SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

INSTITUTE OF INFORMATION ENGINEERING, AUTOMATION AND MATHEMATICS

S T U • •
• • • • •
F C H P T
• • • • •

Geometric Approach of Approximation of Explicit MPC

Diploma thesis

FCHPT-5414-50950

2012 **Bc. Bálint Takács**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

INSTITUTE OF INFORMATION ENGINEERING, AUTOMATION AND MATHEMATICS

Geometric Approach of Approximation of Explicit MPC

Diploma thesis

FCHPT-5414-50950

Study program: Automation and information engineering in chemical and food Technologies

Study field: 5.2.14 Automation

Workplace: Faculty of Chemical and Food Technology

Supervisor: Doc. Ing. Michal Kvasnica, PhD.

Consultant: Ing. Alexander Szűcs

2012 **Bc. Bálint Takács**

I would like to express my gratitude to my supervisor Doc. Ing. Michal Kvasnica and to my consultant Ing. Alexander Szűcs for their help, guidance, patience which inspired me. Special thanks go to my family, friends and schoolmates who supported me during my study.

Bálint Takács Bratislava, 2012

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

Faculty of Chemical and Food Technology
Academic year: 2011/2012
Reg. No.: FCHPT-5414-50950

S T U
F C H P T

DIPLOMA THESIS TOPIC

Student:	**Bc. Bálint Takács**
Student's ID:	50950
Study programme:	Automation and Informatization in Chemistry and Food Industry
Study field:	5.2.14 Automation
Thesis supervisor:	doc. Ing. Michal Kvasnica, PhD.
Consultant:	Ing. Alexander Szűcs

Topic: **Geometric Approach to Approximation of Explicit MPC**

Specification of Assignment:

Explicit Model Predictive Control (MPC) allows to reduce the implementation effort by pre-computing the optimal feedback law and storing it as a look-up table. Such a table, however, is often complex, which often induces problems in practical implementation. Objective of this thesis is therefore to reduce complexity the explicit MPC feedback law. This task is achieved by approximating the optimal value function by a simpler function, from which a simpler, albeit suboptimal, feedback law is recovered.

Partial goals of the thesis are as follows:
1) provide an overview of Model Predictive Control;
2) propose a method for approximating the value function in a way such that closed-loop stability is preserved;
3) analyze complexity of the approximation scheme;
4) explain the method by means of illustrative examples.

Length of thesis: 50

Assignment procedure from:	13. 02. 2012
Date of thesis submission:	19. 05. 2012

Slovenská technická univerzita v Bratislave
Fakulta chemickej a potravinárskej technológie
dekanát
- 11 -

L. S.

Bc. Bálint Takács
Student

[illegible]

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

[illegible]

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

Abstract

This work is aimed to solve predictive control problems. The aim of this work is to study the basic ideas of the predictive control. In this work we are dealing mainly with the explicit predictive control. The main goal of this work was to explore the abilities to reduce the cost of the controller. The work can be divided in to the two parts. The first part discusses the theoretical part of the predictive problem. In the second part we show the method how to reduce the size of the look up table.

Key words: Predictive controls, convex optimization, look-up table

Abstrakt

Táto práca je zameraná na riešenie problémov prediktívneho riadenia. Cieľom práce je štúdium základnej myšlienky prediktívneho riadenia. Práca sa zaoberá predovšetkým s explicitným prediktívnym riadením. Hlavnou úlohou práce je preskúmať možnosti zníženia ceny riadenia. Práca je rozdelená do dvoch častí. Prvá sa zaoberá s teoretickými základmi prediktívneho riadenia. V druhej časti je rozpísaná metóda na redukciu zložitosti vyhľadávacej tabuľky.

