciation goes to my thesis consultant in NTNU Trondheim, PhD Candidate Henrik Manum and to Prof. Sigurd Skogestad, for their patient guidance and support during my stay in Norway.

ABSTRACT

Given diploma thesis deals with the systematic method for analysis of performance loss when using simplified model predictive control formulations. Aim of this thesis is to analyze and compare system response using model predictive control (MPC) implemented on a reference and simplified controller. To find the maximum difference between these controllers and to solve this problem we use bilevel programming. The main drawback of MPC is in increasing of the complexity in both cases (off-line and on-line) as the size of the system model grows larger as well as the control horizon and the number of constraints are increasing. One part of the thesis deals with introduction into MPC and with techniques how to make MPC faster. There are some techniques as model reduction, move blocking, changing the prediction horizon and changing the sampling time, which can be used for simplify MPC problem that makes the optimization problem easier to solve and thus make MPC faster. Using the model reduction to reduce model state variables is important, e.g. the more states variables model contains, the more complex the regulator must be. This fact is very important especially for explicit MPC. Using input blocking we fix the inputs to be constant and using delta-input blocking we fix the difference between two consecutive control inputs to be constant over a certain number of time-steps which reduce degrees of freedom. Reducing prediction horizon we make MPC problem easier to solve. As an example of controlling a typical chemical plant we here consider MPC for a distillation column. Using a bilevel program and model of distillation column we compare these simplify techniques and we focus on the connection between control performance and computational effort. Finally, results are compared and the best way of simplification for our example of plant is found, which turns out to be delta input blocking.

Keywords: analysis of MPC, simplified MPC formulations, analysis of MPC performance

ABSTRAKT

Diplomová práca sa zaoberá metódou na analýzu zníženia kvality riadenia pri použití zjednodušených formulácií prediktívneho riadenia s modelom. Cieľom tejto diplomovej práce je analyzovať a porovnať odozvy systému pri použití prediktívneho riadenia (MPC) na referenčnom regulátore a na zjednodušenom regulátore. Na vyriešenie problému nájdenia maximálneho rozdielu medzi týmito regulátormi používame bilevel programovanie. Hlavnou nevýhodou MPC je že s nárastom veľkosti modelu systému ako aj s nárastom predikčného horizontu a počtu obmedzení sa zvyšuje zložitosť regulátora a to v oboch prípadoch (off-line aj on-line) MPC. Časť práce sa zaoberá úvodom do problematiky MPC a technikami ako urobiť MPC rýchlejšie. Existuje niekoľko techník ako redukcia modelu, blokovanie pohybu, zmena predikčného horizontu, zmena periódy vzorkovania, ktoré môžu byť použité na zjednodušenie MPC problému, čo zabezpečí jednoduchšiu riešiteľnosť optimalizačného problému a tým aj zvýši rýchlosť MPC. Použitie redukcie modelu za účelom redukcie počtu stavov je z tohto hľadiska dôležité, pretože čím viac stavov model obsahuje tým zložitejší regulátor musí byť. Tento fakt je veľmi dôležitý najmä pre explicitné MPC. Použitím blokovania vstupov fixujeme vstupy na konštantnú hodnotu a použitím blokovania zmeny vstupov fixujeme zmenu medzi dvoma po sebe nasledujúcimi vstupmi na konštantnú hodnotu a tým znižujeme počet stupňov voľnosti. Redukciou predikčného horizontu urobíme MPC problém jednoduchšie riešiteľný. Ako príklad riadenia typického chemického zariadenia uvažujeme MPC pre destilačnú kolónu. Použitím bilevel programu a modelu destilačnej kolóny porovnávame zjednodušujúce techniky a zameriavame sa na vzťah medzi kvalitou riadenia a výpočtovou náročnosťou. Na uvedenom príklade destilačnej kolóny porovnávame výsledky rôznych zjednodušujúcich techník a prezentujeme najlepšie riešenie, ktorým sa ukázalo byť blokovanie zmeny vstupov.

Kľúčové slová: analýza MPC, zjednodušené formulácie MPC, analýza kvality riadenia MPC

CONTENTS

LIST OF APPENDICES..... 9
LIST OF SYMBOLS AND ABBREVIATIONS..... 10
LIST OF FIGURES..... 11
1 INTRODUCTION..... 13
2 INTRODUCTION TO MODEL PREDICTIVE CONTROL..... 15
2.1 Model Predictive Control..... 15
2.2 General Formulation of Optimal Control Problem..... 18
2.2.1 Objective Function..... 20
2.2.2 Model of the System..... 22
2.2.3 Constraints..... 24
2.3 How to Make MPC Faster..... 26
2.3.1 Move Blocking..... 26
2.3.1.1 Input Blocking..... 27
2.3.1.2 Delta Input Blocking..... 30
2.3.2 Model Reduction..... 33
2.3.2.1 Balanced Representation..... 35
2.3.2.2 Truncation..... 36
2.3.3 Change of the Prediction Horizon..... 36
2.3.4 Change of the Sampling Time..... 37
2.4 Karush-Kuhn-Tucker Conditions..... 38
3 IMPLEMENTATION OF THE MODEL WITH DISTURBANCES IN MPC..... 40
3.1 Model of the Distillation Column..... 40
3.1.1 Disturbance Model..... 42
3.2 Formulation of the MPC Problems..... 43
3.2.1 Formulation of Problem 1..... 43
3.2.2 Formulation of Problem 2..... 43
3.2.3 Formulation of Problem 3..... 46
3.3 Implementation of the MPC Problems..... 48
3.3.1 Implementation of Problem 1..... 48
3.3.2 Implementation of Problem 2..... 48

3.3.3 Implementation of Problem 3 48
3.4 Comparison of the Solutions to the MPC Problems 49
3.5 Conclusion 50
4 WORST-CASE ERROR ANALYSIS 51
4.1 Model Reduction Worst-case Error Analysis 51
4.1.1 Simulations 55
4.1.1.1 WCE for a Set of Different Reduced-order Models 56
4.1.1.2 Closed Loop Simulation 57
4.1.1.3 WCE for a set of Different Reduced-order Models with Changing N_{sim} 58
4.1.1.4 The Worst Possible Initial Disturbances 59
4.1.1.5 WCE sum for a set of Different Reduced-order Models with Changing N_{sim} 61
4.1.1.6 Comparison of WCE Sum Using Real Updating Objective Function and MPC Simulation Calculation 62
4.1.1.7 Check of the WCE Using Closed Loop Simulation 65
4.1.2 Conclusion 65
4.2 Move Blocking Worst-case Error Analysis 66
4.2.1 Simulations 67
4.2.1.1 Input Blocking 67
4.2.1.2 Delta Input Blocking 72
4.2.1.3 Comparison of Input Blocking and Delta Input Blocking 76
4.2.2 Conclusion 78
5 COMPARISON OF TECHNIQUES FOR SIMPLIFICATION OF MPC 79
5.1 Example 1 Desired Speed up $\approx$ 25 % 80
5.2 Example 2 Desired Speed up $\approx$ 50 % 82
5.3 Example 3 Desired Speed up $\approx$ 75 % 84
5.4 Conclusion 86
6 CONCLUSION 87
7 RESUMÉ 88
8 REFERENCES 90
9 APPENDICES 93

LIST OF APPENDICES

Appendix A: Numerical values of matrices: *A, B, C, D, B_d, D_d*..... 93

Appendix B: List of software on CD..... 94

LIST OF SYMBOLS AND ABBREVIATIONS

MPC	model predictive control
PID	proportional-integral-derivative controller
LQR	linear quadratic control
CFTOC	constrained finite time optimal control
SISO	single input, single output
IB	Input Blocking
DIB	Delta-Input Blocking
BT	Balanced Truncation
BTA	Balance and Truncate Algorithm
SVD	Singular Value Decomposition
KKT	Karush–Kuhn–Tucker conditions
MILP	mixed-integer linear program
WCE	worst-case error
DOF	degrees of freedom
U	control inputs
U^*	optimal control inputs
T_s	sampling time
x_0	initial states
d_0	initial disturbances
y	outputs
P, Q, R	weight matrices
A, B, C, D, B_d, D_d	system matrices
$\boldsymbol{M}$	move blocking matrix
N_{sim}	simulation steps (simulation time)
N	prediction horizon
n_{full}	number of states in full-order controller
n_{red}	number of states in low-order controller
m	number of inputs

LIST OF FIGURES

Figure 1: Difference between classical control and implicit MPC [7]..... 17
Figure 2: Analogy MPC with driving a car [6].....17
Figure 3: Strategy of moving horizon [7]..... 18
Figure 4: A feedback control scheme with implicit solution [4]..... 19
Figure 5: A feedback control scheme with explicit solution [4]..... 20
Figure 6: Convex, concave, non-convex functions [7].....20
Figure 7: Constraints [7].....25
Figure 8: Input blocking type [1 4 3], DOF = 3..... 30
Figure 9: Delta input blocking type [4 3 4 2], DOF = 5.....33
Figure 10: Distillation column controlled with LV-configuration [21].....40
Figure 11. Close loop simulation with the MPC Problem 1, 2, 3.....49
Figure 12: WCE for a set of different reduced order models..... 56
Figure 13: Closed loop simulation for the full order controller $(n_{full} = 16)$ and low order controller $(n_{red} = 4)$ and with number of simulation steps $N_{sim} = 10$ 57
Figure 14: WCE for a set of different reduced order models with changing $N_{sim} = 1-20$ 58
Figure 15: WCE for a set of different reduced order models with changing $N_{sim} = (1,4,8,12,16,20)$ 59
Figure 16: Using initial disturbances d_{01}, d_{02} for calculating WCE for reduced model n_{red} = 10 and with changing $N_{sim} = (1,4,8,12,16,20)$ 60
Figure 17: Sum of WCE for a set of different reduced order models with changing $N_{sim} = 1-20$ 61
Figure 18: Sum of WCE for a set of different reduced order models with changing $N_{sim} = (1,4,8,12,16,20)$ 62

Figure 19: Comparison real sum of WCE (11) and sum of WCE (12) obtain from disturbances calculated in the last simulation step 63
Figure 20: Zoom 1 of figure 8..... 63
Figure 21: Zoom 2 of figure 8..... 64
Figure 22: Comparison real sum of WCE (11) and sum of WCE (12) obtain from disturbances calculated in the last simulation step $N_{sim} = 10$ 64
Figure 23: Compare worst-case errors for a set of different reduced order models using $N_{sim} = 18$ reached as a solution of bilevel problem and it closed loop check..... 65
Figure 24: WCE for a set of DOF using different IB..... 68
Figure 25: WCE for a set of DOF using different IB Zoom..... 68
Figure 26: Predicted inputs with IB type = [4 4] and DOF = 2..... 69
Figure 27: Predicted inputs with IB type = [1 2 2 3] and DOF = 4..... 69
Figure 28: IB types for same degree of freedom – free inputs at the beginning.... 70
Figure 29: IB types for same degree of freedom – free inputs in the end..... 71
Figure 30: WCE for a set of DOF using different DIB..... 72
Figure 31: WCE for a set of DOF using different DIB Zoom..... 73
Figure 32: WCE for a set of DOF using different DIB free inputs in the end..... 73
Figure 33: Predicted inputs with DIB type = [8] and DOF = 2..... 74
Figure 34: Predicted inputs with DIB type = [6 2 2] and DOF = 4..... 74
Figure 35: DIB types for same degree of freedom – first free..... 75
Figure 36: DIB types for same degree of freedom – last free..... 75
Figure 37: Comparison IB and DIB for different DOF – first free..... 77
Figure 38: Comparison IB and DIB for different DOF – first free Zoom..... 77
Figure 39: Comparison IB and DIB for different DOF – last free..... 78
Figure 40. Example 1..... 81
Figure 41. Example 1..... 81
Figure 42. Example 2..... 83
Figure 43. Example 2..... 83
Figure 44. Example 3..... 85
Figure 45. Example 3..... 85

1 INTRODUCTION

Model predictive control (MPC) is advanced control technique that has a significant impact on industrial control engineering. Mathematical model of the system is used to calculate predictions of the future outputs and the control inputs are used to optimize the future response of the system. Because of this, it is very important to have model of the system that adequately describes its dynamic properties.

One of the greatest strength of the MPC is the possibility of effectively involving constraints on inputs, states and output variables. On the other hand in both cases (off-line MPC and on-line MPC) as the size of the system model grows larger as well as the control horizon and the number of constraints is increasing then the complexity of MPC is increasing. This means more time to compute optimal control action and bigger hardware requirements.

The first chapter is devoted to MPC introduction and possibilities of simplifying MPC problem. There are some simplification methods, such as Model Reduction, Move Blocking, Change of the Prediction Horizon and Change of the Sampling Time. Relevant question is the trade-off between speed and performance of MPC using reduced model or some other simplify method, because with increasing reduction of degrees of freedom, the control performance is decreasing.

The second chapter deals with implementation of the mathematical model with disturbances into MPC problem and compare three ways of solving the MPC problem like MPC with the model as equality constraints, MPC with the model substituted into the objective function and first-order optimality conditions of the MPC. As an example of the plant we used a typical simple distillation column by Prof. Skogestad.

The goal of this thesis is to analyze and compare system response using MPC implemented on a reference and simplified controller. To find the maximum (worst-case) difference between the full-order controller and low-order controller we used bilevel programming to solve this problem.

In the third and fourth chapters of thesis we answered the questions: *What is the worst-case difference between an MPC using the full model and an MPC using the reduced model and what* d_0 *maximizes difference between outputs from full model and reduced model, when we consider* $x_0 = B_d d_0$ *and we will use different simulation time? What is the worst-case difference between an MPC without using* ***move blocking*** *and an MPC using* ***move blocking*** *which we use to make MPC faster? Another question is, which move blocking type gives us less worst-case error, when we compare different types of move blocking?*

At the end in the last chapter, results of worst-case error obtained from using different simplification methods that can be used to speed up the computation of the control action in MPC are compared numerically and graphically.

2 INTRODUCTION TO MODEL PREDICTIVE CONTROL

2.1 Model Predictive Control

Model predictive control (MPC) is a successful control technique that has a significant and widespread impact on industrial process control [3]. MPC is used mainly in the oil refineries and petrochemical industry where taking account of the safety constraints is very important. Currently the MPC covers a wide range of methods that can be categorized using various criteria. In this chapter, we cover the main principle of MPC and ways of making the MPC faster.

One of the greatest strengths of MPC using a model of the system is the possibility to include constraints on inputs, states and outputs variables already in the design of the controller. That is why performance of control is better than standard proportional-integral-derivative (PID) controller, which does not provide physical, safety and other constraints on the input, output and state variables.

As the title (Model Predictive Control) suggests the prediction of the future output of the controlled system MPC is calculated using a mathematical model of the system. Because of this, it is very important to have model of the system that adequately describes its dynamic properties. Some models include models of disturbances directly while others assume that the disturbances are constant.

The idea of MPC is to use the control inputs (U) to optimize the future response of the system while, given the information about current states (x) and disturbances (d). Calculation of the future optimal control input $U^* = \left(u_0^T, u_1^T, \ldots, u_{N-1}^T\right)$ is based on the minimization of the objective function on the prediction horizon. Only the optimal value obtained for the current time is actually implemented. Then the system evolves one sample, new measurements are collected and the optimization is repeated. With a fixed length of the horizon, the horizon is shifted one sample further at each new

measurement as given in Fig. 3. Because of this, MPC is often termed *moving horizon* control [5]. In Fig. 1 the difference between classical feedback control and MPC is shown. Strategy of MPC overcomes drawbacks of other methods, such as linear quadratic control (LQR), that are using optimization with infinity horizon without taking constraints into account.

Strategy of the future forecasting is typical in our everyday life. For instance, one can imagine a situation when driving a car as given in Fig. 2.

Our control tasks:

- stay on the road,
- don't crash into the car ahead of us,
- respect speed limits.

When driving a car, we are looking on the road through the windscreen, it is similar to the predictive control strategy as shown in figure 3.

Inputs are usually signals to the plant that can (e.g. gas pedal, brake pedal) or cannot (e.g. side wind, slope of the road, disturbances) be manipulated. The actual information about the plant is given by *state variables*, such as car speed. Of course, even though this comparison is not absolutely precise, it describes very simply the idea of predictive control, that is trying to control the system (in this case a car) forecasting its future (the next position on the road) using a model of the controlling system (car controllers, acceleration, braking, etc.), while respecting constraints (traffic rules, speed limits, vehicle operating characteristics, etc.) [6].

One of the important elements is the choice of adequate prediction horizon N . Using a prediction horizon too short can cause poor control quality or instability of control. In automobile analogy it is if the driver views only too short of a distance ahead, what could lead to accident (collision with slower car, by not having enough reaction time upon obstacle, etc.) [7]. Another problem is when the controlling system model is not representing the real plant and when there are some random disturbances. Using such mathematical model of the system for the prediction of future outputs calculation could be inaccurate and cause incorrect control inputs. MPC works with discrete time system models. Because of this, we need a good choice of the sampling time T_s value for discretization of our model. Sampling time length is a very important since it is the time when new measurements are made, new prediction calculated and new optimal control

inputs $U^* = \left(u_0^T, u_1^T, \ldots, u_{N-1}^T\right)$ determined. However, sampling time must be short enough so that updated measurements from the plant can be taken. There are some good rules in place on how to set the right sampling time, for example we can use Nyquist-Shannnon sampling theorem.

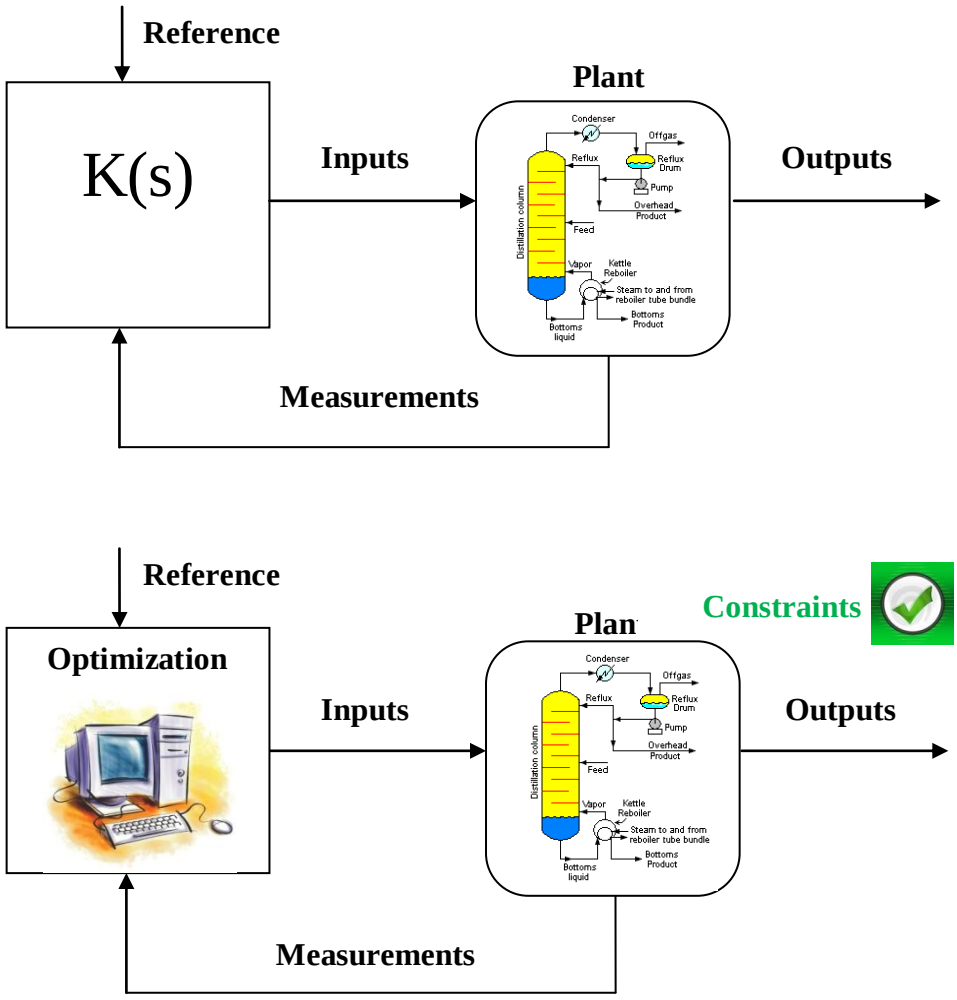


Figure 1: Difference between classical control and implicit MPC [7]

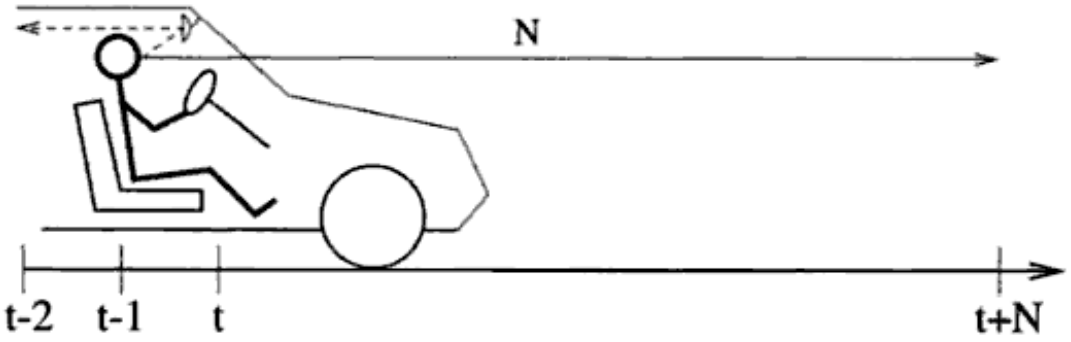


Figure 2: Analogy MPC with driving a car [6]

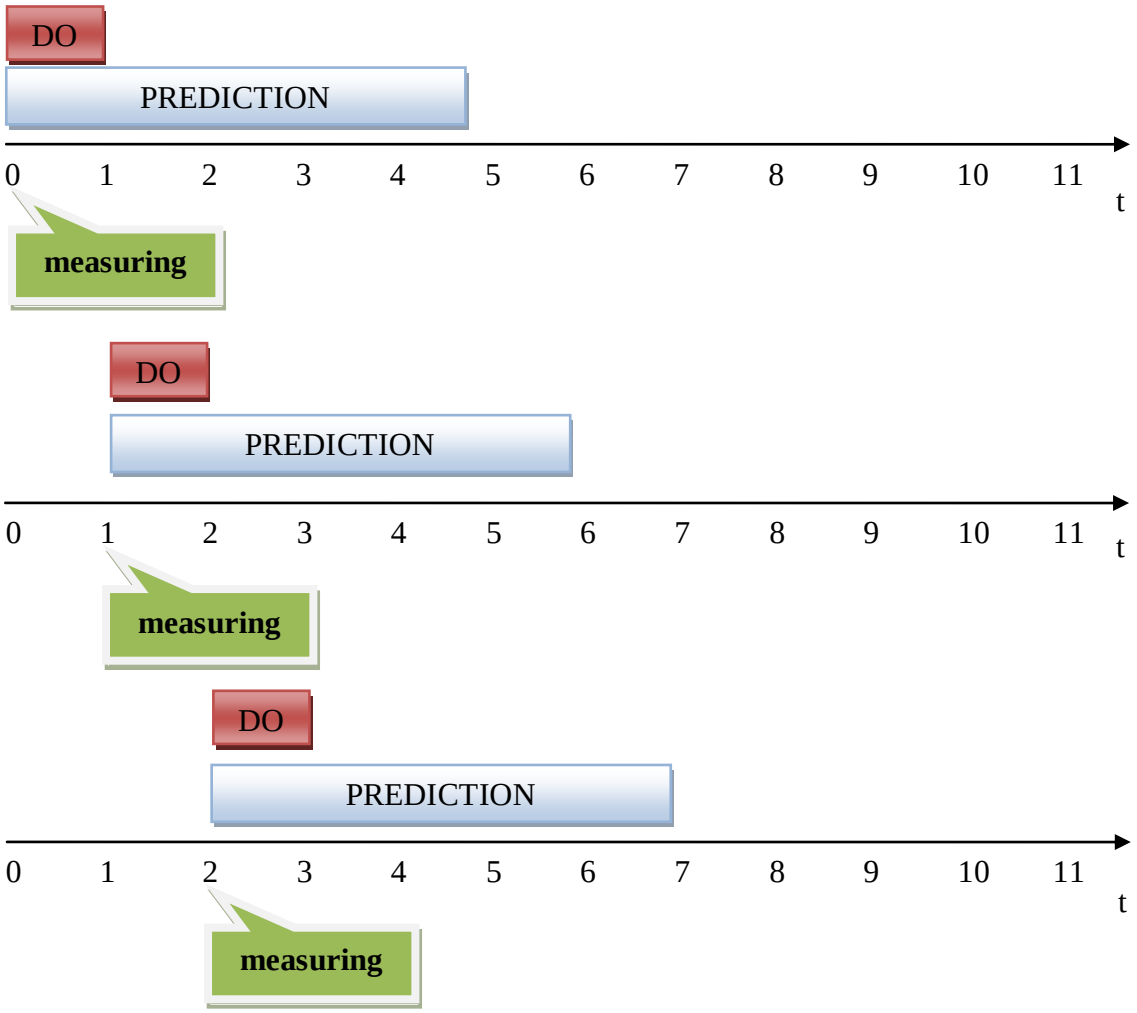


Figure 3: Strategy of moving horizon [7]

2.2 General Formulation of Optimal Control Problem

As is written in [4] the "optimal control problem" is to find optimal control inputs $U^* = \left(u_0^T, u_1^T, \ldots, u_{N-1}^T\right)$ that drive the system from the current initial state x_0 at time t_0 towards the origin.