Kľúčové slová: prediktívne riadenie, konvexná optimalizácia, vyhľadávacia tabuľka

Figures

Nomenclature

α	slope
β	affine term
x	vector of states
u	vector of inputs
J	objective function
Q	weighting matrix for states
R	weighting matrix for inputs
A	state space representation
B	state space representation
S	shape
R	region
λ	auxiliary variable
$\|x\|_1$	1 norm
$\|x\|_2$	2 norm
$\|x\|_2^2$	square of 2 norm
$\|x\|_\infty$	∞ norm
C_up	slopes of the upper objective function
D_up	affine term of the upper objective function
C_down	slopes of the lower objective function
D_down	affine term of the lower objective function
K	gain

T_i	integral time constant
T_d	derivate time constant
e	regulation error
w	reference
u	inputs
y	outputs

Contents

1. Introduction

1.1. History of predictive control

The first control algorithm, which was successfully implemented on real plants date back to nineties of the last century. The idea of model predictive control (MPC) was published in the early seventies. Despite of this fact, it was not used in real-time applications due to the lack of powerful computers. Mainly, this was the reason why such controllers were first implemented on processes with slow dynamics (e.g. chemical and petrochemical industry). The petrochemical industry was the first place where it was successfully used, but after it became useful in other part of industries, for example in mechanical engineering or robotics. It can be used wherever where is necessary to count to the optimal control some restrictions. The most general optimization problem can be formulated in next way: get the maximum (heat, money …) at minimum input (energy, work, …).

1.2. Idea of model predictive control

The efficiency of predictive control is the following: it is able to calculate, respectively produce an optimal input which is a product of optimization. The optimization produces a sequence of input which respects the constraints. There can be constraints on the state variables, output variables or on the input variables. The main goal of the optimization is to calculate the best input to the system in the actual state. In the sequel I will describe the basic control strategies: PID, LQR, MPC and characterize its advantages and disadvantages. I would like to start with the one of the easiest kind of control, with the PID controller.

1.2.1. PID controller

This is one of these control methods which is used in the industry. More than 80% of the controllers work in this principle. The advantage of this kind of control is following: it has really easy structure, it is cheap, and easy to set. There are only few parameters which are necessary to set. These parameters are the gain (Z), the integral time constant (T_i) and the derivate time constant (T_D). If we know these parameters of controller and we know the control error (e) we can calculate the input (u) to the system. We obtain information of the system behavior (y). The control error is calculated as a difference between the output and the set point (w). The control law of the PID controller has the next form:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{d}{dt}e(t)\right) \tag{1}$$

As you can see this is a really easy type of equation. There is only addition and multiplication, so this kind of controller doesn't need a performance hardware or CPU. Another advantage is the simple structure, so we need only few bytes in memory of our hardware. The big disadvantage of PID is that, it can be used only in case of SISO (single input-single output) systems. In case of MIMO (multi input- multi output) system, there exist a method how to control it with the PID controller. This method name is a decoupling compensator, which is able to create SISO loops from the MIMO system, but it don't have to work in every single case, and sometimes it is really difficult to cut up system into SISO loops. The next disadvantage is that, it can't work with constraints. There is a possibility to use the anti-windup, but if we have to control the unstable system this approach is not the best, because it can more destabilize the controlled system, and this is unacceptable, because the first task of control is to stabilize the system. The simple structure, easy set and the cheap implementation are the best advantages of this kind of control. Usually it is used to control for example the valves, which are able to control the pressure and level in the tank, or temperature. These controllers can be set experimentally. Sometimes occur that the model of the system is not punctual. Then they set the parameters by one of many methods (Ziegler-Nichols, Strejc, Cohen-Coon, Haalman, Smith-Morari, …) and leave the possibility for the operators to change the parameters of the controller. After few try-outs they are usually able to set the parameters of the controller which will create an acceptable input. On the 1st figure you can see the scheme of the PID control in the closed loop.