Optimal control problem [4] is then:

$$\min \quad F(x_N) + \sum_{i=0}^{N-1} L(x_i, u_i) \tag{1a}$$

subject to:

$$x_{i+1} = f(x_i, u_i), \qquad \forall i = 0,\ldots,N-1, \tag{1b}$$
$$x_i \in \mathbf{X}, \qquad \forall i = 1,\ldots,N-1, \tag{1c}$$
$$u_i \in \mathbf{U}, \qquad \forall i = 0,\ldots,N-1. \tag{1d}$$
$$x_0 = x(t) \tag{1e}$$

Expression (1a) is an objective function, (1b) is the process model and, $\mathbf{X}$, $\mathbf{U}$ are the constraints on states and inputs, respectively. This optimal control problem is often called constrained finite time optimal control (CFTOC), because of the constraint on states, inputs and finite horizon N. Predictions have length N steps to the future and control inputs U^* are the optimized degrees of freedom [4].

There are two ways how the optimization problem can be characterized [4]:

Implicit solution: The computed input U^* is given as a sequence of numerical values $u_0^*, u_1^*, \ldots, u_{N-1}^*$, which depend on the particular values of x_0 at specific times within the interval $[t_0, t_N]$.

Explicit solution: The control input U^* is given as a sequence of function typically with plant state as its argument, i.e. $u_0^* = \pi_0(x_0), u_1^* = \pi_1(x_0)$

In Fig. 4 and Fig. 5 feedback controls using implicit and explicit solution are compared.

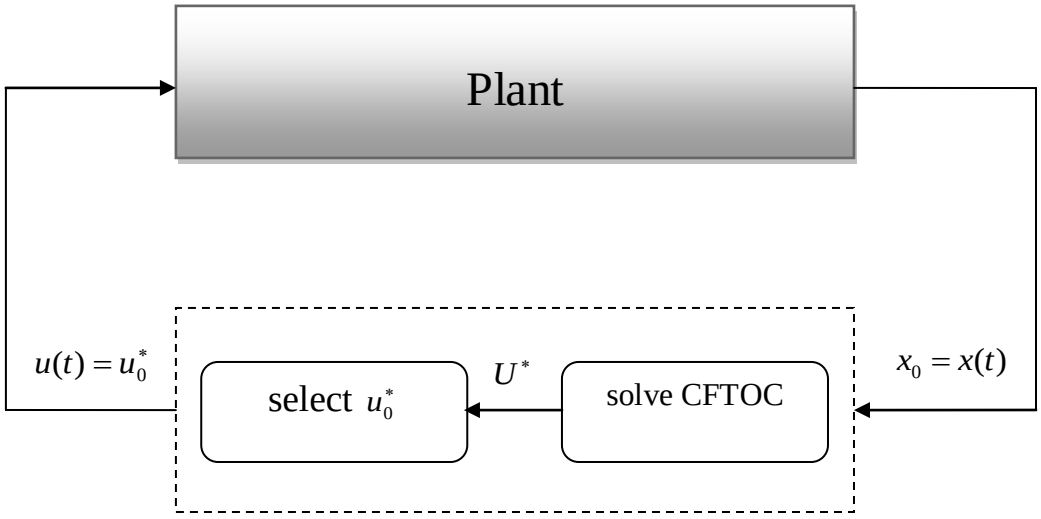


Figure 4: A feedback control scheme with implicit solution [4]

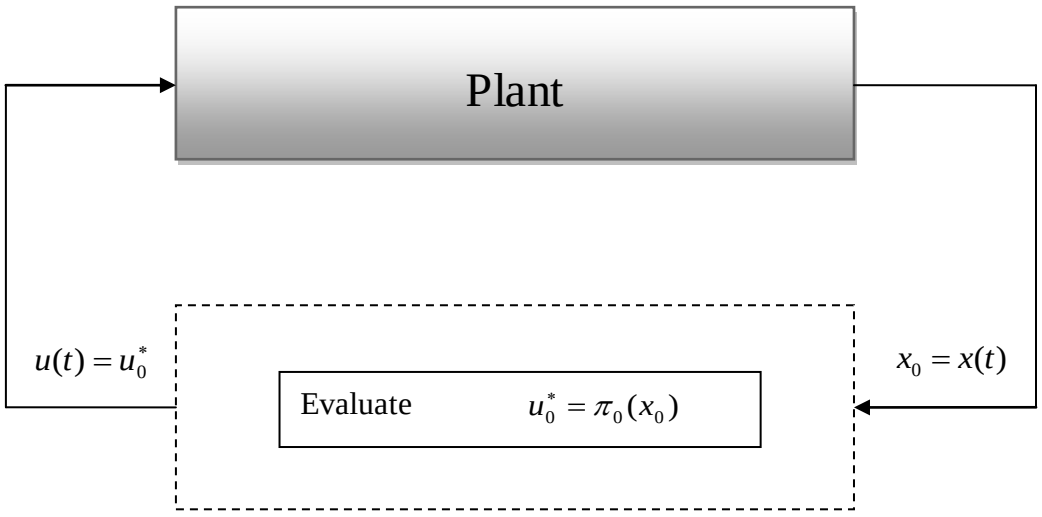


Figure 5: A feedback control scheme with explicit solution [4]

2.2.1 Objective Function

We can divide objective functions on easy and hard to solve.

- EASY : objective function is a convex function
- HARD : objective function is a non-convex function or concave function (minimization of concave function is hard to solve)

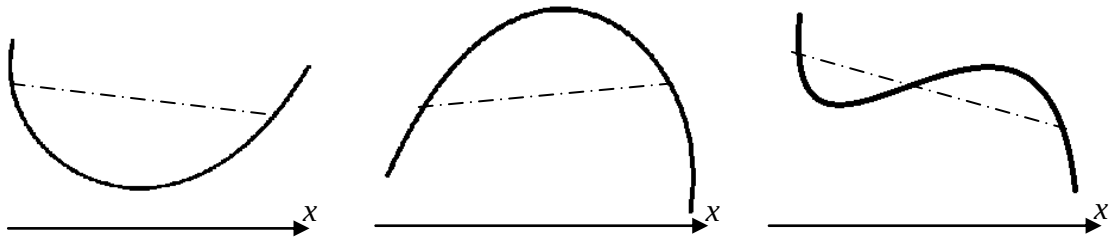


Figure 6: Convex, concave, non-convex functions [7]

An objective function $f(x)$ is convex if for $\theta \in [0,1]$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \tag{2}$$

Usually we can use 3 types of norm of convex objective function.

We can define objective function using general l_p norm as:

$$\min_{\mathbf{u}} J(\mathbf{x},\mathbf{u}) = \left\| Px_N \right\|_p + \sum_{i=0}^{N-1} \left(\left\| Qx_i \right\|_p + \left\| Ru_i \right\|_p \right) \tag{3}$$

For one norm $p=1$, for infinity norm $p=\infty$ and for 2 norm $p=2$.

Objective function is typically quadratic in the states and in the control inputs (4).

$$\min_{\mathbf{u}} J(\mathbf{x},\mathbf{u}) = \frac{1}{2} x_N' P x_N + \frac{1}{2} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i \tag{4}$$

Where N is prediction horizon, P, Q and R are weight matrices for states x_i and inputs u_i respectively. Weight matrices Q, R can be chosen freely, but it is required that Q is positive semi definite $Q \geq 0$ and R is positive definite $R > 0$ so that the objective function becomes convex. These matrices are used to tune the MPC performance and most commonly are diagonal.

Here (5) we consider objective function with Δu formulation and tracking.

$$\min_{u} J(\mathbf{x},\mathbf{u}) = \left\| P(x_{t+N} - x_{ref}) \right\|_p + \sum_{k=0}^{N-1} \left\| Q(k)(x_{t+k} - x_{ref}) \right\|_p + \left\| R(k)\Delta u_{t+k} \right\|_p \tag{5}$$

$$\begin{aligned} s.t. \quad & x_k \in \mathbf{X} \\ & u_k \in \mathbf{U} \\ \Delta u_{t+k} &= u_{t+k} - u_{t+k-1} \end{aligned}$$

Using this formulation, matrices $P, Q(k)$ penalize deviation of the state vector from some reference and $R(k)$ penalize difference of the actual and the last calculated input. Increasing the weights $R(k)$ on the control moves relative to the weights $P, Q(k)$ on the tracking errors has the effect of reducing the control activity. Because of this the elements of $R(k)$ are in some MPC products called *move suppression factors* [2]. We can say that increasing these weight matrices indefinitely will reduce the control activity

to zero, which "switch off" the feedback action. In other words, the penalization of changes in inputs will be so big, that it will not affect the controller. As is stated in [2] if the plant is stable, it will result in a stable system, but not vice versa. Thus with a stable plant, we can expect getting a stable closed loop by sufficient increasing of the control weight. The penalty for doing this will be slow response to disturbances, since it will result in only little control actions. With an unstable plant we can expect an unstable feedback loop, if the $R(k)$ s are increased too much. Because of this there are better ways of ensuring closed-loop stability than using heavy penalty weights $R(k)$.

As it was written using weight matrices $P, Q(k)$ we can penalize states vector or penalize deviation of states vector from some reference. It is possible to penalize some state more heavily than other. That is a way of how to change weight and decide on which states are important for us.

2.2.2 Model of the System

The model of the system represents a mathematical abstraction of the plant's behaviour.

There are different choices of models possible:

- linear (transfer function, state-space, finite impulse response, ...),
- nonlinear (state-space, fuzzy, neural networks, ...),
- hybrid (combination of continuous dynamics and discrete logic).

It is very important to make compromise between quality and complexity of the system model. Complex models are better for predictions, but make optimization more difficult, which takes the optimization problem a lot of time to solve.

Models are very important part of MPC, because they are used to predict the future.

A linear state space model is given by:

$$x_{i+1} = Ax_i + Bu_i, \qquad \forall i = 0,\ldots,N-1, \tag{6a}$$

$$y_i = Cx_i + Du_i, \qquad \forall i = 0,\ldots,N, \tag{6b}$$

where x_i denote states, y_i are outputs (measurements), u_i are controlled inputs.

The state space formulation of model with disturbances is given by

$$x_{i+1} = Ax_i + Bu_i + B_d d_i, \qquad \forall i = 0,...,N-1, \tag{7a}$$

$$y_i = Cx_i + Du_i + D_d d_i, \qquad \forall i = 0,...,N, \tag{7b}$$

where x_i note states, y_i are outputs (measurements), u_i are controlled inputs and d_i are disturbances.

There are many types of disturbance models. For example we can define a disturbance model as:

$$\begin{aligned} d_0 &= given \\ d_1 &= \alpha d_0 \\ &\vdots \\ d_{N-1} &= \alpha^{N-1} d_0 \\ 0 < \alpha &< 1 \end{aligned} \tag{8}$$

When we know initial conditions as initial states x_0, initial disturbances d_0 and the vector of inputs U we can calculate every state x_{i+1} $\forall i = 1,...,N-1$.

$$\begin{aligned} &i = 0 \\ &x_1 = Ax_0 + Bu_0 + B_d d_0, \\ \\ &i = 1 \\ &x_2 = Ax_1 + Bu_1 + B_d d_1 = A\left(Ax_0 + Bu_0 + B_d d_0\right) + Bu_1 + B_d \alpha d_0 = \\ &\quad = A^2 x_0 + ABu_0 + AB_d d_0 + Bu_1 + B_d \alpha d_0, \\ \\ &i = 2 \\ &i = 1 \\ &x_3 = Ax_2 + Bu_2 + B_d d_2 = A\left(A^2 x_0 + ABu_0 + AB_d d_0 + Bu_1 + B_d \alpha d_0\right) + Bu_2 + B_d \alpha^2 d_0 = \\ &\quad = A^3 x_0 + A^2 Bu_0 + A^2 B_d d_0 + ABu_1 + AB_d \alpha d_0 + Bu_2 + B_d \alpha^2 d_0, \\ &\vdots \end{aligned} \tag{9}$$

We can formulate the prediction equation for calculating every state $x_{i+1} \quad \forall i = 1,..., N-1$ as:

$$x_N = \sum_{j=0}^{N-1} \alpha^{N-1-j} A^j B_d d_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} + \sum_{j=1}^{N} A^j x_0 \tag{10}$$

$$\bar{x} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}}_{\bar{x}} = \underbrace{\begin{bmatrix} Bd \\ \alpha Bd + ABd \\ \alpha^2 Bd + \alpha ABd + A^2 Bd \\ \vdots \\ \vdots \end{bmatrix}}_{X} d_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 & 0 \\ AB & B & \cdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ A^{N-2}B & A^{N-3}B & \cdots & B & 0 \\ A^{N-1}B & A^{N-2}B & \ldots & AB & B \end{bmatrix}}_{Y} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N-1} \\ A^N \end{bmatrix}}_{Z} x_0$$

(11)

Considering a model without disturbances, the prediction equation is:

$$x_N = A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \tag{12}$$

$$\bar{x} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}}_{\bar{x}} = \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{bmatrix}}_{X} x_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 & 0 \\ AB & B & \cdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ A^{N-2}B & A^{N-3}B & \cdots & B & 0 \\ A^{N-1}B & A^{N-2}B & \ldots & AB & B \end{bmatrix}}_{Y} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} \tag{13}$$

2.2.3 Constraints

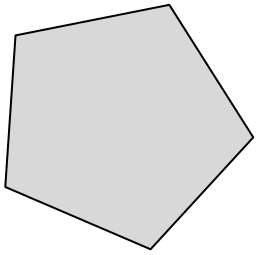
We encounter constraints in our daily live. Physical constraints (temperature, pressure, etc.), safety constraints, environmental constraints but also economical constraints are needed in the industry. It is important to account for safety constraints in the systems control. One of the greatest strengths of MPC is the possibility of effectively involving constraints on inputs, states and outputs variables. We can also make use of constraint on the maximal change of inputs, which makes our controlling more realistic.

The model (4) and (5) are equality constraints and we use them for calculation of predictions. Beside these equality constraints there are inequality constraints too, which define some operating space for allowed values of our variables.
In general we can have two types of constraints. First types are convex constraints that are common in many optimization problems. Second types are non-convex constraints which lead to difficult optimization problems.

Constraints can be divided [7]:

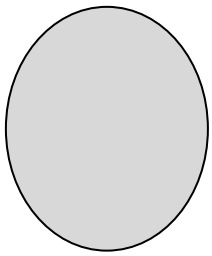
- Polytopic constraints – relatively easy to solve

$x \in P \quad P = \{x \mid Ax \leq B\}$



- Ellipsoids – quadratic constraints which are more difficult to solve

$x \in \varepsilon \quad \varepsilon = \{x \mid (x - x_0)P(x - x_0) \leq r\}$



- Non-convex constraints – extremely hard to solve

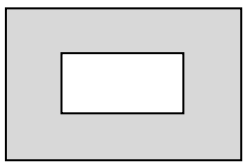


Figure 7: Constraints [7]

2.3 How to Make MPC Faster

In order to make MPC faster and make optimization problem easier to solve we can use some techniques as:

- Move Blocking,
- Change of the Prediction Horizon,
- Change of the Sampling Time Ts,
- Model Reduction.

2.3.1 Move Blocking

In this part we would like to dwell more on the possibility of using move blocking strategies and also compare different types of move blocking. As is stated in [8] it is common practice to reduce the degrees of freedom by fixing the input or its derivatives to be constant over several time-steps. This approach is referred to as 'move blocking'. MPC problem containing *move blocking* is then:

$$\min \quad F(x_N) + \sum_{i=0}^{N-1} L(x_i, u_i) \tag{14a}$$

subject to:

$$x_{i+1} = f(x_i, u_i), \qquad \forall i = 0,..., N-1, \tag{14b}$$
$$x_i \in \mathbf{X}, \qquad \forall i = 1,..., N-1, \tag{14c}$$
$$u_i \in \mathbf{U}, \qquad \forall i = 0,..., N-1. \tag{14d}$$
$$x_0 = x(t) \tag{14e}$$
$$MU = \mathbf{0} \tag{14f}$$

Expression (14a) is an objective function, (14b) is the process model and, $\mathbf{X}$, $\mathbf{U}$ are the constraints on states and inputs, respectively. We here consider *move blocking* constraint (14f) where M is *blocking matrix* consists of ones and zeros and U is vector of optimal inputs.

In the standard MPC problem, the degrees of freedom of a *Receding Horizon Control* problem correspond to the number of inputs n_u multiplied with the length of prediction horizon N. The degrees of freedom are the factor for complexity, regardless of whether the optimization problem is solved on-line or off-line [9, 10].

Move blocking schemes can be divided to [8]:

- Input Blocking (IB),
- Delta-Input Blocking (DIB),
- Offset Blocking (OB),
- Delta-Offset Blocking (DOB),
- Moving Window Blocking (MWB).

2.3.1.1 Input Blocking

Computation complexity of solving the optimization problem in MPC depends directly on the degrees of freedom and it is possible do it with fixing the inputs to be constant over a certain number of time-steps. There are some ways how to implement the input blocking. One of them is using matrix called *blocking matrix* [8].

Using Input Blocking (IB) can be illustrated on one simple example. We have classic MPC problem (3). This problem is solving for the optimal vector $U = [u_0', \cdots, u_{N-1}']' \in R^{Nn_u}$, where n_u is number of inputs multiplied with the prediction horizon *N*. We also consider move blocking constraint (14f).

For example of a SISO with input blocking type $u_1 = u_2 = u_3, u_4 = u_5 = u_6$, prediction horizon $N = 6$, number of inputs $n_u = 2$, it means that every input is vector of two numbers $u_i = (u_{i1}, u_{i2}),\ i \in 1:N$.

$$u_1 = u_2 = u_3 \qquad u_4 = u_5 = u_6 \tag{15}$$

$$\begin{array}{ll} \Downarrow & \Downarrow \\ u_1 - u_2 = \mathbf{0} & u_4 - u_5 = \mathbf{0} \\ u_2 - u_3 = \mathbf{0} & u_5 - u_6 = \mathbf{0} \end{array} \tag{16}$$

Using this input blocking type we reduce the degree of freedom (DOF) from value DOF = 6 to DOF = 2, which makes the MPC problem easier to solve.

From input blocking equation (15) we get this equation which we then use to define our *input blocking matrix*.

$$\underbrace{\begin{bmatrix} I_m & -I_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_m & -I_m & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_m & -I_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I_m & -I_m \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{0}} \tag{17}$$

For calculation of this *input blocking matrix* ***M*** we created a function ***make_blocking*** (Appendix B). Inputs to this function are number of inputs ***nu***, prediction horizon ***N***, type of input blocking ***ibtype***. Output from this function is IB matrix ***M***. Entries of the input blocking type (***ibtype***) define how many consecutive inputs are set to constant. Sum of all entries has to be equal to the prediction horizon N.

Input blocking type can be divided into 2 groups:

1. ibtype = [**number**]

Example 1: ibtype = [5] *N* = 5

Means that first 5 predicted inputs are set to constant $u_1 = u_2 = u_3 = u_4 = u_5$.

Example 2: ibtype = [5] *N* = 10

Means that first 5 predicted inputs are set to constant $u_1 = u_2 = u_3 = u_4 = u_5$ and next 5 inputs are automatically fixed too $u_6 = u_7 = u_8 = u_9 = u_{10}$.

Example 3: ibtype = [1] *N* = 1

If ibtype = 1, then first predicted input is independent.

Example 4: ibtype = [1] *N* = 5

If we have just ibtype = 1 and prediction horizon longer than 1, then all inputs are independent. This means input blocking is not applied.

2. ibtype = [**number1, number2, ...**]

Example 1: ibtype = [3, 2] *N* = 5

Means that fist 3 predicted inputs are set to constant $u_1 = u_2 = u_3$ and that next 2 predicted inputs are set to constant too $u_4 = u_5$.

Example 2: ibtype = [3, 2] *N* = 9

Means that first 3 predicted inputs are set to constant $u_1 = u_2 = u_3$ and next inputs are fixed with input blocking type 2, that means $u_4 = u_5, u_6 = u_7, u_8 = u_9$.

Example 3: ibtype = [1, 4, 3] *N* = 8 nu = 2

First predicted input is independent and next 4 predicted inputs are set to constant $u_2 = u_3 = u_4 = u_5$. Also last 3 inputs are fixed $u_6 = u_7 = u_8$. In figure 8 we can see inputs prediction for both inputs $u_i = (u_{i1}, u_{i2}),\ i \in 1:N$ using IB with ibtype = [1 4 3]. Using this IB we reduce the number of degrees of freedom from 8 to DOF = 3.

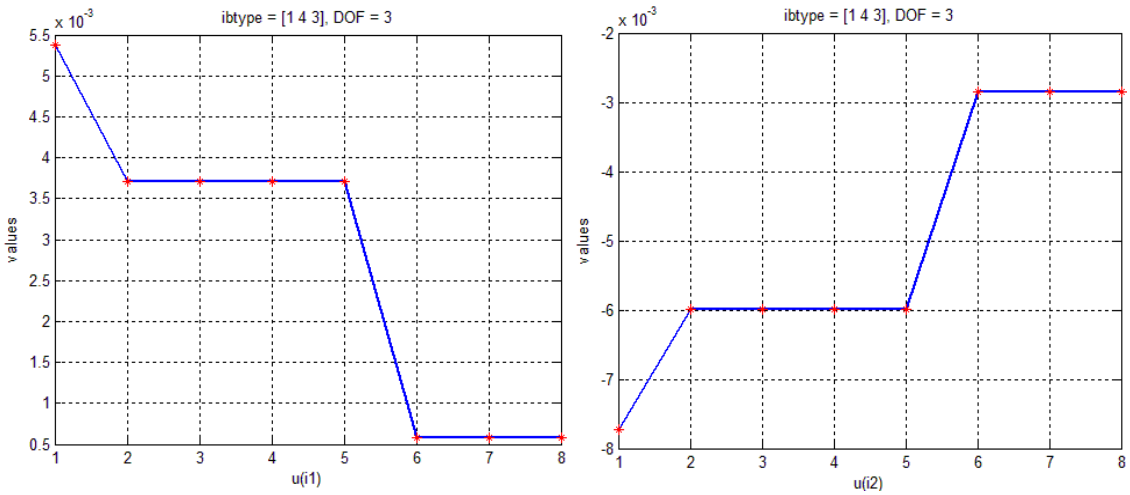


Figure 8: Input blocking type [1 4 3], DOF = 3

2.3.1.2 Delta-Input Blocking

Delta-Input Blocking (DIB) is a method that shows us that instead of just fixing the input to be constant over a certain number or steps, it is too possible to fix the difference between two consecutive control inputs to be constant over several steps. As is written in [8] compared to IB strategy, the DIB strategy may lead to greater flexibility in the controller since only the difference between successive inputs and not the actual inputs are blocked. As the previously presented IB scheme, the DIB has one drawback too. Both of these strategies reduce the complexity of optimization problem but do not guarantee closed-loop stability or feasibility.

The principle of DIB can be illustrated on a very simple example. We consider a SISO system with DIB, prediction horizon $N = 6$, number of inputs $n_u = 2$, meanings that every input is vector of two numbers $u_i = (u_{i1}, u_{i2})$, $i \in 1:N$. For us the first input $u_1 = (u_{11}, u_{12})$ is free now and it can be any number.

$$\begin{aligned} u_2 - u_1 &= \mathbf{C} \\ u_3 - u_2 &= \mathbf{C} \\ u_4 - u_3 &= \mathbf{C} \end{aligned} \tag{18}$$

Equations (18), (19) mean that the difference between these states is constant.

$$\begin{aligned} -u_1 + u_2 + \mathbf{0}u_3 + \mathbf{0}u_4 &= \mathbf{C} \\ \mathbf{0}u_1 - u_2 + u_3 + \mathbf{0}u_4 &= \mathbf{C} \\ \mathbf{0}u_1 + \mathbf{0}u_2 - u_3 + u_4 &= \mathbf{C} \end{aligned} \tag{19}$$

It is possible to rewrite these equations into a matrix form:

$$\underbrace{\begin{bmatrix} I_m & -I_m & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_m & -I_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_m & -I_m \end{bmatrix}}_{\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} \mathbf{C} \\ \mathbf{C} \\ \mathbf{C} \end{bmatrix}}_{\tilde{\mathbf{C}}} \tag{20}$$

Using some elimination process we want to separate constant *C*. For example from last equation (18) we know that $C = u_4 - u_3$. Substituting this equation into first and second equations in (18) we obtain (21).

$$\begin{aligned} -u_1 + u_2 + u_3 - u_4 &= \mathbf{0} \\ \mathbf{0}u_1 - u_2 + 2I_m u_3 - u_4 &= \mathbf{0} \end{aligned} \tag{21}$$

$$\underbrace{\begin{bmatrix} I_m & -I_m & I_m & -I_m \\ \mathbf{0} & I_m & -2I_m & \mathbf{0} \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{0}} \tag{22}$$

From (22) it is clear that we get the same equation like equation as the (14) in IB. In (22) ***M*** is Delta-*Input blocking matrix*.

For calculation this *Delta-Input blocking matrix* ***M*** we created a function ***make_delta_blocking*** (Appendix B). Inputs to this function are number of inputs ***nu***, prediction horizon ***N***, type of delta input blocking ***dibtype***. Output from this function is DIB matrix ***M***.

Entries of the DIB type (***dibtype***) define for how many consecutive inputs are the differences between these inputs constant. Sum of all entries has to be equal to the prediction horizon N.

The DIB type can be divided into 2 groups:

1. dibtype = [**number**]

Example 1: dibtype = [5] *N* = 5

Means that differences between first 5 consecutive inputs are set to constant $u_2 - u_1 = u_3 - u_2 = u_4 - u_3 = u_5 - u_4 = \mathbf{C}$.

Example 2: dibtype = [5] *N* = 10

Means that differences between first 5 consecutive inputs are set to constant $u_2 - u_1 = u_3 - u_2 = u_4 - u_3 = u_5 - u_4 = \mathbf{C}$ and differences between next consecutive inputs are free.

Example 3: dibtype = [2] *N* = 5

In this case only the difference between first 2 consecutive inputs is set to be $u_2 - u_1 = \mathbf{C}$, but that is the same as when not using delta input blocking, as there is always some constant difference between two consecutive inputs. It means that, in this case, the differences between consecutive inputs are free.

2. dibtype = [**number1, number2, ...**]

Example 1: dibtype = [2, 4] *N* = 5

The difference between first 2 consecutive inputs is set to constant $u_2 - u_1 = \mathbf{C1}$ and it means that for first two we do not use the delta input blocking and difference between next 4 consecutive inputs is set to constant $u_3 - u_2 = u_4 - u_3 = u_5 - u_4 = \mathbf{C2}$.

Example 2: dibtype = [4, 3, 4, 2] *N* = 10 nu = 2

The difference between first 4 consecutive inputs is set to constant $u_2 - u_1 = u_3 - u_2 = u_4 - u_3 = \mathbf{C1}$. The difference between next 3 consecutive inputs is constant $u_5 - u_4 = u_6 - u_5 = \mathbf{C}2$ and also the difference between next 4 consecutive inputs is set to constant $u_7 - u_6 = u_8 - u_7 = u_9 - u_8 = \mathbf{C}3$ while last inputs are variable

because only two inputs are set to constant $u_{10} - u_9 = \mathbf{C}4$. In figure 9 we can see inputs prediction for both inputs $u_i = (u_{i1}, u_{i2}),\ i \in 1:N$ using DIB with dibtype = [1 4 3]. This type of DIB reduces the number of degrees of freedom from 10 to DOF = 5.

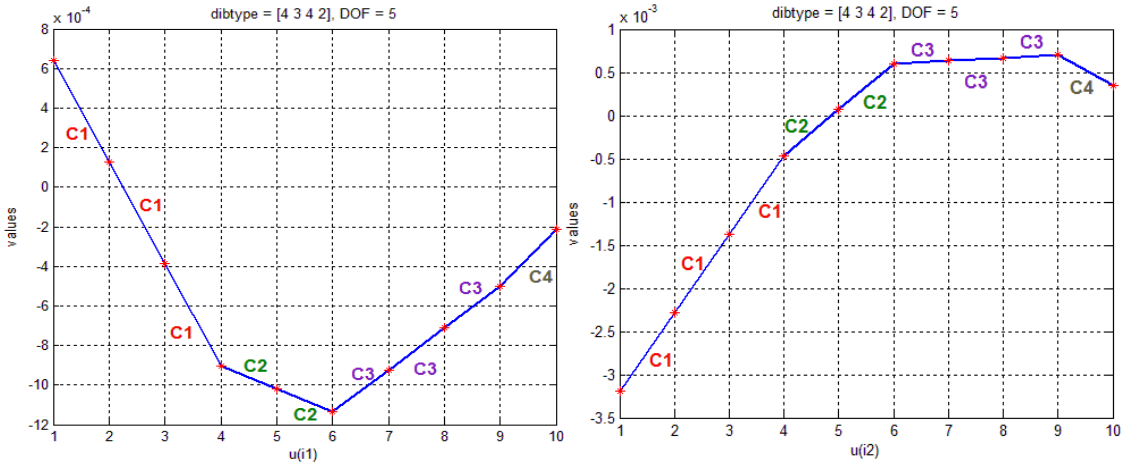


Figure 9: Delta input blocking type [4 3 4 2], DOF = 5

2.3.2 Model Reduction

As mentioned, the MPC controller uses mathematical model to obtain a prediction of outputs. There are many types of models complexity. From models consisting of few states to models which containing many states. With rising number of states in the model, also the complexity of this model grows and, of course, the more complex the MPC controller gets. In other words, as stated in [12], the main drawback of MPC is the large increase in controller complexity as the optimization problem increases. Thus it takes longer time to compute the sequence of optimal control actions. For this reason, usually the low-order models are used with small number of constraints and short control horizons. But applications of this simplification cause control performance loss.

A challenging question is whether it is possible to simplify these complex models and make MPC faster by using some kind of model states reduction. Another relevant question is the trade-off between speed and performance of MPC using reduced model. Answer to first question can be found in [12], [13], [14], [15], [16], where it is also mentioned that the goal of model reduction methods is to derive a model of low order (less number of states) that approximates the input-output behavior of the original

model in an appropriate manner. There are some methods for model reduction. These methods can be divided to stable or unstable systems or to methods based on stochastic or deterministic principles. Another big group of model reduction methods is Truncation methods.

This group of methods includes [13]:

- Balanced Truncation (BT)
- Balance and Truncate Algorithm (BTA)
- Square Root Truncation Algorithm (SRTA)
- Balancing Free Square Root Truncation Algorithm (BFS-RTA)

Other model reduction techniques are Optimal Hankel Model Reduction, or LQG Balanced Truncation.

In this project we will use the Balanced Truncation (BT) [16] as an example of a model reduction scheme that can be than analyzed using our program in order to get the optimal reduction. Main principle of methods like Balanced Truncation or Square Root Truncation Algorithm is to compute Lyapunov functions that wouls satisfy stability of system [13]. Afterwards, Cholesky factorization and Singular Value Decomposition (SVD) is used for choosing the states with the biggest influence in model. With application of truncation we obtain a reduced model.

We consider a continuous linear system [17]:

$$\begin{aligned} &\dot{x} = A^c x + B^c u, \qquad\qquad y = C^c x + D^c u \\ &x \in R^{n_x}, y \in R^{n_y}, u \in R^{n_u}. \end{aligned} \tag{13}$$

Balanced truncation is well known for preserving stability. When we consider that the original model of the system is asymptotically stable, balanced truncation produces asymptotically stable reduced models. Controllability and observability are also preserved in the model reduction process [16].

The BT model reduction method consists of two steps which is clear from the name of this method. First step is called balancing and its aim is to find a balanced representation of system we would like to reduce (13). Second step is truncation of the states corresponding to the smallest Hankel singular values of the balanced representation [17].

2.3.2.1 Balanced Representation

As an example of balanced system we can say that the system is balanced when the states that are excited most by input are at same time the states that produce the most output energy [12]. The gramians can be found by solving the Lyapunov equations below. The controllability and observability gramians of a linear system are defined [16]:

$$A^c W_c + W_c A^{c\prime} + B^c B^{c\prime} = 0 \tag{14}$$

$$A^{c\prime} W_0 + W_0 A^c + C^{c\prime} C^c = 0 \tag{15}$$

$$W_c, W_0 \geq 0 \tag{16}$$

A balanced representation (13) is obtained through a transformation matrix *T*, such that $\overline{W_c}$ and $\overline{W_0}$ (of the transformed system) are equal. Let *z* donate the states of the balanced system, i.e. $z = Tx$.

It can be shown that

$$\begin{aligned} \overline{W_c} &= \overline{W_0} = diag(\sigma_1, \sigma_2, ..., \sigma_{n_x}) \\ \overline{W_c} &= T W_c T^{-1} \\ \overline{W_0} &= (T^{-1})' W_c T^{-1} \end{aligned} \tag{17}$$

The diagonal elements $\sigma_{i,k} = 1,2,...,n_x$ are called the system's Hankel singular values of the balanced representation, ordered according to $\sigma_1 > \sigma_2 > ... > \sigma_{n_x} > 0$.

2.3.2.2 Truncation

Main purpose of truncation is to cut off states that are not useful for system, i.e. have no major influence on the model behaviour and to keep only states that are important for our model.

Let $z' = [z_1' \; z_2']$. In balanced truncation we simply delete z_2 from the vector of balanced states z. Denote T_l and T_r as

$$T = \begin{bmatrix} \overbrace{\begin{bmatrix} T_{11} & \dots & T_{1n} \\ \vdots & & \vdots \\ T_{\tilde{n}1} & \dots & T_{\tilde{n}n} \end{bmatrix}}^{T_l} \\ \vdots \\ T_{n1} \quad \dots \quad T_{nn} \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} \underbrace{\begin{bmatrix} T_{11}^{-1} & \dots & T_{1\tilde{n}}^{-1} \\ \vdots & & \vdots \\ T_{n1}^{-1} & \dots & T_{n\tilde{n}}^{-1} \end{bmatrix}}_{T_r} \dots & T_{1n}^{-1} \\ & \vdots \\ \dots & T_{nn}^{-1} \end{bmatrix} \tag{18}$$

We can now express the balanced and truncated result as

$$\begin{aligned} \dot{z}_1 &= T_l A^c T_r + T_l B^c u \\ \bar{y} &= C^c T^r z_l + Du \end{aligned} \tag{19}$$

and finally $z_l = T_l x$.

2.3.2 Change of the Prediction Horizon

Another approach of how to reduce the degrees of freedom is to use different control and prediction horizons, i.e. the inputs are kept constant beyond a certain point in the prediction horizon, or a linear controller is used beyond that point [8].

MPC has an internal model that is used to predict the behaviour of the plant, starting at the current time, over a future prediction horizon. Predicted behaviour depends on input trajectory that is calculated over that prediction horizon. Inputs promising the best predicted behaviour are then selected and applied to the system [2]. Length of the prediction horizon is the number of steps that optimal inputs are calculated over. Longer length of the prediction horizon provides better performance of control, but simultaneously with longer prediction horizon also the number of decision variables

grows, and this increases the complexity of the optimization problem. On the other hand using too short prediction horizon can cause poor control quality or instability of control. Shortening of the prediction horizon is one way of making the MPC faster, but shorter prediction increases the risk that the control performance will not be satisfactory.

2.3.3 Change of the Sampling Time

Strategy of moving horizon or strategy of future prediction is based on mathematical model of the system to be controlled. MPC works with discrete time system models. Because of this it is necessary to discretize the mathematical model. For this reason the right choice of sampling time T_s is needed for discretization of our model.

The main idea of how to use changing sampling time T_s to make MPC faster is very simple. One of these techniques is described in [11] where the optimization is repeated at each time-step by dividing the prediction horizon into two parts. In the first half of prediction horizon is the sampling rate doubling and the second part of the solution is keeping fixed, until a reasonable sampling time is reached. If we double the sampling time Ts, it will reduce the prediction length by a factor of 2. Therefore the speed-up in terms of sampling time can be measured in the prediction length N. This method shows one major drawback in loss of quality of the model, which transforms into less precise description of the real system. In the worst case, the model can lose its dynamic and will be describing only steps between steady states. Also we cannot omit the fact that the length of sampling time is very important since during this time the new measurements are taken and also new prediction and calculation of optimal inputs is realized.

2.4 Karush-Kuhn-Tucker Conditions

The Karush–Kuhn–Tucker conditions (also known as the Kuhn-Tucker or KKT conditions) are very important for solving constrained optimization problems. The conditions are named after William Karush, Harold W. Kuhn, and Albert W. Tucker and were described in a 1951 paper of Kuhn and Tucker [19], though they were derived earlier (and independently) in an unpublished 1939 master's thesis of W. Karush.

The KKT conditions are the first-order conditions on the gradient for an optimal point. It is a generalization of the method of Lagrange multipliers to inequality constraints. Lagrange multipliers extend the unconstrained first-order condition (derivative or gradient equal to zero) to the case of equality constraints; KKT adds inequality constraints. KKT conditions are necessary for the local optimality of a feasible point in a constrained optimization problem [20].

It is about minimizing functions subject to constraints on the variables. A general formulation for these problems is [18]:

$$\min_{x \in R^n} f(x) \quad \text{subject to} \begin{cases} c_i(x) = 0, & i \in \mathrm{E}, \\ c_i(x) \geq 0, & i \in \mathrm{I}, \end{cases} \tag{20}$$

where f and functions c_i are all smooth, real-valued functions on a subset of R^n, I and E are two finite sets of indices. f is the objective function, while c_i, $i \in \mathrm{E}$ are the equality constraints and c_i, $i \in \mathrm{I}$ are inequality constraints.

As a preliminary to stating the necessary conditions, we define the Lagrangian function for the general problem (20) as:

$$L(x, \lambda) = f(x) - \sum_{i \in \mathrm{E} \cup \mathrm{I}} \lambda_i c_i(x). \tag{21}$$

Following conditions (22) are called *first-order conditions* because they are concerned with properties of the gradients (first-derivative vectors) of the objective and constraint functions.

Suppose that x^* is a local solution of (20), that the function f and c_i in (20) are continuously differentiable, and that the linear independence constraint qualification (LICQ) holds at x^*. Then there is a Lagrange multiplier vector λ^*, with components λ^*_i, $i \in \mathrm{E} \cup \mathrm{I}$, such the following conditions are satisfied at (x^*, λ^*) [18]

$$\nabla_x L(x^*, \lambda^*) = 0, \tag{22a}$$

$$c_i(x^*) = 0, \quad \text{for all } i \in \mathrm{E}, \tag{22b}$$

$$c_i(x^*) \geq 0, \quad \text{for all } i \in \mathrm{I}, \tag{22c}$$

$$\lambda^*_i \geq 0, \quad \text{for all } i \in \mathrm{I}, \tag{22d}$$

$$\lambda^*_i c_i(x^*) \geq 0, \quad \text{for all } i \in \mathrm{E} \cup \mathrm{I}, \tag{22e}$$

The conditions (22) are often knows as the Karush-Kuhn-Tucker conditions, or KKT conditions for short. The conditions (22e) are complementarity conditions; they imply that either constraints i is active or $\lambda^*_i = 0$, or possibly both. In particular, the Lagrange multipliers corresponding to inactive inequality constraint are zero, we can omit the terms for indices $i \notin A(x^*)$ from (22a) and rewrite this condition as [1]

$$0 = \nabla_x L(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in A(x^*)} \lambda^*_i c_i(x^*). \tag{23}$$

Given a local solution x^* of (20) and a vector λ^* satisfying (22), we say that the strict complementarity condition holds if exactly one of λ^*_i and $c_i(x^*)$ is zero for each index $i \in \mathrm{I},$. In other words, we have that $\lambda^*_i > 0,$ for each $i \notin \mathrm{I} \bigcap A(x^*)$.

Satisfaction of the strict complementarity property usually makes it easier for algorithms to determine the active set $A(x^*)$ and converge rapidly to the solution x^*. For a given problem (20) and solution point x^*, there are many vectors λ^* for which the conditions (22) are satisfied [18].

3 IMPLEMETATION OF THE MODEL WITH DISTURBANCES IN MPC

In this part of the Project we compare three ways of implementation and solving the MPC problem using mathematical model of system including disturbances:

1. MPC with the model as equality constraints,
2. MPC with the model substituted into the objective function,
3. First-order optimality conditions of the MPC.

3.1 Model of the Distillation Column

As an example of the plant we will use a typical simple distillation column controlled with LV – configuration which is shown in figure 10. The most important notation is summarized in table 1. Our nonlinear model of a distillation column ("column A") by Prof. Skogestad [21] was linearized using a script in MATLAB ('cola_linearize.m') to obtain a linear model. The model has 82 states (liquid composition and liquid hold up) and we reduced it to 16 states because it is easier to work with a 16 states model. This model contains 2 inputs (reflux L, boilup V) and also 2 disturbances (feed rate F, feed composition z_F). We consider that our disturbances are measured and can be included in the mathematical model.

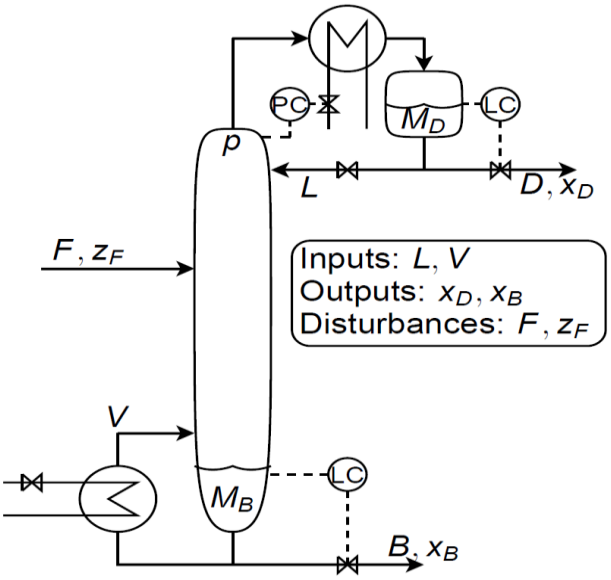


Figure 10: Distillation column controlled with LV-configuration [21]

F	feed rate [kmol/min]
z_F	feed composition [mole fraction]
q_F	fraction of liquid in feed
D and B	distillate (top) and bottoms product flow rate [kmol/min]
x_D and x_B	distillate and bottom product composition (usually of light component) [mole fraction]
L	reflux flow [kmol/min]
V	boilup flow [kmol/min]
N	no. of theoretical stages including reboiler
L_i and V_i	liquid and vapour flow from stage *i* [kmol/min]
x_i and y_i	liquid and vapour composition on stage *i* (usually of light component) [mole fraction]
M_i	liquid and holdup on stage *i* [kmol] (M_B - reboiler, M_D - condenser holdup)
α	relative volatility between light and heavy component
τ_L	time constant for liquid flow dynamics on each stage [min]
λ	constant for effect of vapour flow on liquid flow

Table 1: Notation [21]

The model and assumptions [21]:

- binary separation,
- 41 stages, including rebolier and total condenser,
- each stage is at equilibrium, with constant relatives volatility $\alpha = 1.5$,
- linearized liquid flow dynamics,
- negligible vapor holdup,
- constant pressure.
- constant molar flows
- no vapor holdup

More details about this distillation column: model equations, linearization of this model, steady-state operating point, column temperatures, important MATLAB files, etc. can be found in [22].
MPC is based on a discrete time representation of the system dynamics. Because of this we must discretize our model with sample time *Ts* = 1.

Consider the linear system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k, \qquad k \in \{0,1,2,...\} \\ y_k &= Cx_k + Du_k + D_d d_k \end{aligned} \tag{24}$$

with constraints

$$x_k \in \mathbf{X} \subset \mathrm{R}^{n_x}, y_k \in \mathbf{Y} \subset \mathrm{R}^{n_y}, u_k \in \mathbf{U} \subset \mathrm{R}^{n_u}, \tag{25}$$

where x_k note states, y_k are measurements, u_k are controlled inputs and d_k are disturbances. Further **X**, **Y**, **U** are polytopes.
Matrices *A, B, C, D, B_d, D_d* are given in (Appendix A).
And our constraints on inputs are $-1 \le u_k \le 1$.

3.1.1 Disturbance Model

There are many types of disturbance models. In this part of project we defined a disturbance model as:

$$\begin{aligned} d_0 &= given \\ d_1 &= \alpha \cdot d_0 \\ &\vdots \\ d_{N-1} &= \alpha^{N-1} \cdot d_0 \end{aligned} \tag{26}$$

$$0 < \alpha < 1$$

Disturbance is measured and changing in every step, d_0 is a parameter.

3.2 Formulation of the MPC Problems

Here we use three ways how to formulate and solve MPC problems. In the end we will make results if solving of these problems gives us the same solutions as is expected.

3.2.1 Formulation of Problem 1

MPC formulation:

$$\min_{x,u} \; J(x,u) = \frac{1}{2} x_N' P x_N + \frac{1}{2} \sum_{i=0}^{N-1} x_i' Q x_i + u_N' R u_i$$

subject to (27)

$$\begin{aligned}
& x_{i+1} = Ax_i + Bu_i + B_d d_i, && \forall i = 0,\ldots,N-1, \\
& x_i \in \mathbf{X}, && \forall i = 1,\ldots,N-1, \\
& u_i \in \mathbf{U}, && \forall i = 0,\ldots,N-1, \\
& u_{\min} \le u_i \le u_{\max} && \forall i = 0,\ldots,N-1, \\
& x_0 = 0 \vee given, \\
& d_0 = given
\end{aligned}$$

3.2.2 Formulation of Problem 2

We use linear model to rewrite Problem 1 to the form of Problem 2.

$$\begin{aligned}
& x_{i+1} = Ax_i + Bu_i + B_d d_i, && \forall i = 0,\ldots,N-1, \\
& x_0 = 0 \vee given, \\
& d_0 = given \\
& d_i = \alpha^i \cdot d_0 && \forall i = 1,\ldots,N-1, \\
& 0 < \alpha < 1
\end{aligned} \qquad (28)$$

When we know x_0, d_0 and U we can calculate every state $x_{i+1} \quad \forall i = 1,..., N-1$.

$i = 0$

$$x_1 = Ax_0 + Bu_0 + B_d d_0,$$

$i = 1$

$$\begin{aligned} x_2 &= Ax_1 + Bu_1 + B_d d_1 = A\left(Ax_0 + Bu_0 + B_d d_0\right) + Bu_1 + B_d \alpha d_0 = \\ &= A^2 x_0 + ABu_0 + AB_d d_0 + Bu_1 + B_d \alpha d_0, \end{aligned} \tag{29}$$

$i = 2$

$$\begin{aligned} x_3 &= Ax_2 + Bu_2 + B_d d_2 = A\left(A^2 x_0 + ABu_0 + AB_d d_0 + Bu_1 + B_d \alpha d_0\right) + Bu_2 + B_d \alpha^2 d_0 = \\ &= A^3 x_0 + A^2 Bu_0 + A^2 B_d d_0 + ABu_1 + AB_d \alpha d_0 + Bu_2 + B_d \alpha^2 d_0, \end{aligned}$$

$\vdots$

Formula for calculating every state $x_{i+1} \quad \forall i = 1,..., N-1$:

$$x_N = \sum_{j=0}^{N-1} \alpha^{N-1-j} A^j B_d d_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} + \sum_{j=1}^{N} A^j x_0 \tag{30}$$

Transforming (30) into matrix form we get:

$$\bar{x} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}}_{\bar{x}} = \underbrace{\begin{bmatrix} Bd \\ \alpha Bd + ABd \\ \alpha^2 Bd + \alpha ABd + A^2 Bd \\ \vdots \\ \vdots \end{bmatrix}}_{X} d_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 & 0 \\ AB & B & \cdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ A^{N-2}B & A^{N-3}B & \cdots & B & 0 \\ A^{N-1}B & A^{N-2}B & \ldots & AB & B \end{bmatrix}}_{Y} U + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N-1} \\ A^N \end{bmatrix}}_{Z} x_0 \tag{31}$$

here $\bar{x} = Xd_0 + Yu + Zx_0$.

Objective function (27) can be rewritten in this matrix form:

$$\begin{aligned} \min_{x,u} \; J(x,u) &= \frac{1}{2} x'_N P x_N + \frac{1}{2} \sum_{i=0}^{N-1} x'_i Q x_i + u'_N R u_i = \\ &= \frac{1}{2} \bar{x}' \underbrace{\begin{bmatrix} Q & 0 & \cdots & 0 & 0 \\ 0 & Q & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & Q & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix}}_{\tilde{Q}} \bar{x} + \frac{1}{2} U' \underbrace{\begin{bmatrix} R & 0 & \cdots & 0 & 0 \\ 0 & R & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & R & 0 \\ 0 & 0 & \cdots & 0 & R \end{bmatrix}}_{\tilde{R}} U \end{aligned} \tag{32}$$

By using (30) the MPC problem defined in (27) can be rewritten as:

$$V(x_0, d_0) = \frac{1}{2} x'_0 Y x_0 + \min_u \left\{ \frac{1}{2} U' H U + x'_0 F_x U + d'_0 F_d U \right\}, \tag{33a}$$

subject to $GU \le W + E_x x_0 + E_d d_0$ (33b)

We need to rewrite the MPC problem from formulation (32) to the formulation (33). We will use equation (31) and put it into the problem (32).

$$\begin{aligned} \min_{x,u} \; J(x,u) &= \frac{1}{2} \bar{x}' \tilde{Q} \bar{x} + \frac{1}{2} U' \tilde{R} U = \\ &= \frac{1}{2} \left(Z x_0 + X d_0 + Y U \right)' \tilde{Q} \left(Z x_0 + X d_0 + Y U \right) + \frac{1}{2} U' \tilde{R} U \end{aligned} \tag{34}$$

We define θ as:

$$Z x_0 + X d_0 = \underbrace{\begin{bmatrix} Z & X \end{bmatrix}}_{\theta} \underbrace{\begin{bmatrix} x_0 \\ d_0 \end{bmatrix}}_{z} \tag{35}$$

We can use equation (35), put it in the objective function (34) and we get objective function in form (36).

$$\begin{aligned} &\frac{1}{2} \left(Z x_0 + X d_0 + Y U \right)' \tilde{Q} \left(Z x_0 + X d_0 + Y U \right) + \frac{1}{2} U' \tilde{R} U = \frac{1}{2} \left(\theta z + Y U \right)' \tilde{Q} \left(\theta z + Y U \right) + \frac{1}{2} U' \tilde{R} U = \\ &= \frac{1}{2} z' \theta' \tilde{Q} z \theta + z' \theta' \tilde{Q} Y U + \frac{1}{2} U' Y' \tilde{Q} Y U + \frac{1}{2} U' \tilde{R} U = \\ &= \underbrace{\frac{1}{2} z' \theta' \tilde{Q} z \theta}_{const} + \underbrace{z' \theta' \tilde{Q} Y U}_{[x_0' \; d_0'] \begin{bmatrix} Z' \\ X' \end{bmatrix} \tilde{Q} Y U} + \frac{1}{2} U' \underbrace{\left(Y' \tilde{Q} Y + \tilde{R} \right)}_{H} U \end{aligned} \tag{36}$$

From objective function (36) we can extract matrices H, F_x, F_d .

$$\begin{aligned} H &= Y' \tilde{Q} Y + \tilde{R} \\ F_x &= Z' \tilde{Q} Y \\ F_d &= X' \tilde{Q} Y \\ f' &= d_0' F_d + x_0' F_x \\ f &= F_d' d_0 + x_0' F_x \end{aligned} \tag{37}$$

Now we have matrices H, F_x, F_d which we need to formulate the objective function in problem (33).

We also need to formulate constraints and get matrices G, W (33). We have only one input constraint:

$$u_{\min} \le u \le u_{\max} \tag{38}$$

$$\Downarrow$$

$$u \le u_{\max} = Iu \le 1u_{\max}$$

$$u_{\min} \le u \quad \Rightarrow \quad -u \le -u_{\min} = -Iu \le -u_{\min}1 \tag{39}$$

From (16) we can express G, W as:

$$G = \begin{bmatrix} I \\ -I \end{bmatrix}; W = \begin{bmatrix} u_{\max} \cdot 1 \\ -u_{\min} \cdot 1 \end{bmatrix} \tag{40}$$

3.2.3 Formulation of Problem 3

In formulation of the Problem 3 we need to define the KKT conditions (44) for the problem (33). Problem (33) is a typical problem of a single inequality constraint which we can solve using KKT conditions.

Our optimization problem is:

$$\min_u \left\{ \frac{1}{2} U'HU + x'_0 F_x U + d'_0 F_d U \right\} \tag{41a}$$

Subject to the single inequality constraint:

$$\begin{aligned} GU &\le W + E_x x_0 + E_d d_0 \\ -GU + W + E_x x_0 + E_d d_0 &\ge 0 \end{aligned} \tag{41b}$$

We define the Lagrangian function (42) for our problem using Lagrange multiplier λ and equations (41a) and (41b):

$$L(u, \lambda) = \left(\frac{1}{2} U'HU + x'_0 F_x U + d'_0 F_d U \right) - \lambda' \left(-GU + W + E_x x_0 + E_d d_0 \right) \tag{42}$$

First condition of KKT conditions is gradient (first-derivation vector) of the objective and constraint functions. Here *u* is a local solution of (41) and the functions in (41) are continuously differentiable.

$$\begin{aligned} \nabla_u L(u,\lambda) &= 0 \\ HU + F_x x_0 + F_d d_0 + G'\lambda &= 0 \end{aligned} \tag{43}$$

From (43) we get fist KKT condition (44a). Second KKT condition (44b) is our constraint (41b) and the Lagrange multiplier must be greater than equal to 0.

$$HU + F_x x_0 + F_d d_0 + G'\lambda^* = 0 \tag{44a}$$

$$GU - W - E_x x_0 - E_d d_0 \le 0 \tag{44b}$$

$$\lambda \ge 0 \tag{44c}$$

$$\begin{aligned} \lambda &\le Ms \\ GU - W - E_x x_0 - E_d d_0 &\ge M(1-s) \end{aligned} \tag{45}$$

In (45) we use binary variables $s \in \{0,1\}^{n_W}$ and big-M formulation, where n_W is the number of inequality constraints in (41b).

The big-M is a constant that is large enough such the solution to (44), (45) corresponds to the solution of (33). Big-M reformulations are used to convert a logic or nonconvex constraint to a set of constraints describing the same feasible set, using auxillary binary variables and additional constraints [23].

3.3 Implementation of the MPC Problems

3.3.1 Implementation of Problem 1

We use Yalmip [Löfberg, 2004] under MATLAB to set up the optimization model, objective function with constraints of the Problem 1. To solve Problem 1 we use the function **solvesdp** which is the common function for solving standard optimization problems with Yalmip. The script which we use is in (Appendix B: Script to solve Problem 1).

3.3.2 Implementation of Problem 2

To implement Problem 2 we need matrices $X, Y, Z, \widetilde{Q}, \widetilde{R}$ which are defined in (31) and (32). To calculate these matrices we wrote functions "xyfun.m" and "QRfun.m". Using these functions we can get matrices $X, Y, Z, \widetilde{Q}, \widetilde{R}$ with using different length of prediction horizon N, disturbances parameters d_0 and α, model of system, sample time *Ts* and different weight matrices Q, R. In the script (Appendix B: Script to solve Problem 2) we calculate also matrices H, F_x, F_d (37) and G, W (40) which are necessary to solve the Problem 2 with function **quadprog** which is a function that can solve quadratic programming problems.

3.3.3 Implementation of Problem 3

For implementation and solving Problem 3 we use the script in (Appendix B: Script to solve Problem 3). In this script we also use functions "xyfun.m" and "QRfun.m" as before when we was solving the Problem 2, because we need define matrices H, F_x, F_d and G, W which are part of the KKT conditions. To define a KKT conditions with big-M formulation we use Yalmip and to solve the problem we use the function **solvesdp**.

3.4 Comparison of the Solutions to the MPC Problems

Problems 1, 2, 3 should have the same solution because they represent the same problem.

We here consider MPC for our distillation column example [21] with 16 states, sample time *Ts* = 1, prediction horizon $N = 8$, weight matrices *Q* = diag(ones(16,1),0) and *R* = diag(ones(2,1),0). Disturbances parameters are $d_0 = [0.5;1], \alpha = 0.1$ and the initial state $x_0 = \text{zeros}(16,1)$.

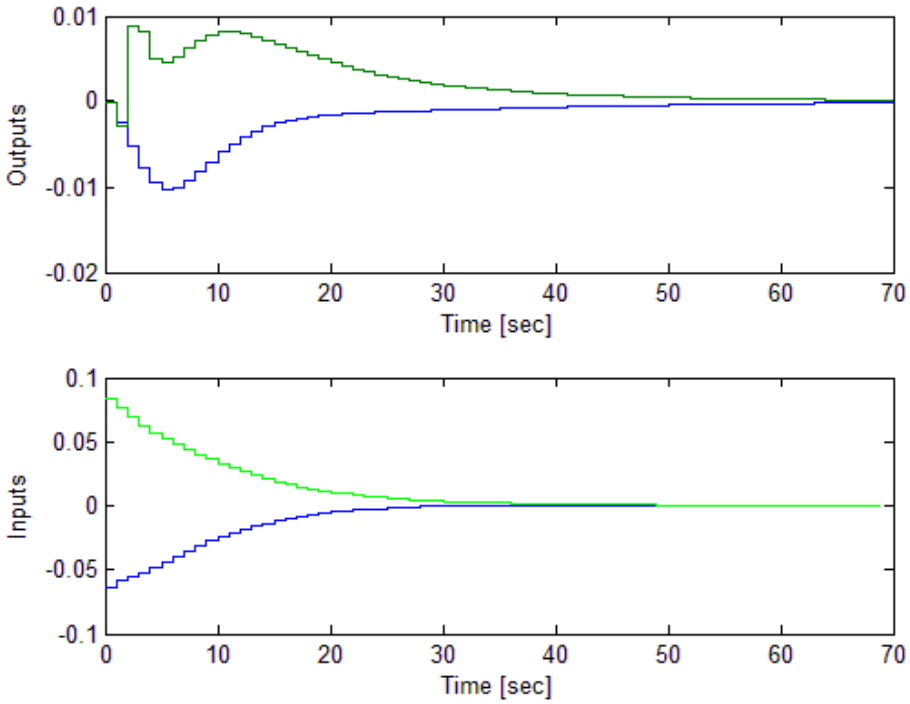


Figure 11. Close loop simulation with the MPC Problem 1, 2, 3

Problems 1, 2, 3 have the same solution because we use only other formulations of the same MPC problem. It is clear in figures 11 where we obtain the same system response from close loop simulations using Problem 1, 2, 3. In Table 2 are optimal inputs obtained as a solution from optimization problems 1, 2, 3.

	Problem 1		Problem 2		Problem 3	
	u1	u2	u1	u2	u1	u2
1	-0.0638	0.0833	-0.0638	0.0833	-0.0638	0.0833
2	-0.0575	0.0771	-0.0575	0.0771	-0.0575	0.0771
3	-0.0558	0.0691	-0.0558	0.0691	-0.0558	0.0691
4	-0.0530	0.0625	-0.0530	0.0625	-0.0530	0.0625
5	-0.0489	0.0572	-0.0489	0.0572	-0.0489	0.0572
6	-0.0443	0.0526	-0.0443	0.0526	-0.0443	0.0526
7	-0.0397	0.0482	-0.0397	0.0482	-0.0397	0.0482
8	-0.0354	0.0441	-0.0354	0.0441	-0.0354	0.0441
9	-0.0313	0.0401	-0.0313	0.0401	-0.0313	0.0401
10	-0.0275	0.0363	-0.0275	0.0363	-0.0275	0.0363
11	-0.0241	0.0328	-0.0241	0.0328	-0.0241	0.0328
12	-0.0210	0.0295	-0.0210	0.0295	-0.0210	0.0295
13	-0.0182	0.0264	-0.0182	0.0264	-0.0182	0.0264
14	-0.0157	0.0236	-0.0157	0.0236	-0.0157	0.0236
15	-0.0135	0.0211	-0.0135	0.0211	-0.0135	0.0211
16	-0.0116	0.0188	-0.0116	0.0188	-0.0116	0.0188
17	-0.0099	0.0167	-0.0099	0.0167	-0.0099	0.0167
18	-0.0084	0.0149	-0.0084	0.0149	-0.0084	0.0149
19	-0.0071	0.0132	-0.0071	0.0132	-0.0071	0.0132
20	-0.0059	0.0118	-0.0059	0.0118	-0.0059	0.0118
⋮	⋮	⋮	⋮	⋮	⋮	⋮
60	0.0002895	0.0003659	0.0002895	0.0003659	0.0002895	0.0003659
61	0.0002761	0.0003443	0.0002761	0.0003443	0.0002761	0.0003443
62	0.0002631	0.0003241	0.0002631	0.0003241	0.0002631	0.0003241
63	0.0002506	0.0003053	0.0002506	0.0003053	0.0002506	0.0003053
64	0.0002385	0.0002877	0.0002385	0.0002877	0.0002385	0.0002877
65	0.0002269	0.0002712	0.0002269	0.0002712	0.0002269	0.0002712
66	0.0002158	0.0002557	0.0002158	0.0002557	0.0002158	0.0002557
67	0.0002051	0.0002412	0.0002051	0.0002412	0.0002051	0.0002412
68	0.0001949	0.0002276	0.0001949	0.0002276	0.0001949	0.0002276
69	0.0001851	0.0002149	0.0001851	0.0002149	0.0001851	0.0002149
70	0.0001757	0.0002029	0.0001757	0.0002029	0.0001757	0.0002029

Table 2: Inputs obtained from Problem1, Problem 2, Problem 3 MPC regulators

3.5 Conclusions

In this part of thesis we wrote short introduction into the formulation and solving MPC using mathematical model of the system including disturbances. We defined our MPC problem using three ways of formulations. At first we used normal formulation of MPC problem with objective function and disturbances. This problem we implemented and solved using Yalmip in Matlab. As a Problem 2 we reformulated the first problem into the quadratic programming problem and then we used function **quadprog** to solve it. For the last way of formulation and solving MPC problem we used KKT conditions and we solved this problem as a typical problem of a single inequality constraint using Yalmip in Matlab. We reached the same solutions of this MPC problem using Problem 1, 2, 3.

4 WORST-CASE ERROR ANALYSIS

4.1 Model Reduction Worst-case Error Analysis

In this chapter we consider model predictive control (MPC) [1] and we would like to answer the question: *What is the worst-case difference between an MPC using the full model* (2) *and an MPC using the reduced model* (3) *and what* d_0 *maximizes difference between outputs from full model* (2b) *and reduced model* (3b) *when we consider* $x_0 = B_d d_0$ *and we will use different simulation time?*

To find the maximum difference between the reference and simplified controller we use bilevel programming. We could calculate the distance between the controllers as $\left\| u_k - u_k^{red} \right\|_\infty$, but we focused on difference in outputs $\left\| Q_y (y_k - y_k^{red}) \right\|_\infty$. We use the infinity norm because then the problem can be reformulated as a mixed-integer linear program (MILP). [17]

$$\begin{aligned} & \max_{d \in D} \ \text{distance}(\mathrm{y_k}, \mathrm{y_k^{red}}) \\ \text{subject to} \quad & \mathrm{y_k} = \arg\min \{\mathrm{MPC\,formulation\ with\ full\ model}\} \\ & y_k^{red} = \arg\min \{\mathrm{MPC\,formulation\ with\ reduced\ model}\} \end{aligned} \tag{46}$$

We don't use an explicit formulation of the controllers, but we simply express them as solutions to optimization problems. Problem (46) can be rewritten as a mixed-integer linear program (MILP) and solved using standard software.

We consider following system:

$$x_{i+1} = Ax_i + Bu_i, \qquad \forall i = 0,\ldots,N-1, \tag{47a}$$

$$y_i = Cx_i, \qquad \forall i = 0,\ldots,N, \tag{47b}$$

and also "reduced" model

$$x_{i+1}^{red} = A^{red} x_i^{red} + B^{red} u_i, \qquad \forall i = 0,\ldots,N-1, \tag{48a}$$

$$y_i^{red} = C^{red} x_i^{red}, \qquad \forall i = 0,\ldots,N. \tag{48b}$$

To reduce the model we use balanced truncation. We note that the map from the full state vector x to the balanced and truncated system is given by $z = T_l x$ [17].

MPC controller:

$$\min_{x,u} J(x,u) = \frac{1}{2} x'_N P x_N + \frac{1}{2} \sum_{i=0}^{N-1} x'_i Q x_i + u'_N R u_i \tag{49}$$

subject to

$$\begin{aligned}
& x_{i+1} = A x_i + B u_i, && \forall i = 0,..., N-1, \\
& x_i \in \mathbf{X}, && \forall i = 1,..., N-1, \\
& u_i \in \mathbf{U}, && \forall i = 0,..., N-1, \\
& u_{\min} \le u_i \le u_{\max} && \forall i = 0,..., N-1, \\
& x_0 = B_d d_0, \\
& d_0 = given
\end{aligned}$$

We use $x_i = A^i x_0 \sum_{j=0}^{i-1} A^j B u_{i-1-j}$ to rewrite our MPC problem (49) to this form [9]:

$$V(x_0, d_0) = \frac{1}{2} x'_0 Y x_0 + \min_u \left\{ \frac{1}{2} U' H U + x'_0 F_x U + d'_0 F_d U \right\}, \tag{50a}$$

subject to $GU \le W + E_x x_0 + E_d d_0$ (50b)

By using a modification of (50a) and (50b) we can define our lower-level problem in bilevel programming and the KKT conditions (51). For this problem can be defined as [17]:

$$\begin{aligned}
HU + F_x x_0 + G'\lambda &= 0 \\
GU - W - E_x x_0 &\le 0 \\
\lambda &\ge 0 \\
\lambda &\le Ms \\
GU - W - E_x x_0 - E_d d_0 &\ge M(1-s)
\end{aligned} \tag{51}$$

In (49) we use binary variables $s \in \{0,1\}^{n_W}$ and big-M formulation, where n_W is the number of inequality constraints in (50b). The big-M is a constant that is large enough such the solution to (51) corresponds to the solution of (50).

The same method we will use to get MPC controller using reduced model. Defined KKT conditions (51) and our MPC problem (50) we will use as a lower-level problem in bilevel programming. Using full-order model (47) we get matrices $(H^{full}, F^{full}, G^{full}, W^{full}, E^{full})$ from KKT conditions (51) and using low-order model (48) we get matrices $(H^{red}, F^{red}, G^{red}, W^{red}, E^{red})$ from KKT conditions (51) [17].

Here in (52) and (53) we define the one-step problem as:

$$\max_{d_0 \in D} \left\| Q_y (\mathrm{y}^{\mathrm{full}} - \mathrm{y}^{\mathrm{red}}) \right\|_\infty \tag{52}$$

subject to

$$\begin{aligned} &\mathrm{KKT}(\mathrm{MPC}^{\mathrm{full}}, x) \\ &\mathrm{KKT}(\mathrm{MPC}^{\mathrm{red}}, T_l x) \end{aligned} \tag{53}$$

Using objective function (52) we define our one-step problem which means that we are looking for maximal difference of outputs $\mathrm{y}^{\mathrm{full}}$ from full and outputs $\mathrm{y}^{\mathrm{red}}$ from reduced model. A part of the constraints are KKT conditions, prediction models for full-order and reduced-order model, equality constraints $x_0^{full} = B_d d_0, x_0^{red} = B_d d_0$ and d_0 is in D interval, where

$$D = \left\{ d \in R^2 \mid \|d\|_\infty \leq 1 \right\}. \tag{54}$$

Equations (52), (53) represents only one-step problem. We would like to calculate the worst-case error (WCE) over some steps using simulation time or number of simulation steps N_{sim}. Objective function for our bilevel problem can be defined (55). Using simulation steps N_{sim} we get more KKT conditions (56), one for each simulation step N_{sim}.

$$\max_{d_0 \in D} \left\| Q_y \left(\mathrm{y}_{\mathrm{Nsim}}^{\mathrm{full}} - \mathrm{y}_{\mathrm{Nsim}}^{\mathrm{red}} \right) \right\|_\infty \tag{55}$$

subject to

We consider that $x_0^{full} = B_d d_0$, $x_0^{red} = B_d d_0$.

$$\text{for} \qquad \mathrm{i} = 1 : \mathrm{N}_{\mathrm{sim}} \tag{56}$$

$$\mathrm{x}_{\mathrm{i}}^{\mathrm{full}} = A x_{i-1}^{full} + B u_{MPC\,x(i-1)}^{full}$$
$$\mathrm{x}_{\mathrm{i}}^{\mathrm{red}} = A x_{i-1}^{red} + B u_{MPC\,x(i-1)}^{red}$$

$$\mathrm{u}_{\mathrm{MPC}\,x(i-1)}^{full} = \mathrm{KKT}\left(\mathrm{MPC}^{\mathrm{full}}, x_{i-1}^{full} \right)$$
$$\mathrm{u}_{\mathrm{MPC}\,x(i-1)}^{red} = \mathrm{KKT}\left(\mathrm{MPC}^{\mathrm{red}}, T_l x_{i-1}^{red} \right)$$

end

We wanted originally to calculate the WCE as a sum of the worst-case outputs differences between full and reduced order MPC controllers during some simulation steps N_{sim} (57).

$$\sum_{i=1}^{N_{sim}} \left\| \mathrm{y}_{\mathrm{i}}^{\mathrm{full}} - \mathrm{y}_{\mathrm{i}}^{\mathrm{red}} \right\|_\infty \tag{57}$$

However it was not possible because this optimization problem is really difficult to solve, respectively solving this problem takes very long time. Finally, we decided to try one-step formulation of objective function (58), where the WCE is calculated in the last simulation step and these results of worst-case disturbances are used to obtain the initial states for closed loop MPC simulation. Using simulation steps N_{sim} in this MPC simulation we will get a sum of the worst-case outputs for concrete worst-case disturbances. We assume there is some difference between real calculations of (57) and using this method, because we are solving different optimization problem, but as is shown later the difference is small and this solution is usable.

$$\left\| \mathrm{y}_{N_{sim}}^{\mathrm{full}} - \mathrm{y}_{N_{sim}}^{\mathrm{red}} \right\|_\infty \tag{58}$$

4.1.1 Simulations

We here present some simulations and plots. These simulations can be divided into several groups:

- in these plots we would like to show how the WCE of outputs is changing when we use different reduced models with different number of states,
- we would like to compare closed loop simulation with number of simulation steps $N_{sim} = 10$ for the full order controller $(n_{full} = 16)$ and low order controller $(n_{red} = 4)$,
- in other plots we would like to show the trend of changing WCE of outputs calculating with objective function in the last step of simulation time N_{sim} while the simulation time is changing from $N_{sim} = 1$ to $N_{sim} = 20$,
- using MPC simulations we would like to calculate and compare the sum of WCE obtained over the simulation time $N_{sim} = 20$, for this simulations we will use the worst-case disturbances from solutions of bilevel problems which are using objective function calculating for the last step of simulation time changing from $N_{sim} = 1$ to $N_{sim} = 20$,
- also we would like to check that we can use MPC simulations to calculate the WCE, when we have worst-case disturbances,
- for one example we would like to compare sum of WCE obtained using updating objective function (12) in our bilevel problem to sum of WCE obtained with calculation of one-step problem and then using worst-case initial disturbances in MPC simulations,
- we would like to show how the worst possible initial disturbances (d_{01}, d_{02}) which we used to calculate initial states $x_0 = B_d d_0$ were changing.

4.1.1.1 WCE for a Set of Different Reduced-order Models

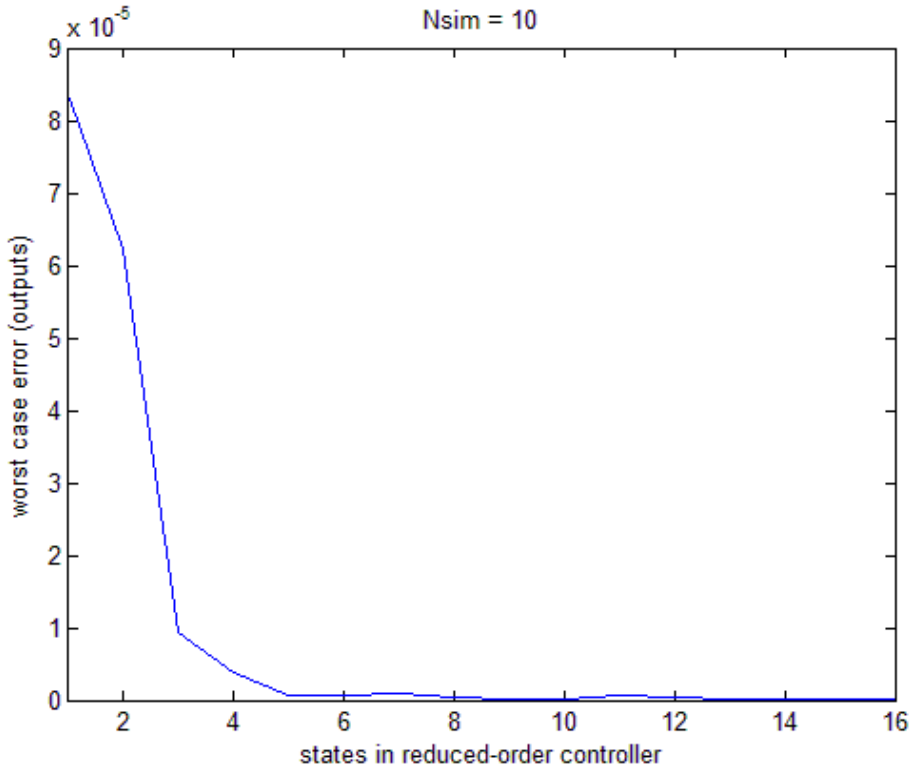


Figure 12: WCE for a set of different reduced order models

In figure 12 is plot of the WCE for a set of different reduced order models, which consist of 1 to 15 states. The WCE is the most different between full-order outputs and reduced-order outputs which we get from equations (2b), (3b), respectively. Number of simulation steps in every calculation WCE for different reduced models is $N_{sim} = 10$. The objective functions which we use to obtain the WCE of outputs use only outputs from last simulation step. This means that the objective function is calculated only in the last simulation step when $N_{sim} = 10$. We can see that the WCE of outputs is decreasing with rising numbers of states in reduced order controller. It was expected because when we use less reduced model also the different between full order model and reduced order model is smaller.

4.1.1.2 Closed Loop Simulation

In figure 13 we can see difference when we compare closed loop simulation for the full-order controller $(n_{full} = 16)$ and low order controller $(n_{red} = 4)$ during simulation steps which number is $N_{sim} = 10$. Reason why full order controller produces other inputs into controlling system as low order controller is very simple. It is because low order controller uses reduced model to calculate prediction as full order controller which use no reduced model of system. There is also a difference between the full-order outputs and the reduced-order outputs, but the difference is very small and it is not clear see in figure 13.

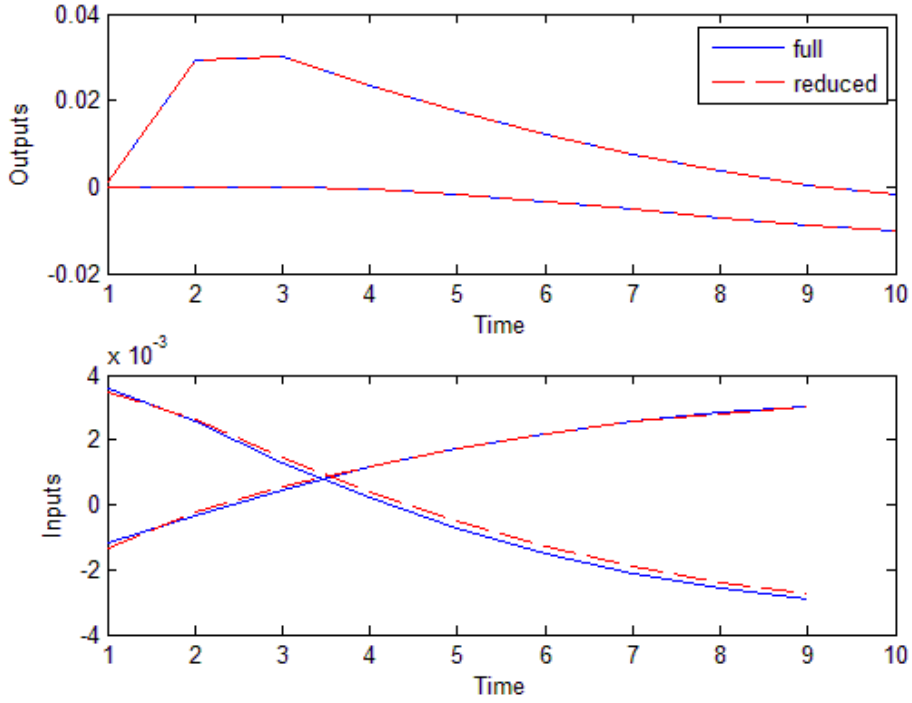


Figure 13: Closed loop simulation for the full order controller $(n_{full} = 16)$ and low order controller $(n_{red} = 4)$ and with number of simulation steps $N_{sim} = 10$

4.1.1.3 WCE for a Set of Different Reduced-order Models with Changing N_{sim}

The figures 14, 15 answer us the question how is changing the dependence of WCE on different reduced-order controllers when we use different simulation times or number of simulation steps which are changing from 1 to 20. The objective function which is used to calculate the worst case difference of outputs is calculated in the last step of changing simulation time. The figure 15 compares dependence WCE of outputs on different reduced-order controllers which was obtained using different simulation times namely $N_{sim} = (1,4,8,12,16,20)$. It is good see that with increasing simulation time N_{sim} is this WCE dependence decreasing especially when we use reduced-order controllers which are using reduced model with number of states in range 1 to 6. Exception is the WCE of outputs when is using simulation time $N_{sim} = 1$. Reason of this is very simple, because these outputs are calculated from the initial states which are for full-order and reduced-order controllers same.

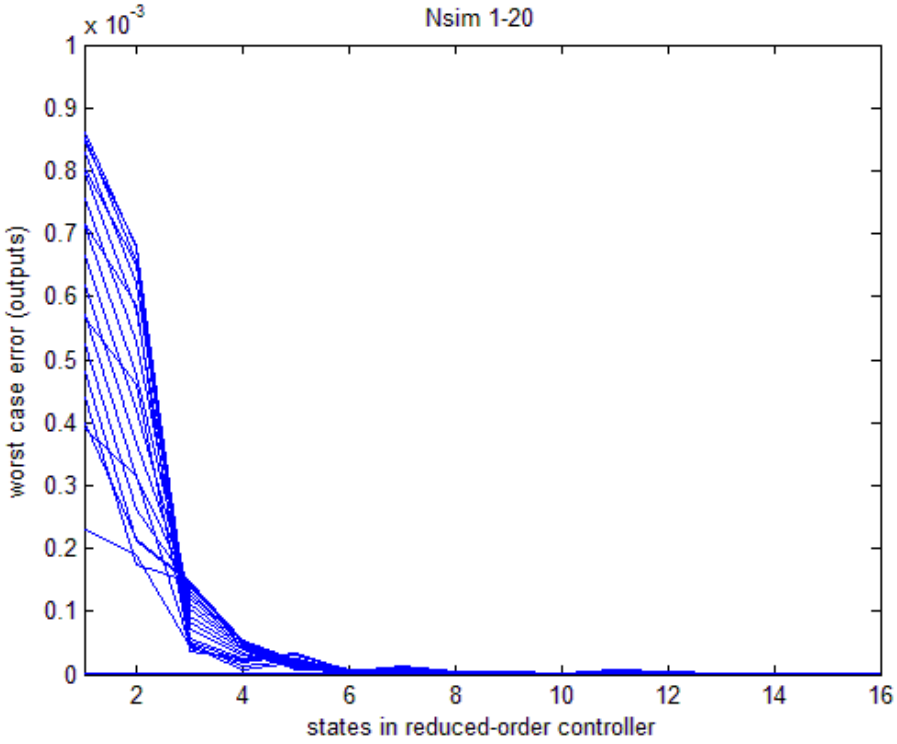


Figure 14: WCE for a set of different reduced order models with changing $N_{sim} = 1-20$

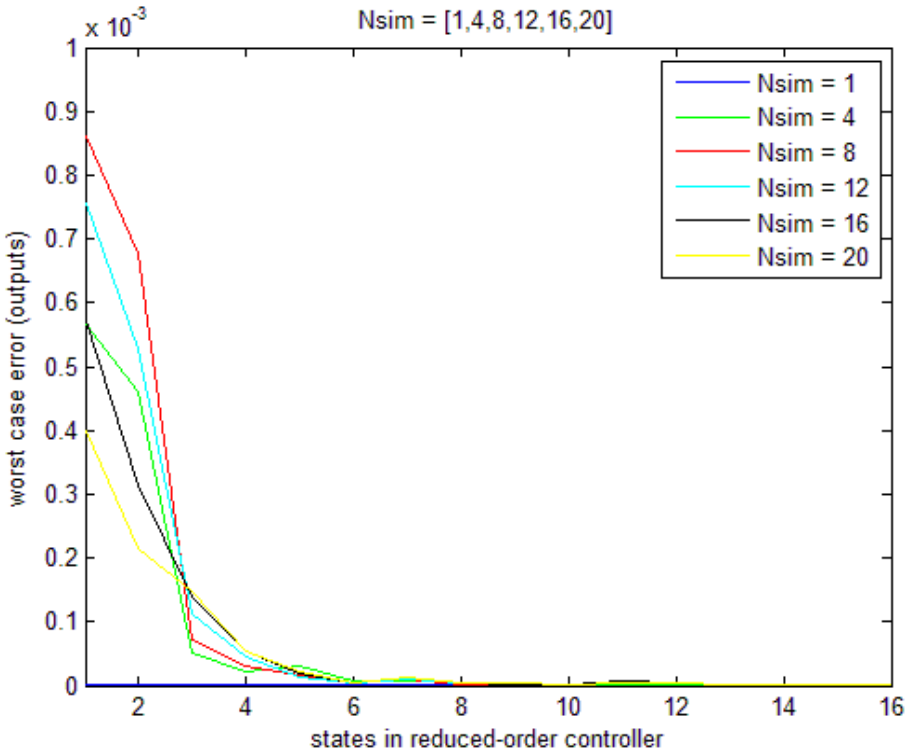


Figure 15: WCE for a set of different reduced order models with changing

$$N_{sim} = (1,4,8,12,16,20)$$

4.1.1.4 The Worst Possible Initial Disturbances

In figure 16 we can see worst possible initial disturbances for reduced-order controller using reduced model with number of states n_{red} = 10 and different time of simulation $N_{sim} = (1,4,8,12,16,20)$. As it was written these initial disturbances are used to calculate the initial states $x_0 = B_d d_0$ which are the worst possible initial states. Values of initial disturbances which are plotted in the figure 16 are also in table 3 in row number 10 for number of states n_{red} = 10 and we can see how are changing from interval $-1 \le d_{0i} \le 1,\ i = 1,2$.

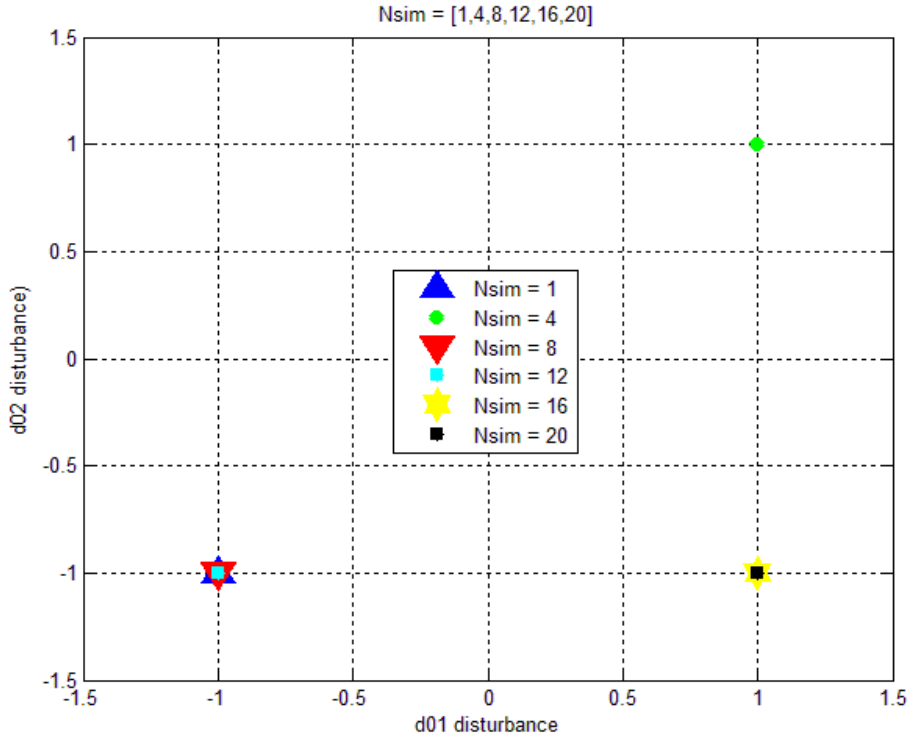


Figure 16: Using initial disturbances d_{01}, d_{02} for calculating WCE for reduced model n_{red} = 10 and with changing $N_{sim} = (1,4,8,12,16,20)$

d01						
States /Nsim	1	4	8	12	16	20
1	1	-1	-1	-1	-1	-1
2	1	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1	-1
4	-1	1	1	1	1	1
5	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1
7	-1	1	1	-1	-1	-1
8	1	-1	-1	-1	-1	-1
9	1	-1	-1	-1	-1	-1
10	-1	1	-1	-1	1	1
11	-1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1
14	1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1

d02						
States/Nsim	1	4	8	12	16	20
1	1	1	1	1	1	1
2	1	1	1	1	1	-1
3	-1	1	-1	-1	-1	-1
4	-1	1	-1	-1	-1	-1
5	-1	1	1	-1	-1	-1
6	-1	1	-1	-1	-1	-1
7	1	1	1	1	1	1
8	1	1	-1	-1	-1	-1
9	1	-1	-1	-1	1	1
10	-1	1	-1	-1	-1	-1
11	1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	1	1	1
14	1	-1	-1	-1	-1	-1
15	-1	-1	-1	1	1	1

Table 3: Initial disturbances

4.1.1.5 WCE sum for a Set of Different Reduced-order Models with Changing N_{sim}

As we said before, we wanted originally to calculate the WCE as a sum of the worst-case outputs differences between reference and reduced MPC controllers (57), because solving this optimization problem takes very long time, we decided to try other formulation of objective function (58), where the WCE is calculating in the last simulation step and calculated results of worst-case disturbances are used to obtain the initial states for closed loop MPC simulation.

It is clear that when we use the worst case initial disturbances calculated for different reduced models and simulation steps as seen in table 1 we will get in some cases the same initial states for closed loop MPC simulation. This also mean that the sum of WCE will be for that simulations same. This argument proves figure 17, where lines represented sums of WCE obtained using different number of simulation steps are identical. This result is distinct also in figure 18 for concrete simulation steps.

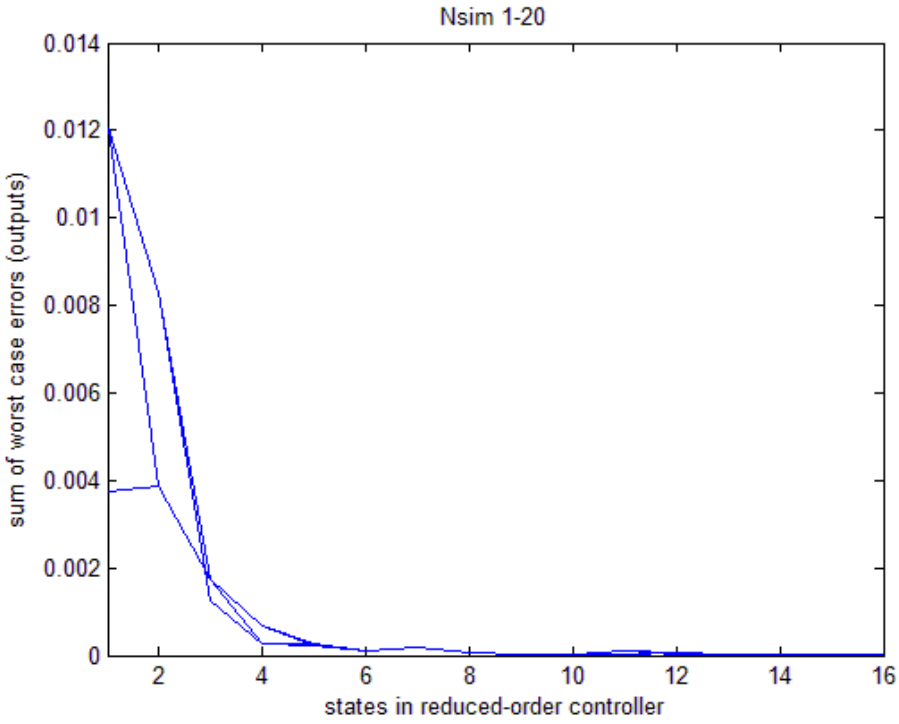


Figure 17: Sum of WCE for a set of different reduced order models with changing $N_{sim} = 1-20$

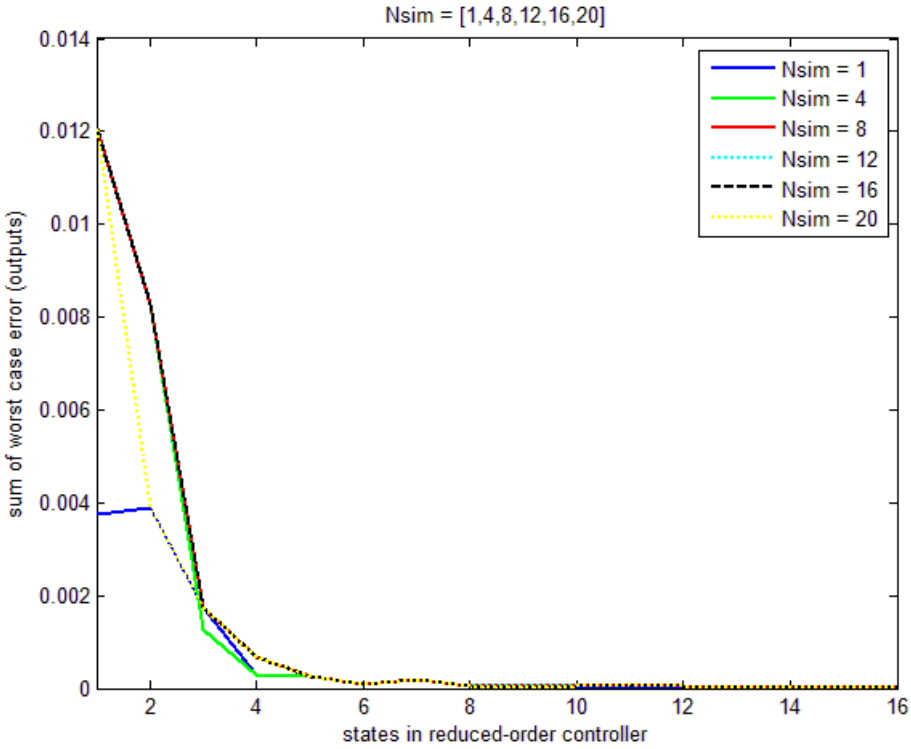


Figure 18: Sum of WCE for a set of different reduced order models with changing $N_{sim} = (1,4,8,12,16,20)$

4.1.1.6 Comparison of WCE Sum Using Real Updating Objective Function and MPC Simulation Calculation

It is important to show the difference when the sum of WCE error is calculated with updating objective function or with our simplified technique which calculates WCE in the last step of simulation and then are using obtained disturbances in MPC simulation. In figures 19, 20, 21 are displayed plots when the objective function (58) is calculated in the last simulation step while numbers of simulation steps $N_{sim} = (1,2,3,4,5,6,7,8,9,10)$ are changing. These measurements are compared with sum of WCE obtained with updating objective function (57). From these plots we can see that the difference between real and simplified technique is very small. And also is shown that calculating objective function in the last step is the best. Reason why we get different WCE is very simple. It is because we are solving different optimization problems.

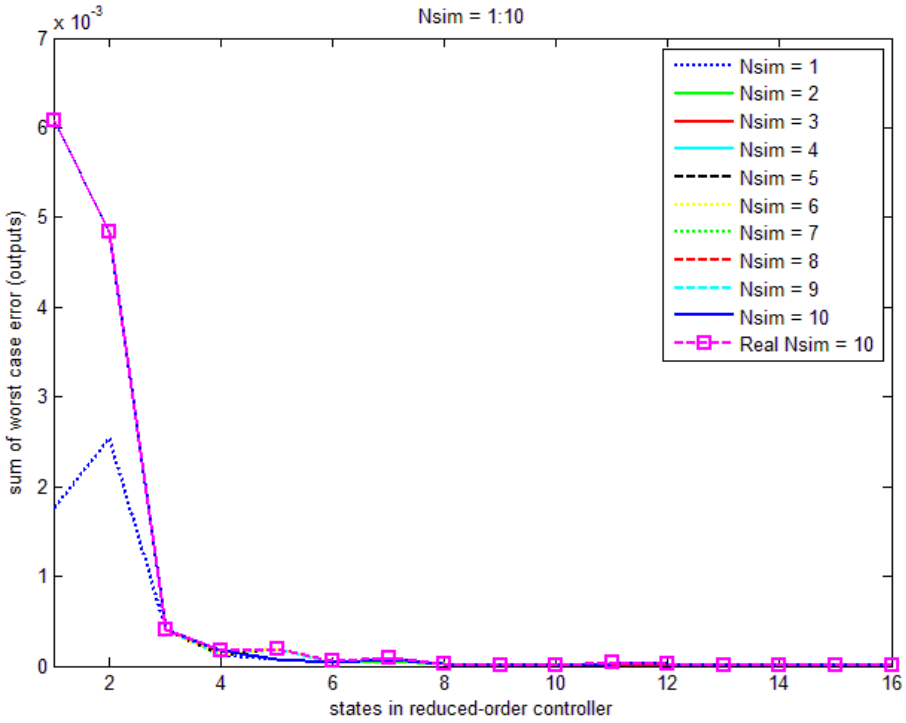


Figure 19: Comparison real sum of WCE (11) and sum of WCE (12) obtain from disturbances calculated in the last simulation step

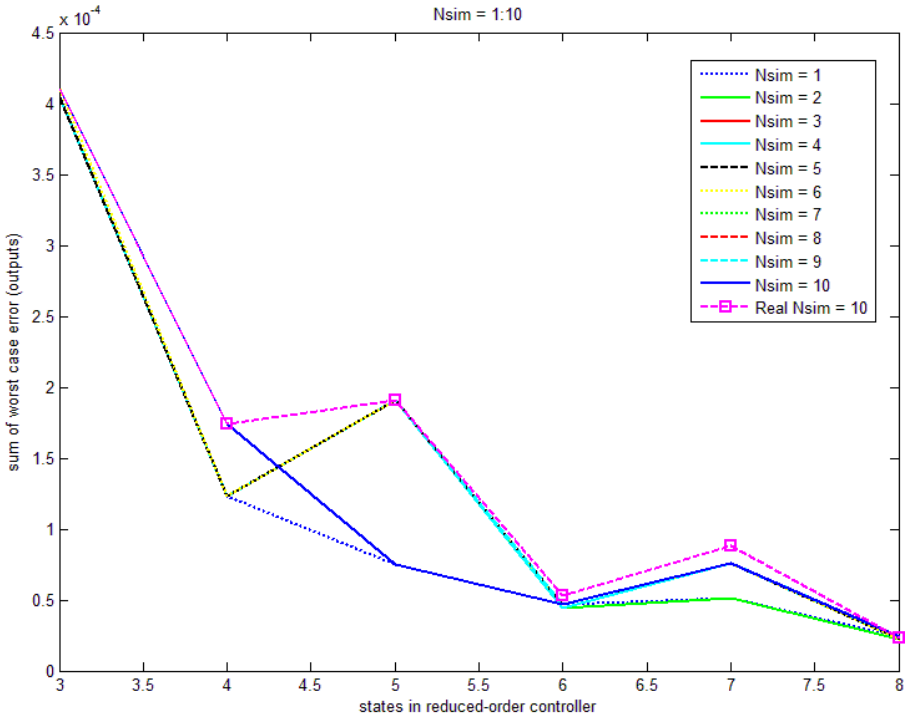


Figure 20: Zoom 1 of figure 8

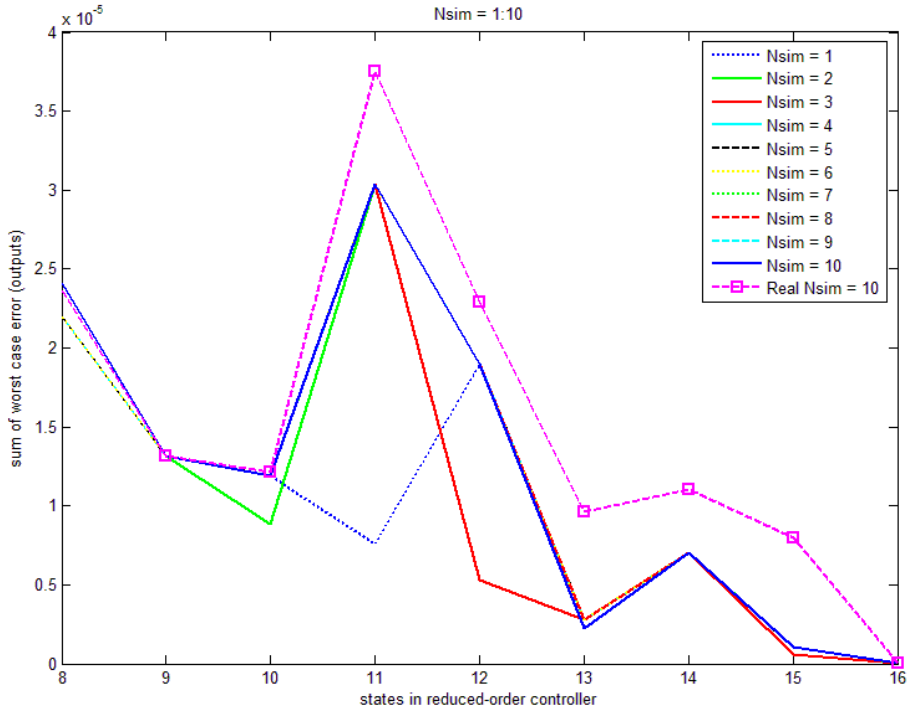


Figure 21: Zoom 2 of figure 8

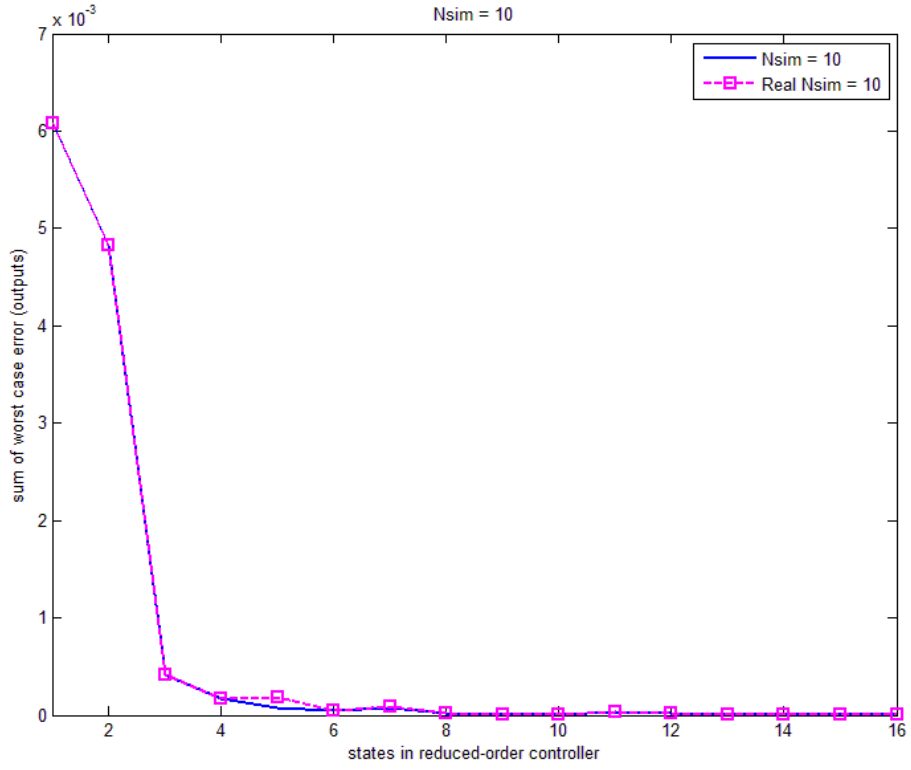


Figure 22: Comparison real sum of WCE (11) and sum of WCE (12) obtain from disturbances calculated in the last simulation step $N_{sim} = 10$

4.1.1.7 Check of the WCE Using Closed Loop Simulation

In this part we would like to check if our calculated WCE for outputs is the same than we can get using closed loop simulation. We chose simulation length $N_{sim} = 18$ and different reduced models with number of states from interval 1 to 15. For these conditions we calculated the maximum difference of outputs in the last simulation step. As initial conditions we were using calculated worst case initial states, respectively the worst initial disturbances which were inserted into equation (3b). In the figure 23 we see that the worst-case errors are the same and the difference between these worst-case errors is zero and we can say that the worst-case error which we got from solving the bilevel problem is correct.

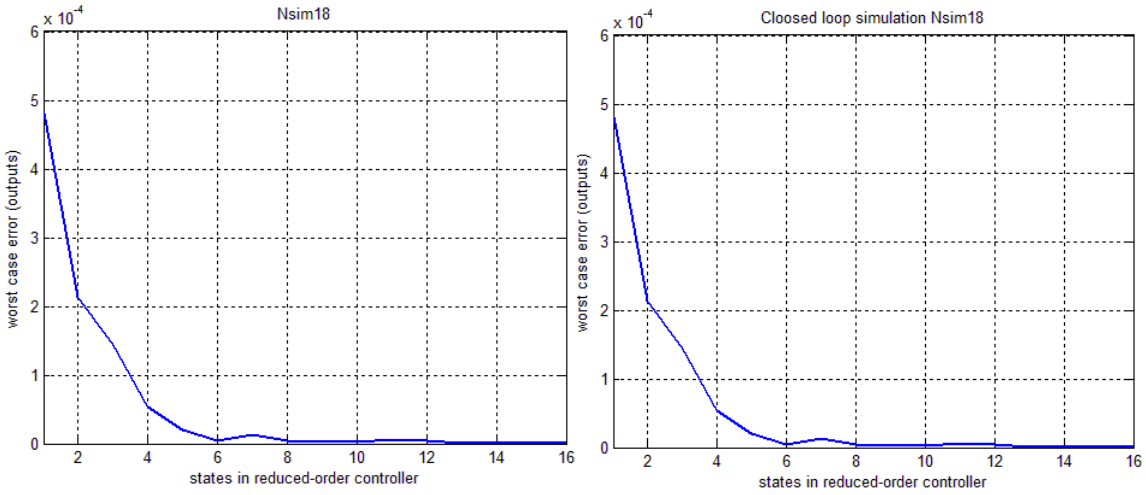


Figure 23: Compare worst-case errors for a set of different reduced order models using $N_{sim} = 18$ reached as a solution of bilevel problem and it closed loop check

4.1.2 Conclusions

In this part of project we obtained the worst-case difference between an MPC using the full model (2) and an MPC using the reduced model (3) with number of states from range 1 to 15 and with different length of simulation time. Solving the bilevel problem we get also the worst-case initial disturbances which we used to check the maximum difference between obtained outputs using full-order and reduce-order MPC controller. We investigated a possibility to use simplified method based on calculating the sum of WCE from MPC simulation which use worst-case disturbances obtained from solving bilevel problem with objective function calculated only in the last simulation step. These results we figured at plots using some simulations.

4.2 Move Blocking Worst-case Error Analysis

In this capture we would like to answer these questions: *What is the worst-case difference between an MPC without using* ***move blocking*** *and an MPC using* ***move blocking*** *which we use to make MPC faster? Another question is, which move blocking type gives us less worst-case error, when we compare different types of move blocking?* As it was written in (MPC theory chapter) move blocking is a method to simplify the complexity of MPC problem, where we can reduce the degrees of freedom using move blocking approach. Principle of *move blocking* it is fixing the input (*input blocking*) or its derivatives (*delta input blocking*) to be constant over several time-steps in calculation of optimal inputs [8]. To get an input blocking (IB) and delta input blocking (DIB) matrices we created functions ***make_blocking*** and ***make_delta_blocking***.

To find the maximum difference between the controller without *move blocking* and the simplified controller using *move blocking* we use bilevel programming. We could calculate the distance between the controllers as $\left\| u_k - u_k^{red} \right\|_\infty$, but we focused on difference in outputs $\left\| Q_y (y_k - y_k^{red}) \right\|_\infty$ as before when we used model reduction to make MPC faster. We use the infinity norm because then the problem can be reformulated as a mixed-integer linear program (MILP) [2].

$$\begin{aligned} & \max_{d \in D} \ \text{distance}(\mathrm{y_k}, \mathrm{y_k^{red}}) \\ \text{subject to} \quad & \mathrm{y_k} = \arg\min \{\mathrm{MPC\ without\ using\ move\ blocking}\} \\ & y_k^{red} = \arg\min \{\mathrm{MPC\ with\ using\ move\ blocking}\} \end{aligned} \tag{59}$$

Formulation and solving of this problem is the same like it was shown in chapter 4.1.

4.2.1 Simulations

In these simulations we would like to show and compare some results concerning to simplify technique - *move blocking*. These simulations can be divided into two groups. First group of simulations is dedicated to compare IB types and show the trend how the worst-case error (WCE) is changing when we use different IB type with different reduction of degree of freedom (DOF). Second group of simulation is about DIB and we here also compare the trend of using different DIB types and what effect it has on WCE. Finally we compare which of these simplify technique is more effective to use. In other words, which of these methods give us less WCE when we reduce degree of freedom at the same value?

4.2.1.1 Input Blocking

Function **make_blocking** allows us to generate different IB types and enables us compare IB with fixing different number of inputs to be constant over a certain number of steps too. Fixing of inputs allows us to change DOF.

We choose these conditions for following simulations. Prediction horizon with length $N = 8$, number of states in mathematical model $n = 16$, simulation time $Nsim = 10$. WCE was calculated in the last step in the simulation time as in (***Model Reduction chapter***). Then we used worst-case disturbances and IB type in MPC simulation to calculate sum of WCE over simulation time $Nsim = 10$. Presented values of WCE are WCE sum obtained from MPC simulations.

In figure 24, 25 we can see that with decreasing DOF the WCE is increasing. This trend of using different IB type was expected, because the more we simplify MPC, we should get an increase in WCE too. For calculation sum of WCE were used worst-case disturbances found by the bilevel program (table 4).

	[8]	[4 4]	[2 3 3]	[1 2 2 3]	[1 1 1 2 3]	[1 1 1 1 1 3]	[1 1 1 1 1 1 2]
d_1	-1	-1	-1	1	1	1	-1
d_2	-1	-1	-1	1	1	1	-1

Table 4: Worst-case disturbances used to calculate x_1 in the MPC simulation

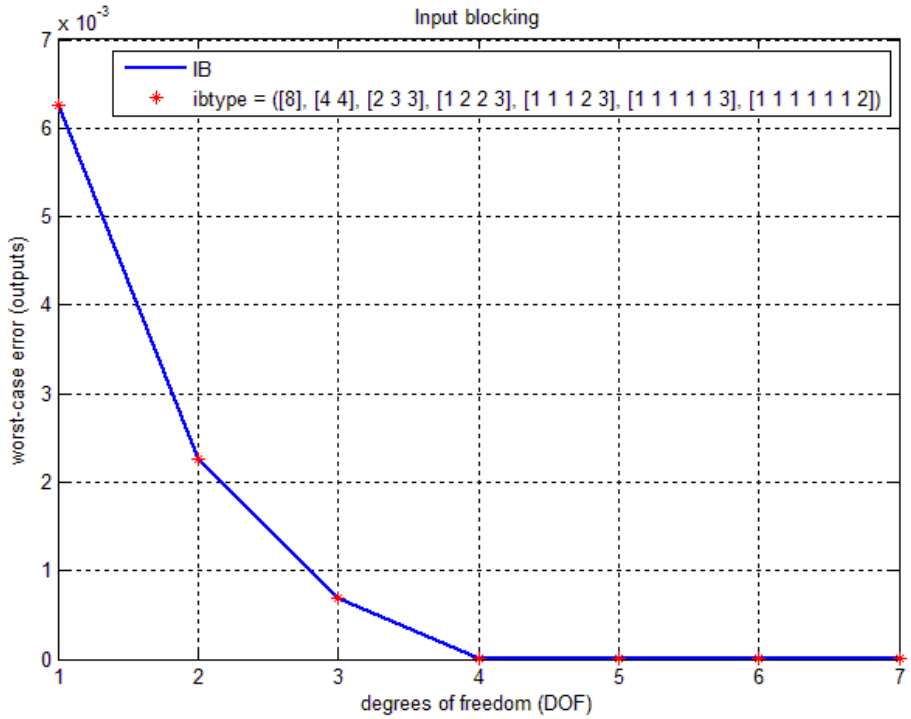


Figure 24: WCE for a set of DOF using different IB

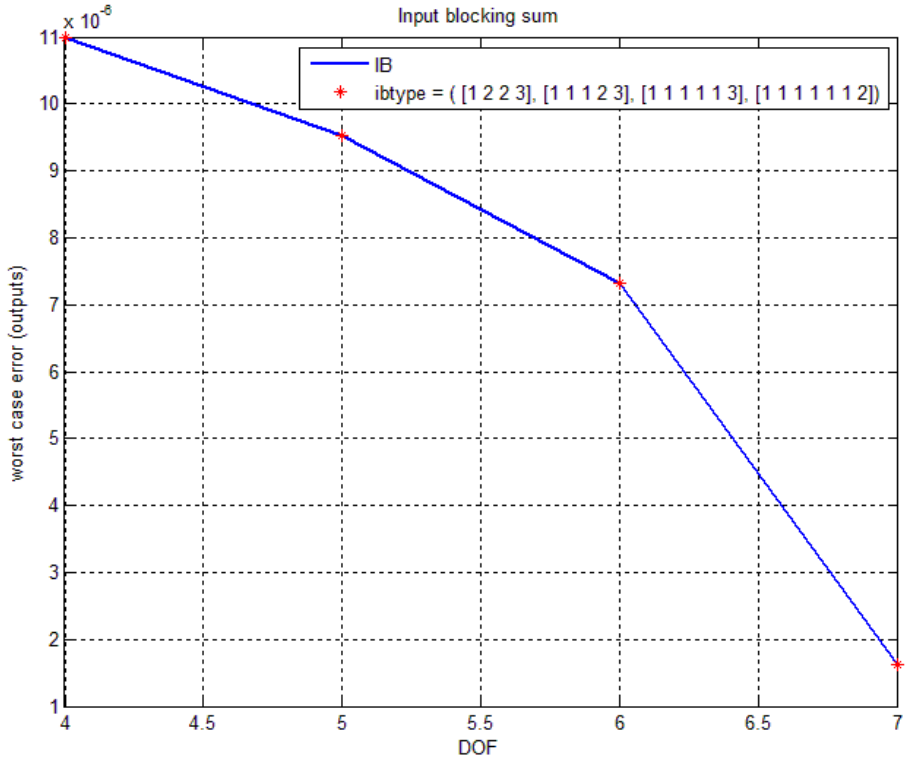


Figure 25: WCE for a set of DOF using different IB Zoom

In figures 26 and 27 we show a predicted inputs calculated for first simulation step. Number of predicted inputs is equal to length of prediction horizon multiplied to number of inputs. In this case for our example of distillation column model we have two inputs u_1 and u_2 and prediction horizon has length $N = 8$. In figure 26 we present IB type which fix four and four inputs to be constant and reduce DOF from eight to two. Another IB type is presented in figure 27, where first inputs is free and then two, two and three inputs are fixed to be constant. Using this IB type we reduced DOF from eight to four.

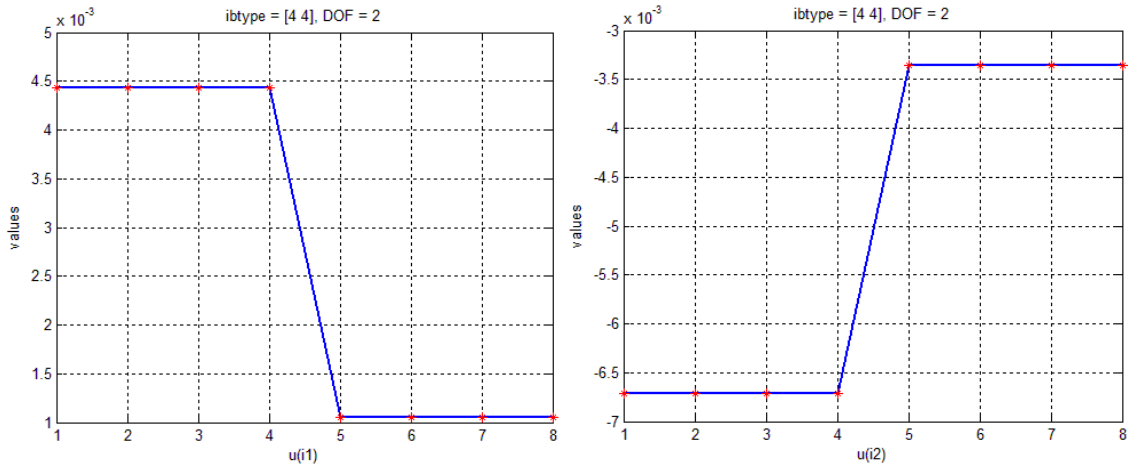


Figure 26: Predicted inputs with IB type = [4 4] and DOF = 2

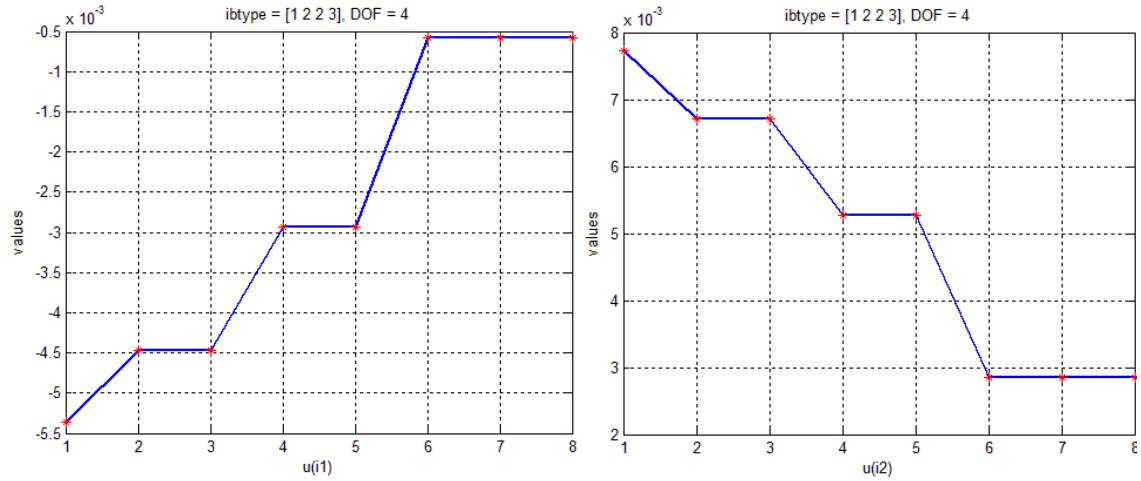


Figure 27: Predicted inputs with IB type = [1 2 2 3] and DOF = 4

We would like to investigate also the case when we are using different IB type and we reduce MPC on the same DOF. If IB type contains also some free inputs then is relevant question if is better to use free inputs at the beginning or in the end of IB? Answer to this question gives us figures 28, 29 and tables 5, 6. From these plots and from values of

WCE we can deduce that using free inputs at the beginning of predicted inputs it is better than in the end. Values of the WCE were obtained from the MPC simulation with worst-case disturbances with values -1.

WCE obtained using free inputs at the beginning of IB:

IB type:		WCE:
• [1 1 6]	$\Rightarrow$	$0.0737 \cdot 10^{-3}$
• [1 2 5]	$\Rightarrow$	$0.0440 \cdot 10^{-3}$
• [1 3 4]	$\Rightarrow$	$0.0270 \cdot 10^{-3}$
• [2 2 4]	$\Rightarrow$	$0.6799 \cdot 10^{-3}$
• [2 3 3]	$\Rightarrow$	$0.6873 \cdot 10^{-3}$

Table 5: Free inputs at the beginning of IB

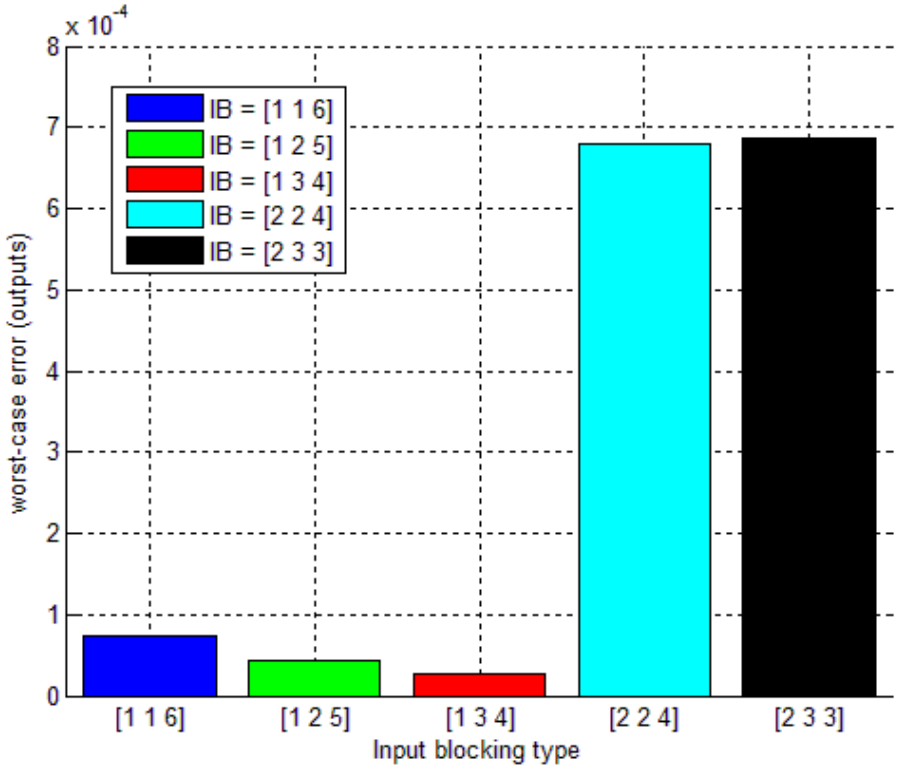


Figure 28: IB types for same degree of freedom – free inputs at the beginning

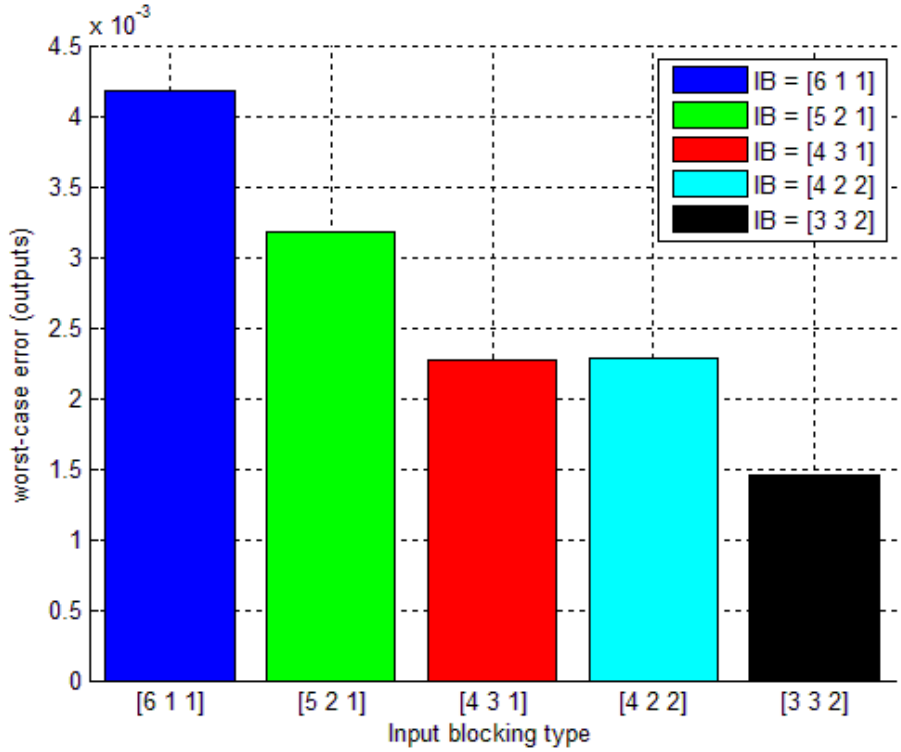


Figure 29: IB types for same degree of freedom – free inputs in the end

WCE obtained using free inputs in the end of IB:

IB type:		WCE:
• [6 1 1]	⇒	0.0042
• [5 2 1]	⇒	0.0032
• [4 3 1]	⇒	0.0023
• [4 2 2]	⇒	0.0023
• [3 3 2]	⇒	0.0015

Table 6: Free inputs in the end of IB

In table 6 we can see that using smaller number of fixed inputs in the end of IB is better than using smaller number of fixed inputs in the middle of IB. This is probably due to "end - effects" in MPC problem.

4.2.1.2 Delta Input Blocking

Similarly as before but now it is function **make_delta_blocking** which allows us to generate different DIB types with different number of fixed differences between two consecutive control inputs over several steps, what allows us to reduce DOF. Here we choose the same conditions as for IB. Prediction horizon N = 8, number of states n = 16, simulation time $Nsim$ = 10 and we calculate WCE sum from MPC simulation using these conditions and worst-case disturbances obtained from bilevel program.

In figures 30, 31 we can see the trend of increasing of WCE with decreasing DOF. Comparing plots 7 and 9 we can see that using free delta inputs in the beginning gives us much better results (less WCE) as using free delta inputs in the end of IB.

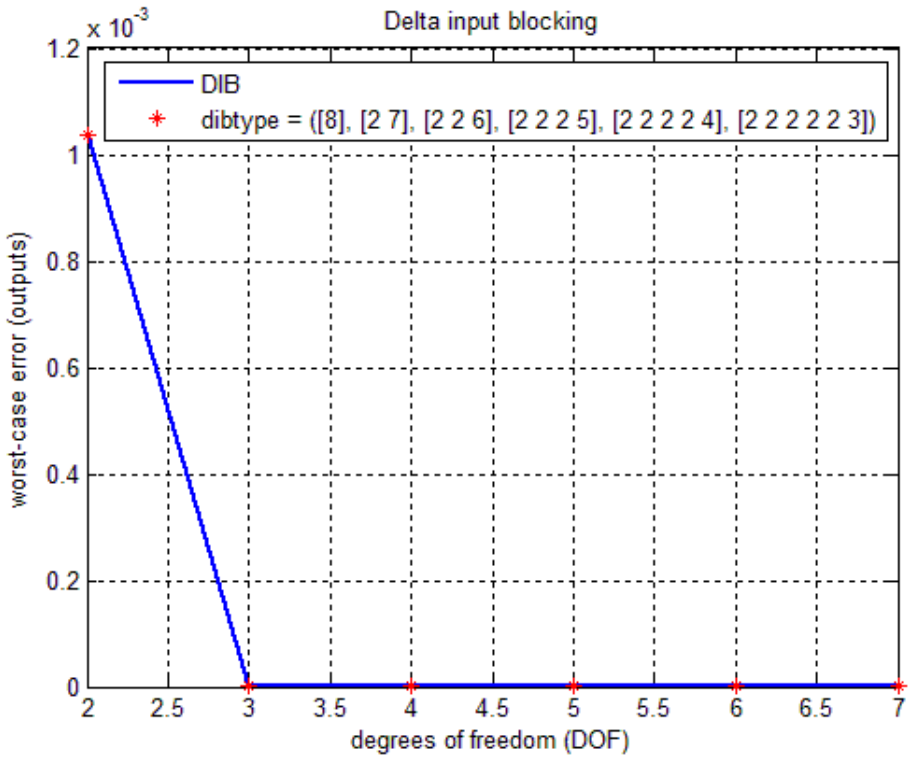


Figure 30: WCE for a set of DOF using different DIB

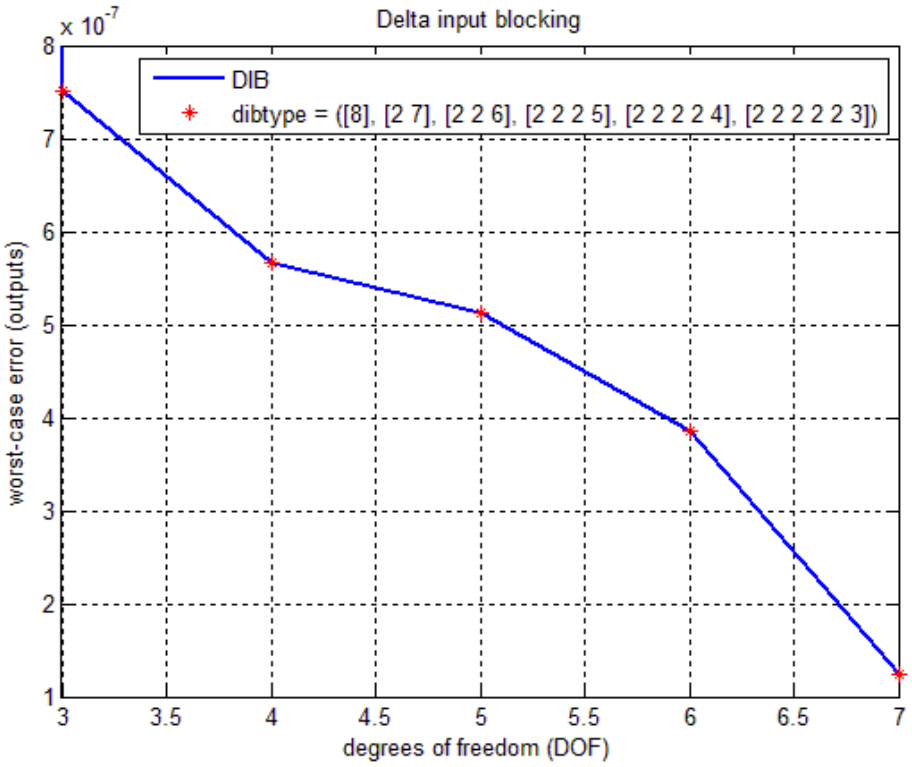


Figure 31: WCE for a set of DOF using different DIB Zoom

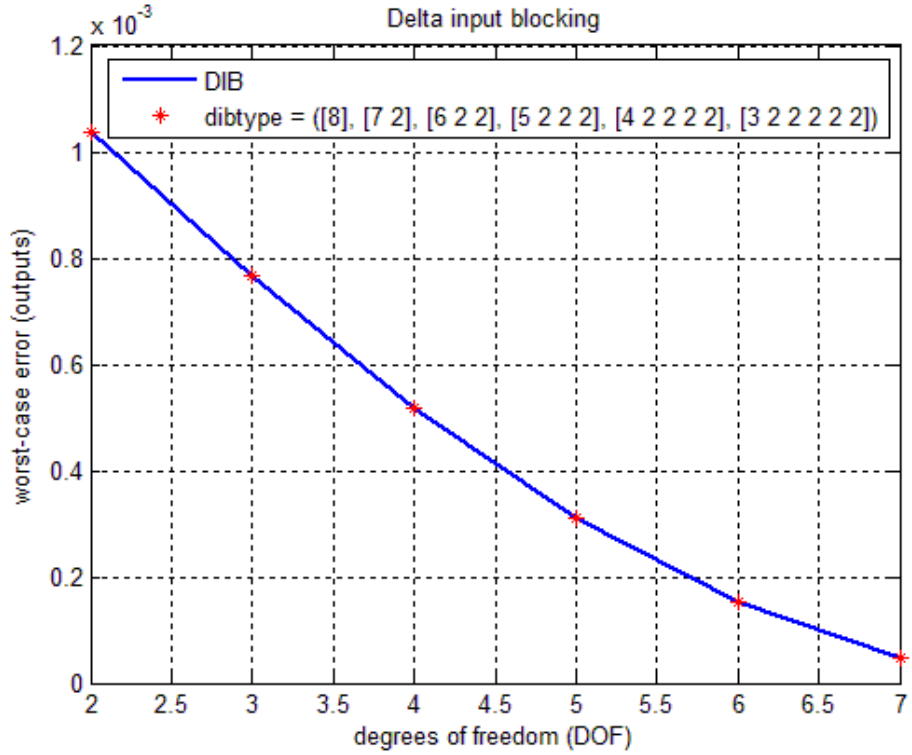


Figure 32: WCE for a set of DOF using different DIB free inputs in the end

To calculate WCE sum for DIB with free inputs in the end, we used worst-case disturbances found by our bilevel program presented in table 7. For DIB with inputs at beginning were used worst-case disturbances with values 1.

	[8]	[2 7]	[2 2 6]	[2 2 2 5]	[2 2 2 2 4]	[2 2 2 2 2 3]
d_1	-1	1	1	1	1	1
d_2	1	1	1	1	1	1

Table 7: Worst-case disturbances used to calculate x_1 in the MPC simulation

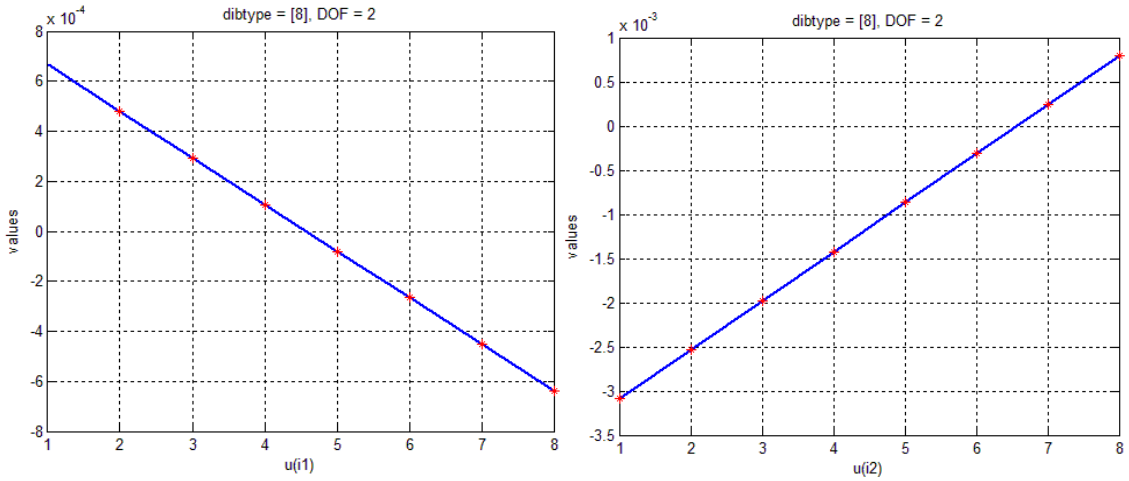


Figure 33: Predicted inputs with DIB type = [8] and DOF = 2

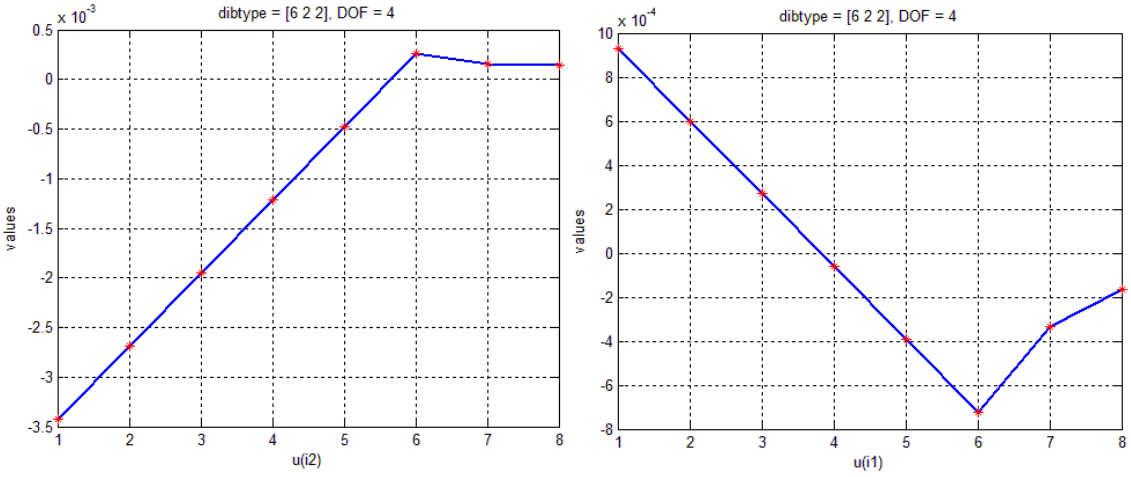


Figure 34: Predicted inputs with DIB type = [6 2 2] and DOF = 4

In figures 33 and 34 we show a predicted inputs calculated for first simulation step. In figure 33 we present DIB type which fix differences between consecutive control inputs and reduce DOF from eight to two. DIB type with fixing first five differences between

inputs and last two differences are free is presented in figure 34.Using this DIB type we reduced DOF from eight to four.

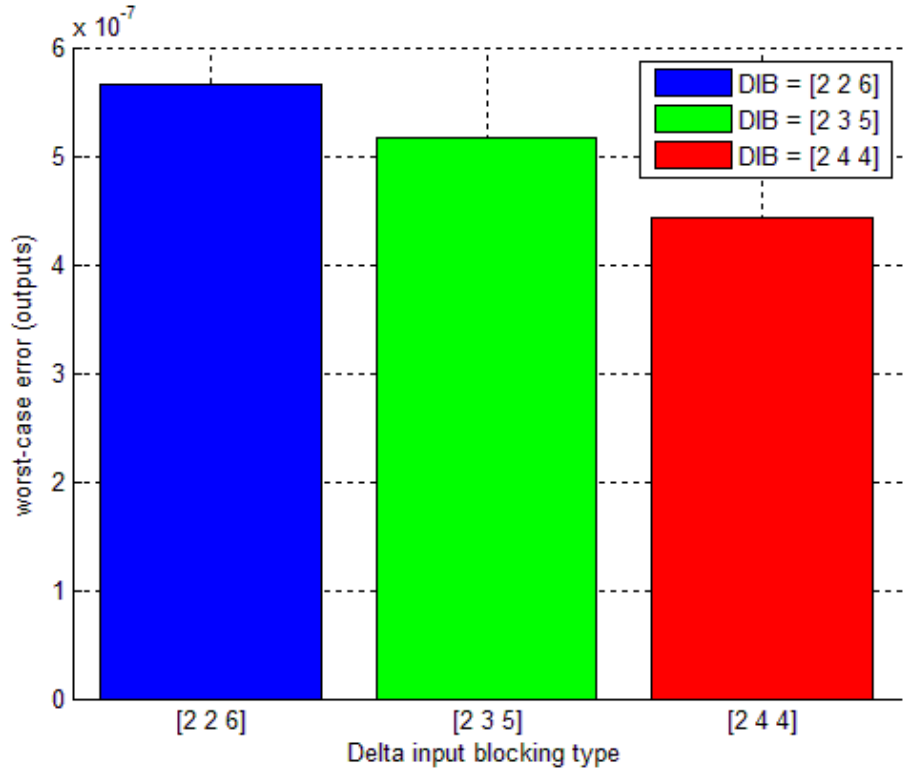


Figure 35: DIB types for same degree of freedom – first free

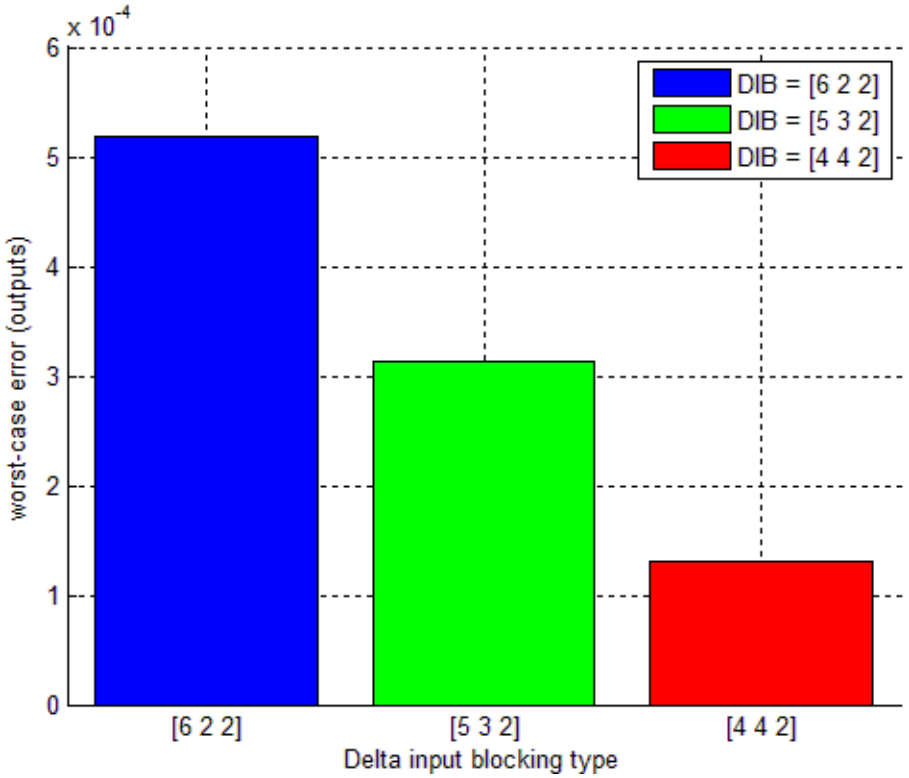


Figure 36: DIB types for same degree of freedom – last free

When we are using different DIB type and we reduce MPC on the same DOF we get different WCE. In figures 35, 36 and in table 8 we can see that using free delta inputs at the beginning of predicted inputs it is better than in the end. Values of WCE were obtained from MPC simulation with worst-case disturbances table 9.

DIB type:		WCE:
• [2 2 6]	$\Rightarrow$	$0.5665 \cdot 10^{-6}$
• [2 3 5]	$\Rightarrow$	$0.5164 \cdot 10^{-6}$
• [2 4 4]	$\Rightarrow$	$0.4429 \cdot 10^{-6}$
• [6 2 2]	$\Rightarrow$	$0.5193 \cdot 10^{-3}$
• [5 3 2]	$\Rightarrow$	$0.3128 \cdot 10^{-3}$
• [4 4 2]	$\Rightarrow$	$0.1313 \cdot 10^{-3}$

Table 8: Free inputs of DIB

	[2 2 6]	[2 3 5]	[2 4 4]	[6 2 2]	[5 3 2]	[4 4 2]
d_1	1	-1	1	-1	-1	-1
d_2	1	-1	1	1	1	1

Table 9: Worst-case disturbances used to calculate x_1 in the MPC simulation

4.2.1.3 Comparison of Input Blocking and Delta Input Blocking

The following figures and also previous results prove that reduction of DOF with DIB gives us less control performance loss (less WCE) compared to IB, except DOF = 2 where we use IB type [1 7] , DIB type [8] (figure 37). But it was expected because in this case DIB do not include free inputs. In figure 39 we compare IB and DIB with free inputs in the end.

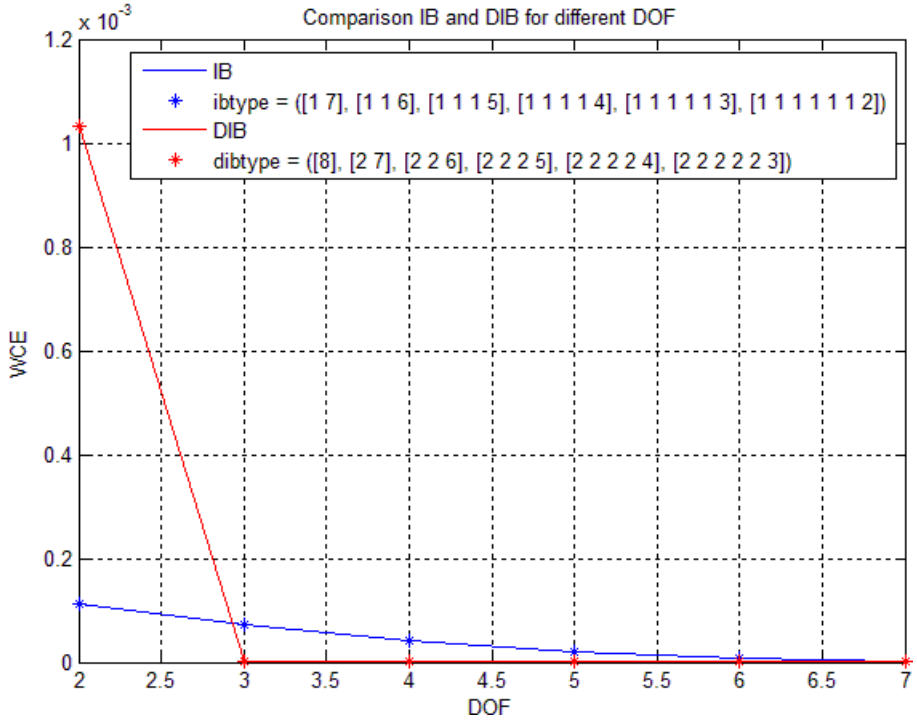


Figure 37: Comparison IB and DIB for different DOF – first free

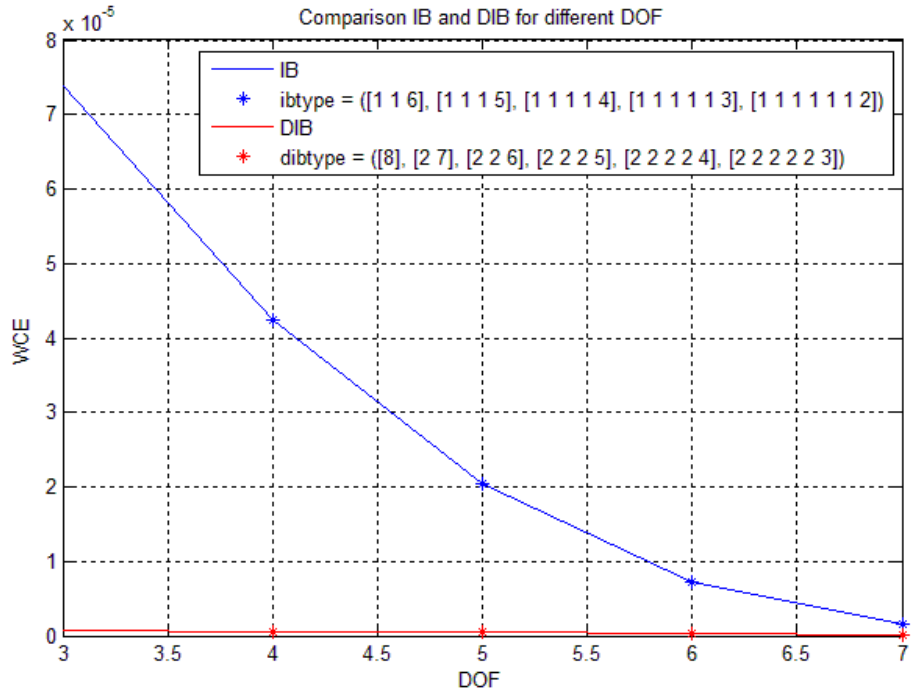


Figure 38: Comparison IB and DIB for different DOF – first free Zoom

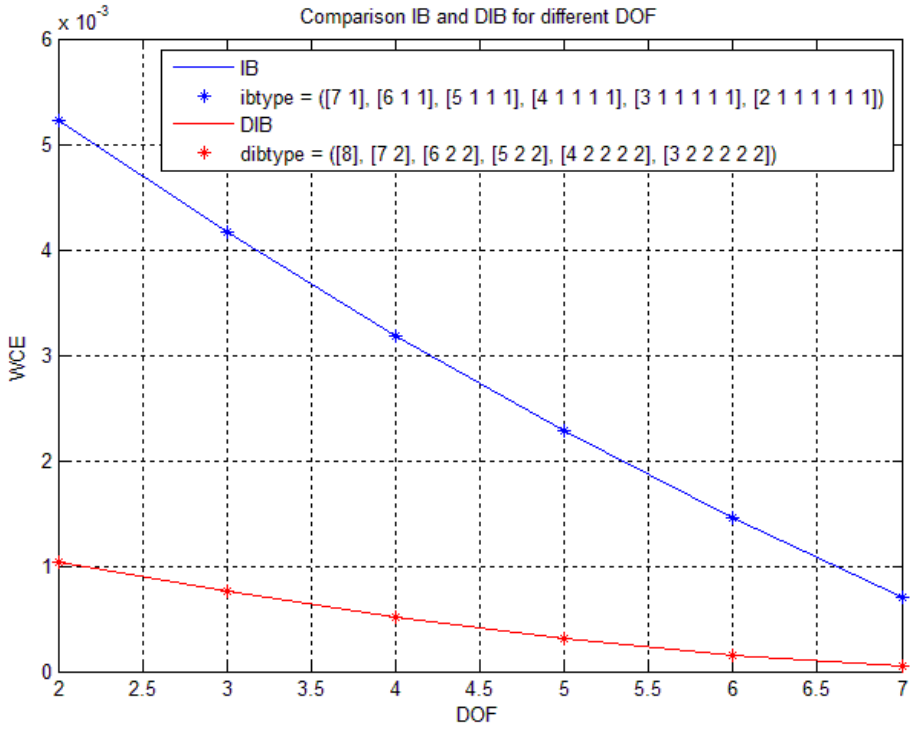


Figure 39: Comparison IB and DIB for different DOF – last free

4.2.2 Conclusion

From these simulations and from values of WCE we can deduce that in both cases (IB and DIB), using free inputs at the beginning of predicted inputs it is better than in the end. Previous results for use example of distillation column prove that reduction of DOF with DIB gives us less control performance loss (less WCE) compared to IB.

5 COMPARISON OF TECHNIQUES FOR SIMPLIFICATION OF MPC

In this part of the thesis we would like to compare a collection of methods that can be used to speed up the computation of MPC. We are using our bilevel program to minimize WCE of outputs, and show how we can use the program to choose the method of simplification with the lowest values of WCE for some desired speed up value. Use of our program is demonstrated on simulations for concrete examples. WCE calculated in the last step of simulation time N_{sim} = 10 and then we are using worst-case disturbances in MPC simulation to calculate sum of WCE over this simulation time. Method of calculation WCE sum was introduced and explained in (chapter - Analazy of WCE - Model reduction).

We would like to compare these methods:

- Input blocking
- Delta input blocking
- Model reduction
- Change of the prediction horizon

We will define speed up coefficient as:

$$\text{Speed up} = \frac{N^3 (n+m_0)^3 f^3}{N_0^3 (n_0+m_0)^3} \tag{60}$$

$$f = \frac{DOF(reuced\ \ with\ \ move\ blocking)}{DOF(original)} \tag{61}$$

where

- n – state dimension
- m – input dimension
- N – prediction horizon
- n_0 – state dimension of full system
- m_0 – input dimension of full system
- N_0 – prediction horizon of full system

We are using similar formula as is in [24], but we include there also f which represent normalized coefficient of degrees of freedom using move blocking.

5.1 Example 1 Desired Speed up ≈ 25 %

Parameters for full controller:

$N_0 = 8$

$n_0 = 16$

$m_0 = 2$

Simplification methods:

a) Model reduction

$N = 8 \qquad f = 1$

$n = 9$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{8^3 (9+2)^3 1^3}{8^3 (16+2)^3} = 0.2282 \Rightarrow 22.82\%$$

b) Change of the prediction horizon

$N = 5 \qquad f = 1$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{5^3 (16+2)^3 1^3}{8^3 (16+2)^3} = 0.2441 \Rightarrow 24.41\%$$

c) Move blocking

$N = 8 \qquad f = \dfrac{DOF(reuced\ \ with\ \ move\ blocking)}{DOForiginal} = \dfrac{5}{8}$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{5^3 (16+2)^3 \left(\frac{5}{8}\right)^3}{8^3 (16+2)^3} = 0.2441 \Rightarrow 24.41\%$$

- Input blocking type: [1 1 1 2 3]
- Delta input blocking type: [2 2 3 4]

In first example we speed up MPC 4 times compared to the speed of the nominal MPC. The calculation time could not be reduced to exactly 25 % of the nominal MPC calculation time. The reason is formulation of our speed up formula. Because of this the values of speed up are different for these methods. Using model reduction we obtained 22.82 % speed up, using change of the prediction horizon and move blocking it was 24.41 %. The biggest WCE of outputs we get using change of the prediction horizon and the difference comparing with other simplify approaches is very big, what is clear see in figure 40. Comparing model reduction, input blocking and delta input blocking in figure 41 we get the best results using delta input blocking.

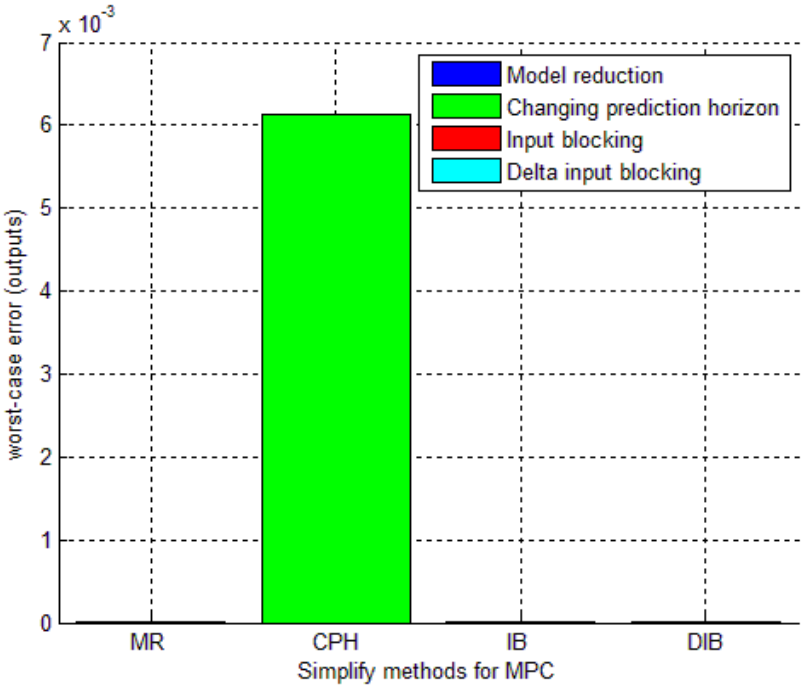


Figure 40. Example 1

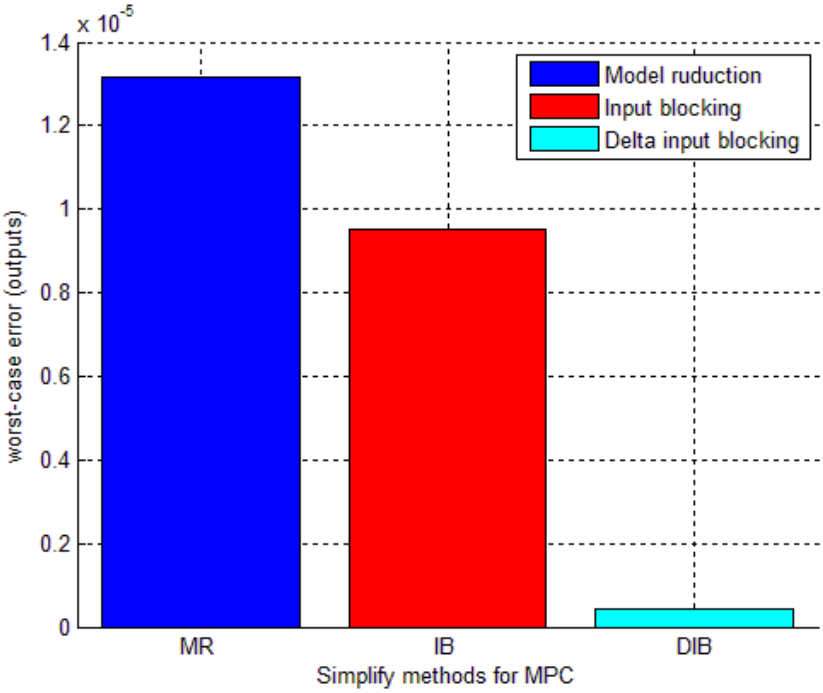


Figure 41. Example 1

5.2 Example 2 Desired Speed up ≈ 50 %

Parameters for full controller:

$N_0 = 8$

$n_0 = 16$

$m_0 = 2$

Simplification methods:

a) Model reduction

$N = 8 \qquad f = 1$

$n = 12$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{8^3 (12+2)^3 1^3}{8^3 (16+2)^3} = 0.4705 \Rightarrow 47.05\%$$

b) Change of the prediction horizon

$N = 6 \qquad f = 1$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{6^3 (16+2)^3 1^3}{8^3 (16+2)^3} = 0.4219 \Rightarrow 42.19\%$$

c) Move blocking

$N = 8 \qquad f = \dfrac{DOF(reuced\ \ with\ \ move\ blocking)}{DOForiginal} = \dfrac{6}{8}$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{5^3 (16+2)^3 \left(\frac{6}{8}\right)^3}{8^3 (16+2)^3} = 0.4219 \Rightarrow 42.19\%$$

- Input blocking type: [1 1 1 1 2 2]
- Delta input blocking type: [2 2 2 3 3]

In our second example we reduce the MPC calculation time to 50 %. Using model reduction we obtained 47.05 %, using change of the prediction horizon and move blocking it was 42.19 %. The biggest WCE of outputs we get as before using change of the prediction horizon and the difference comparing with the other approaches is huge. In figure 43 we can see that WCE obtained using delta input has the lowest value compared to model reduction and input blocking.

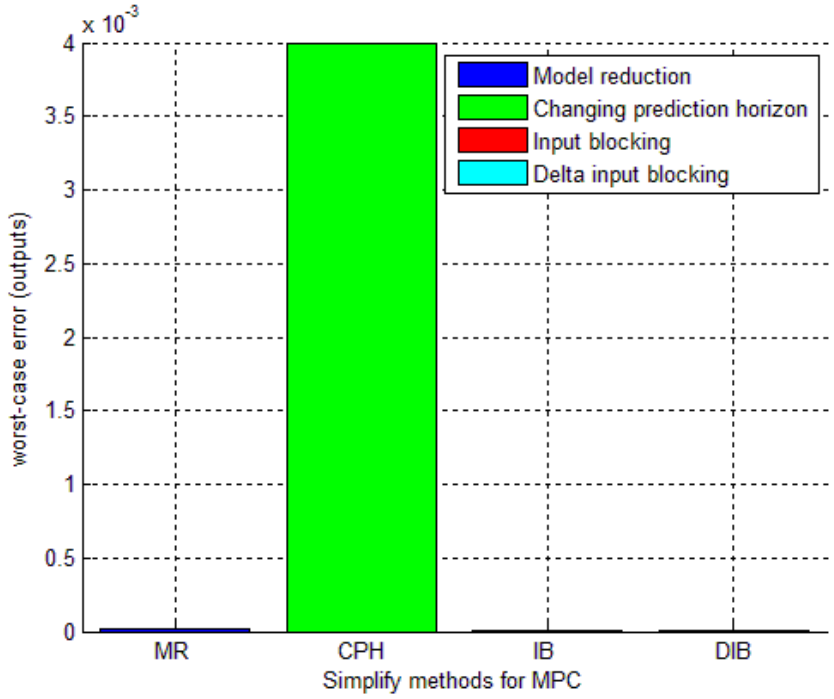


Figure 42. Example 2

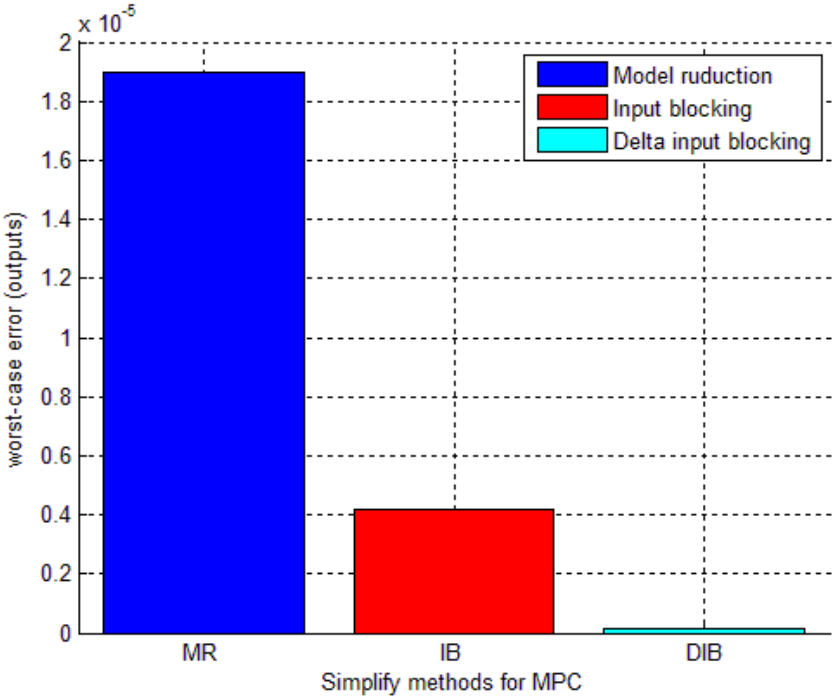


Figure 43. Example 2

5.3 Example 3 Desired Speed up ≈ 75 %

Parameters for full controller:

$N_0 = 8$

$n_0 = 16$

$m_0 = 2$

Simplification methods:

a) Model reduction

$N = 8 \qquad f = 1$

$n = 14$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{8^3 (14+2)^3 1^3}{8^3 (16+2)^3} = 0.7023 \Rightarrow 70.23\%$$

b) Change of the prediction horizon

$N = 7 \qquad f = 1$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{7^3 (16+2)^3 1^3}{8^3 (16+2)^3} = 0.6699 \Rightarrow 66.99\%$$

c) Move blocking

$N = 8 \qquad f = \dfrac{DOF(reuced\ \ with\ \ move\ blocking)}{DOForiginal} = \dfrac{7}{8}$

$n = 16$

$m = 2$

$$\text{Speed up} = \frac{N^3 (n + m_0)^3 f^3}{N_0^3 (n_0 + m_0)^3} = \frac{5^3 (16+2)^3 \left(\frac{7}{8}\right)^3}{8^3 (16+2)^3} = 0.6699 \Rightarrow 66.99\%$$

- Input blocking type: [1 1 1 1 1 1 2]
- Delta input blocking type: [2 2 2 2 2 3]

In our third example we reduce the MPC calculation time to 50 %. Using model reduction we obtained 70.23 %, using change of the prediction horizon and move blocking it was 66.99 %. The biggest WCE of outputs we get as in both previous examples using change of the prediction horizon and the difference comparing with the other approaches is huge. In figure 45 we can see that WCE obtained using delta input has the lowest value compared to model reduction and input blocking.

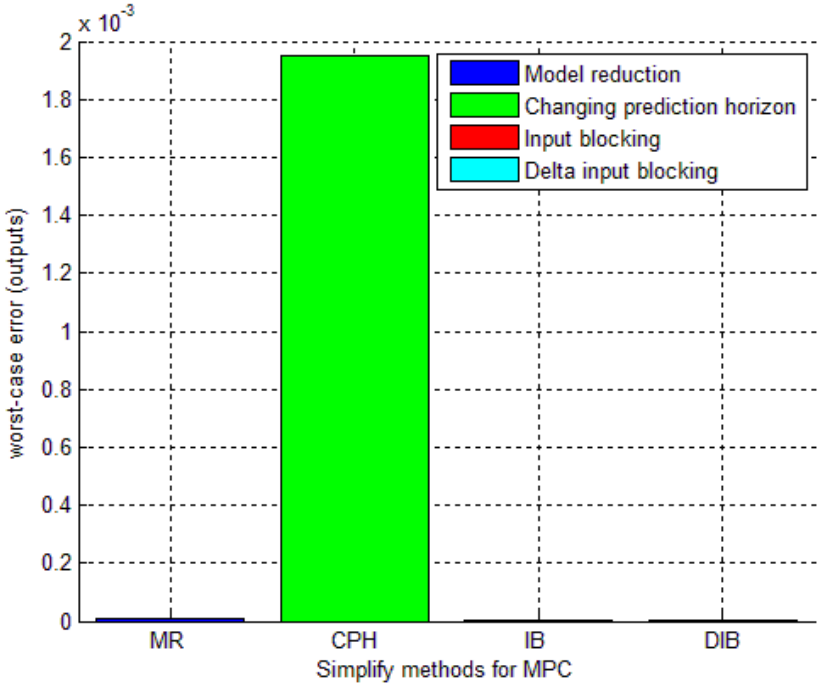


Figure 44. Example 3

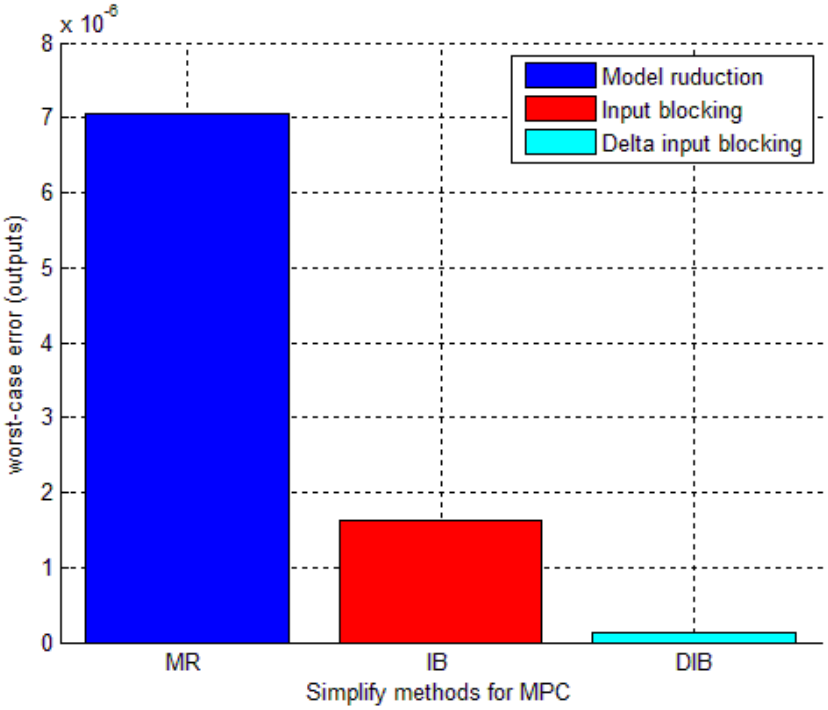


Figure 45. Example 3

5.4 Conclusion

In the previous examples, we tested the functionality of our bilevel program for purposes of finding the best simplification method for MPC problem. It was shown that the best choice for our example of distillation column is delta input blocking, because using this method ensures the lowest WCE. The worst results were achieved using change of the prediction horizon.

6 CONCLUSION

MPC was introduced in first part of the thesis and we were discussing about a problem of increasing of the MPC complexity, when size of the system model grows larger as well as the control horizon and the number of constraints is increasing. Afterwards methods such as Model Reduction, Move Blocking, Change of the Prediction Horizon and Change of the Sampling Time that can be used for simplification of the MPC and making MPC faster were proposed. The other part was about formulation, implementation and then about solving of the MPC problem using mathematical model including disturbances.

Main goal of this diploma thesis was to analyze and compare system response using MPC implemented on a reference and simplified controller. The approach how to find the worst-case difference between the reference controller and simplified controller was introduced. To find the worst-case error between outputs obtained from using full-order and low-order controller we used bilevel program. Using our bilevel program we were comparing different Model Reductions of our distillation column model. We made a analyze of Input blocking and Delta input blocking methods and using our program we found the optimal values of input blocking and delta input blocking types. Also we compare WCE obtained using input blocking and delta input blocking and we got conclusion that using delta input blocking gives us less control performance loss compared to input blocking.

On concrete reduction of computation time we tested the functionality of our bilevel program for purposes of finding the best simplification method for MPC problem. It was shown that the best choice for our example of distillation column is delta input blocking, because using this method ensures the lowest WCE. The worst results were achieved using change of the prediction horizon.

In the future we assume that our bilevel program could be use in form of toolbox used for finding the best choice of simplification method for MPC.

7 RESUMÉ

Prediktívne riadenie s modelom (MPC) patrím medzi pokročilé techniky riadenia, ktoré má významný vplyv na priemyselné riadenie. Matematický model systému sa využíva na výpočet predpovede budúcich výstupov zo systému a riadiace vstupy sú použité na optimalizáciu budúcej odozvy systému. Z toho dôvodu je veľmi dôležité mať model systému, ktorý dostatočne opisuje dynamické vlastnosti riadeného zariadenia.

Jednou z najväčších výhod MPC je možnosť efektívneho zahrnutia obmedzení na vstupy, stavy a výstupy systému. Na druhú stranu v oboch prípadoch (off-line MPC alebo on-line MPC) s rastúcou veľkosťou modelu systému, ako aj s rastúcou dĺžkou predikčného horizontu a s pribúdajúcimi obmedzeniami rastie aj zložitosť samotného MPC. Zložitý regulátor si vyžaduje viac času potrebného na výpočet optimálneho akčného zásahu ako aj väčšie požiadavky na výpočtovú techniku.

Prvá kapitola je venovaná úvodu do problematiky prediktívneho riadenia a možnostiam zjednodušenia MPC problému. Existuje niekoľko zjednodušujúcich metód ako redukcia modelu, blokovanie pohybu, zmena predikčného horizontu, zmena periódy vzorkovania, ktoré môžu byť použité na zjednodušenie MPC problému. Dôležitá je správna voľba kompromisu medzi rýchlosťou a kvalitou riadenia pri použití týchto zjednodušujúcich metód, pretože so zvyšujúcou redukciou stupňov voľnosti, klesá kvalita riadenia.

Druhá kapitola sa zaoberá implementáciou matematického modelu s poruchami do MPC problému. Porovnávané sú tri možnosti riešenia MPC problému na príklade typického chemického zariadenia, ktorým je destilačná kolóna. Jedná sa o nelineárny model destilačnej kolóny („column A“) od Prof. Skogestada. Tento nelineárny model sme linearizovali pomocou funkcie („cola_linearize.m“). Model obsahuje 82 stavov, ktoré sme zredukovali na 16 stavov, pretože je pre nás jednoduchšie pracovať s 16 stavovým modelom. Model obsahuje dva vstupy (spätný tok (reflux) L, tok pár V) a tiež dve poruchy (prietok nástreku F, zloženie nástreku z_F).

Cieľom tejto práce je analyzovať a porovnať odozvy systému pri použití MPC s referenčným a zjednodušeným regulátorom. Pričom sa snažíme nájsť najhoršiu chybu medzi týmito regulátormi. Na základe tejto informácie môžeme určiť ktorú zjednodušujúcu metódu je vhodnejšie použiť. Na nájdenie najhoršej chyby používame bilevel programovanie.

V tretej a štvrtej kapitole sa bližšie zaoberáme využitím redukcie modelu a blokovanie pohybu na zjednodušenie MPC. Využívame pritom náš bilevel program pomocou ktorého hľadáme najhoršiu chybu medzi základným a redukovaným regulátorom, pomocou ktorého porovnávame rôzne stupne redukcie stavov a neskôr aj rôzne druhy blokovania vstupov a blokovania rozdielu medzi vstupmi. Pôvodne sme chceli najhoršiu chybu hľadať ako sumu najhorších chýb počas celej simulácie. Tento spôsob sa však ukázal ako veľmi výpočtovo a časovo náročný. Preto sme sa po analýze rozhodli využiť spôsob pri ktorom hľadáme najhoršiu chybu v poslednom simulačnom kroku a následne využívame MPC simuláciu v ktorej využívame vypočítanú najhoršiu možnú poruchu ako počiatočnú podmienku a počas MPC simulácie počítame sumu rozdielov medzi výstupmi z riadeného systému s použitím referenčného a zjednodušeného regulátora.

Nakoniec v poslednej kapitole porovnávame numericky aj graficky najhoršie chyby získané použitím rôznych zjednodušujúcich techník, ktoré môžu byť použité na zrýchlenie MPC. Výsledkom tohto porovnania je, že pre náš zvolený model destilačnej kolóny je najvhodnejšie použiť blokovanie rozdielu medzi dvoma nasledujúcimi vstupmi, pretože s použitím tejto metódy sme dosiahli najlepšie výsledky. Naopak najhoršou metódou sa ukázala metóda zníženia predikčného horizontu.

8 References

[1] D. Q. Mayne, J. B. Rawlings C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[2] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[3] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering Practice*, 11:733–764, 2003.

[4] M. Herceg. Real-Time Explicit Model Predictive Control of Processes, PhD thesis pages 29 – 46, 2009

[5] Rawlings, J. B. (2000). Tutorial overview of model predictive control. IEEE Control Systems Magazine, 20, 38–52.

[6] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer Verlag, 1st. edition, 1999.

[7] M. Kvasnica. Model predictive control (MPC) Part 1: Introduction. Lecture from MPC

[8] R. Cagienard, P. Grieder, E.C. Kerrigan, M. Morari. Move Blocking strategies in receding horizon control, Journal of Process Control 17 (2007) 563-570

[9] A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, Automatica 38 (1) (2002) 3–20

[10] P. Grieder, F. Borrelli, F.D. Torrisi, M. Morari, Computation of the constrained infinite time linear quadratic regulator, Automatica 40 (4) (2004) 701–708.

[11] U. Halldorsson, M. Fikar, H. Unbehauen, Nonlinear predictive control with multirate optimisation step lengths, IEE Proceedings – Control Theory and Applications 152 (3) (2005) 273–284.

[12] S. Gravdahl J. Hovland. Complexity Reduction in Explicit MPC through Model Reduction. In *17th IFAc World Congress*, 2008.

[13] S. Beck C.L. Sivakumar. MRedTool - a MATLAB Tooolbox for Model Reduction of Multi-dimensionals Systems. In *43rd IEEE Conference on Decision and Control*, 2004.

[14] Quintana-Ort´ı G. Benner P., Quintana-Ort´ı E.S. State-Space Trunctation Methods for
Parallel Model Reduction of Large-Scale Systems. *Parallel Computing*, 8:203–214, June
2003.

[15] Peter Benner, Enrique S. Quintana-Ort´ı, and Gregorio Quintana-Ort´ı. Efficient numerical algorithms for balanced stochastic truncation, 2001.

[16] B. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981.

[17] H. Manum, S. Skogestad. Bilevel programming for analysis of reduced models for use in model predictive control. International Conference – Cybernetics and informatics, Vyšná Boca, SR 2010

[18] J. Nocedal and S. J. Wright. Numerical Optimization. Springer Series in Operations Research. Springer-Verlag, 1999.

[19] H. W. Kuhn and A. W. Tucker, Nonlinear programming, in Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, J. Neyman, ed., Berkeley, CA, 1951, University of California Press, pp. 481-492.

[20] Niclas Andr´easson, Anton Evgrafov, and Michael Patriksson, An Introduction to Continuous Optimization: Foundations and Fundamental Algorithms

[21] S. Skogestad. Dynamics and control of distillation columns - a tutorial introduction. *Trans IChemE, Part A*, 75:539–562, September 1997.

[22] http://www.nt.ntnu.no/users/skoge/book/matlab_m/cola/cola.html#init

[23] http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php?n=Tutorials.Big-M

[24] Yang Wang, Stephen Boyd. Fast Model Predictive Control Using Online Optimization, Proceedings of the 17th World Congress The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

9 APPENDICES

Appendix A: Numerical values of matrices: *A, B, C, D, B_d, D_d*

A =

Columns 1 through 10

0.9790	-0.0072	0.0357	-0.0235	-0.0040	-0.0046	0.0022	0.0023	-0.0013	0.0022
-0.0020	0.9470	0.0173	-0.1052	-0.0076	0.0512	0.0056	0.0067	0.0035	-0.0084
-0.0167	0.0188	0.8654	0.1952	0.0279	0.0633	-0.0160	-0.0175	0.0097	-0.0177
0.0122	0.0233	-0.0648	0.3132	-0.5728	0.1195	0.1603	-0.0105	-0.0366	0.0163
-0.0021	0.0808	-0.0099	0.3355	-0.0752	-0.0459	-0.2058	0.4425	-0.0175	0.0315
-0.0086	0.0036	-0.1116	0.0911	0.1953	0.6889	-0.1367	-0.0275	-0.1021	0.1579
0.0017	0.0108	0.0025	0.0410	0.1080	-0.1490	-0.4306	-0.0742	-0.3602	-0.1293
0.0052	-0.0104	0.0104	0.1375	0.2372	0.0415	-0.0338	0.2258	0.4025	-0.0665
0.0039	0.0019	0.0164	0.0159	0.0666	0.1057	0.0953	0.3466	0.1200	0.1426
-0.0015	-0.0008	-0.0137	-0.0411	0.0471	-0.0894	0.0738	0.2564	-0.0199	0.3037
0.0018	-0.0015	0.0134	-0.0578	-0.0638	0.1306	-0.0145	-0.0009	-0.1849	-0.1456
-0.0010	-0.0020	0.0150	0.0144	0.0460	0.0667	-0.0292	-0.0670	-0.0122	-0.0369
-0.0006	-0.0037	0.0084	-0.0113	-0.0401	0.0528	0.0546	0.1008	0.0174	0.0369
-0.0004	-0.0033	0.0005	0.0234	0.0132	-0.0099	-0.0069	-0.0679	0.0648	0.0412
0.0000	0.0002	-0.0016	0.0165	0.0188	-0.0118	0.0257	-0.0527	0.0249	0.0191
0.0004	-0.0006	-0.0013	0.0005	-0.0252	0.0052	-0.0335	0.0187	0.0476	-0.0831

Columns 11 through 16

-0.0008	-0.0008	0.0002	0.0001	0.0001	-0.0004
0.0081	0.0006	0.0021	0.0008	0.0004	0.0015
0.0077	0.0056	-0.0015	-0.0009	-0.0007	0.0032
-0.0078	-0.0181	0.0215	0.0116	-0.0018	-0.0003
0.0757	0.0074	-0.0058	-0.0130	0.0216	-0.0059
-0.1035	-0.0361	-0.0157	-0.0076	0.0028	-0.0207
0.0299	-0.1352	0.0709	0.0489	-0.0183	0.0062
-0.2971	-0.0533	0.2068	0.0577	-0.0213	-0.0028
0.1788	0.0804	-0.0542	0.0378	-0.1186	-0.0146
-0.0911	-0.3222	-0.1331	0.1186	-0.0641	0.1378
0.2637	0.1010	0.1357	0.2306	-0.0724	-0.0543
0.0505	0.1767	-0.0309	-0.0829	-0.1020	-0.1097
-0.0106	-0.0668	0.1353	0.1295	0.1067	-0.0962
-0.0077	0.1028	-0.1481	0.0798	-0.1039	0.0180
0.1453	0.0169	-0.0078	0.1156	0.0315	0.0393
0.0392	0.0626	-0.0041	-0.0475	-0.0114	0.0721

B =

0.1254	-0.1302
-0.0478	-0.0214
0.0008	0.0390
-0.1356	-0.0239
0.0739	-0.0037
0.0171	-0.0016
-0.0131	-0.0095
-0.0274	0.0113
-0.0030	0.0188
-0.0017	-0.0041
-0.0034	0.0049
0.0012	-0.0156
-0.0000	-0.0078
-0.0013	-0.0035
0.0001	0.0008
0.0002	0.0047

C =

Columns 1 through 10

0.1250	-0.0857	-0.1217	0.0102	-0.0133	0.0948	0.0106	0.0071	0.0721	-0.0786
0.1923	0.0997	-0.1498	0.1939	0.1336	0.0000	0.1262	0.0600	-0.0661	-0.0508

Columns 11 through 16

0.0215	-0.0818	-0.0439	0.0021	-0.0109	-0.0288
-0.0675	0.0372	-0.0437	0.0469	0.0312	-0.0106

Bd =

0.0739	0.1200
0.1210	-0.0183
-0.0029	0.1809
-0.0743	0.0400
-0.0720	-0.0109
0.0127	0.0678
-0.0199	-0.0014
0.0089	-0.0174
-0.0046	-0.0142
0.0009	0.0125
-0.0022	-0.0091
-0.0016	-0.0075
0.0021	-0.0038
0.0043	-0.0006
0.0007	0.0002
0.0017	-0.0001

D =

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Dd =

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Appendix B: List of software on CD

Model of distillation column files ("column A")

- **mathematical model and linearization**
  - cola_lv.m, colamod.m, cola_lv_lin.m,
- **model reduction (truncation)**
  - model_baltrunc.m, baltrunc.m

1 Implementation of the model with disturbances in MPC

- **Functions for obtaining $X, Y, Z, \tilde{Q}, \tilde{R}$ matrices**
  - QRfun.m, xyfun.m

- **Scripts to solve Problem 1,2,3 and comparing these MPC problems**
  - problem1.m, problem2.m, problem3.m, compareP123.m

1 Model reduction worst-case error analysis

a) WCE calculated in the last step of simulation

- **Calculation WCE for different Nsim, and different reduced order models**
  - dif_red_order.m, yalmip_useCHS_trajectory_check.m

- **Functions for calculation WCE using MPC simulation, making closed loop simulation,**
  - analyze_dro.m, matrices_analyze_dro.m, MPC_solv.m

b) WCE sum calculated with MPC sim

- **Functions for calculation WCE sum using MPC simulation**
  - make_WCEsum_plots.m, compare_sum_WCE.m, sum_WCE.m

c) WCE sum calculating with updating objective function

- **Mostly the same like in "a) WCE calculated in the last step of simulation", but in yalmip_useCHS_trajectory_check.m we are using updating objective function**

2 Move blocking worst-case error analysis

- **Script for comparison of IB and DIB WCE calculated in the last step of simulation and WCEsum**
    - **comparison_IB_DIB.m**
- **Functions for calculation IB matrix and DIB matrix**
    - **make_blocking.m, make_delta_blocking.m**
- **Functions using to calculate WCE sum**
    - **sum_IB_WCE.m, sum_DIB_WCE.m, MPC_solvIB.m, MPC_solvDIB.m, dif_mb_WCE.m, dif_delta_mb_WCE.m, matrices_analyze_dro.m**
- **Calculation WCE in last step for IB and DIB**
    - **moveblocking_onestep.m, deltamoveblocking_onestep.m**

3 Comparison of techniques for simplification of MPC

- **Script for comparison results from simplification methods**
    - **speedup_comparison.m**
- **Script for analyze WCE sum using change of the prediction horizon method**
    - **prediction_horizon_WCE_analyze.m**
- **Script for analyze WCE sum using model reduction method**
    - **model_reduction_WCE_analyze.m**
- **Script for analyze WCE sum using move blocking method**
    - **move_blocking_WCE_analyze.m**

4 Figures

Figures using in Diploma thesis:

1. **Implementation of the model with disturbances in MPC**
2. **Model reduction worst-case error analysis**
3. **Move blocking worst-case error analysis**
4. **Comparison of techniques for simplification of MPC**