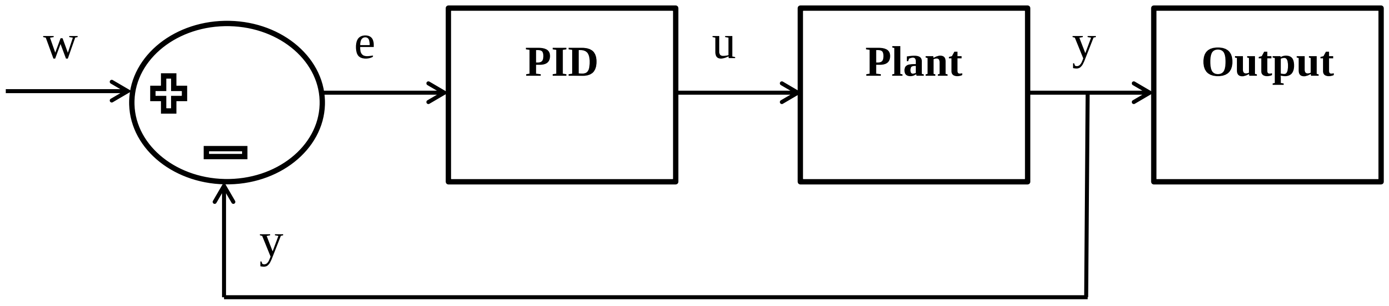

Figure 1.: Control using PID

1.2.2. LQR

The Linear Quadratic Regulator is another type of control. This is a higher level of control, because it can work with more than one state variable. It tries to optimize the input. We are given a linear system which we would like to control and a quadratic objective function. By the objective function the controller is possible to calculate the minimal necessary input to get the system from the current state to the desired state. This method of control can be used not only for the SISO systems, so we don’t need any compensators to create SISO loops from the MIMO system. It works on the base of the Ricatti‘s equation. By solving the algebraic form of Ricatti’s equation we get the parameters of controller, which stabilizes the system and move to the desired state. It calculates the all inputs which are necessary to get the system to the required state. The objective function and the one linear constraint have the following form:

$$min \int_0^\infty x^T Q x + u^T R\, u \tag{2}$$

$$s.t.: \dot{x} = Ax + Bu \tag{3}$$

Where variable x represents all the states, variable u all inputs, and A, B are the matrices of state space representation. What we get is the parameters of the controller (K). The advantage of this approach stems from the fact, the optimization is performed over infinite horizon. It means it calculates until infinity, so it knows what will happen with the system in the future, but it does not take into account the constraints. The optimal input is calculated by next equation:

$$u = K(x).x \tag{4}$$

Where $K(x)$ is the solution of Riccati's equation which a function of the states, and x represents all the states in the system. The $K(x)$ is calculated only once off-line in case of time-invariant system, so if happens something unpredictable it can cause unexpected situation.

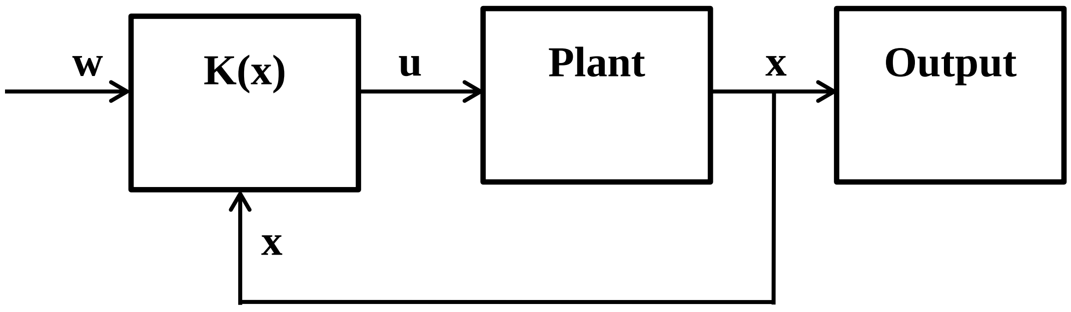

Figure 2.: Control using LQR

Between the advantages of the LQR control is that, it is able to watch to the infinity, so it mean it can give us to best possible performance. This method allowed to used discrete time not only continues. Big disadvantage is that, it can work only with linear model and can't work with constraints.

1.2.3. Model Predictive Control

Model predictive control is one of the best modern methods which was successfully implemented in the wide range of the industry. The main reason why is better than other control method is that because its formulations involves constraints. The basic idea is similar like in the LQR, but now the objective function has a discrete form, so from the integral become summation operator, and the upper bound isn't infinity. So it works only in the finite prediction horizon. Sometimes occurs the LQR gives us better input than MPC, because it can see to the infinity and predictive control works only in finite predictive horizon, but if happened something which wasn't a part of the LQR formulation the LQR isn't able to react on it. In the LQR method the parameter of controller is calculated once at the beginning that is that what we call offline method. The MPC in every single period calculates the optimal input to the system, which satisfies all the criteria and constraints. It is possible to use not only the linear model, but we can use nonlinear or hybrid plant models. The Achilles heel of this method is the difficulty and time consuming of optimization problem. The CPU has only one sample time to calculate the optimal input by solving the optimization problem. If it doesn't able to do that, there is a chance to lost information of the behavior of the system, which in the better case can causes losing of optimality, in worse case loosing the stability. This is the biggest disadvantage of this method. The objective function and the constraints have the following form:

$$J = min \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R\, u_i \tag{5}$$

$$s.t.: x_{t+1} = Ax_t + Bu_t \tag{6}$$

$$x \in X \tag{7}$$

$$u \in U \tag{8}$$

$$x_0 = x(t) \tag{9}$$

Where X and U are set of acceptable values for states and inputs. Other variables are similar as in the section of LQR. N is the size of the predictive horizon. The objective function with the larger predictive horizon will more complex problem, and harder for the solver to solve, so the solving time and the requirements of the memory will increase. On one hand we would like to have a huge predictive horizon, because as bigger is the horizon as much better it can predict the behavior of the system, but in the other way the big prediction horizon causes that the problem will not solvable for the given time. If the solver isn't able to solve the optimization problem in time it is necessary to reduce pre size of the predictive horizon or change the system in other way. For example change the constraints, leave out the not necessary constraints or pre calculate the values of some variables.

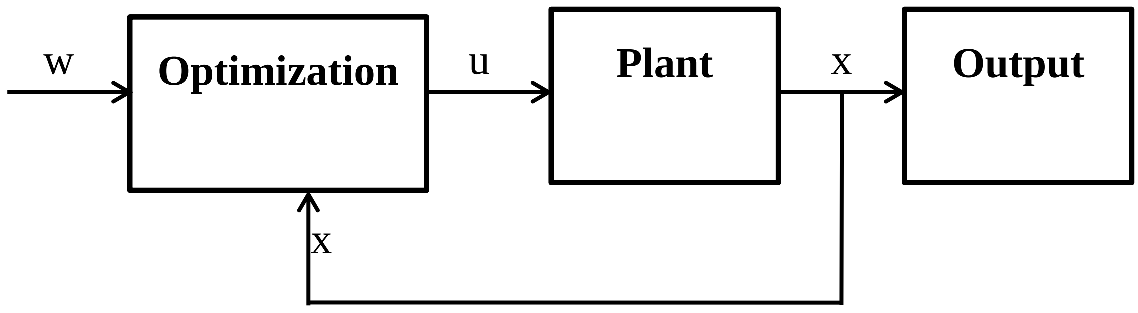

Figure 3.: Control using MPC

1.2.4. Comparison

We would like to show you the difference between these kinds of control. How do they work in the same situation? We are given the car, which is driven by the controller. At first it should be a PID controller. If we will drive the car with the PID it's like to drive without the eye contact on the road. We will not see what is ahead of us. We just see the road from the rearview mirror, so if we leave the road we will register it only than when it appears on the mirror. If we would like to drive the car by the LQR, it's like use the GPS, which know the fastest or the closest way to get from the current location to the required location, so it will be able to control the car, but if there are a crash or an accident on the road, or some table were changed the LQR will not calculate with this fact. The predictive control is like when the car is driven by the human.

1.3. Model Predictive Control

There are two different methods of the predictive control. One is the ON LINE method and the next is the OFF LINE or EXPLICIT method. Both of these methods can be used to control systems, nearly with the same quality. The difference between them is the required time to calculate the optimal input to the system. The online method is the basic method witch calculates the optimization in after obtaining information of the system from the measurement. The explicit method is new approach in the theory of control. It calculates the optimization only once, in the beginning, and after use the explicit form of the control law, which is faster than the online method.

1.3.1. On - line MPC

This kind of method can be used when we have a system with great time constant, for example if we would like to control rectification columns, or other similar process with the big time constant. In the case of the rectification column there can be thousands of variables and hundreds of constraints, so the optimization problem is really complex and hard to solve. There is necessary a good solver and a computer with high performance, which is able to solve the optimization problem. As the size of the predictive horizon will increase, than complicated will the optimization problem. This method has extremely good implementation in the chemical and petrochemical industry, where the processes have big time constant. These slow processes ensure necessary time to solve the optimization problem. The rectification columns time constant is usually between 30 minutes and 60 minutes. This time is enough to calculate the optimal input by the online method.

1.3.2. Explicit MPC

This method can be used to control systems with faster dynamic behavior. Usually is used in a mechanical or electricity engineering. The difference in control between the off line and on line method is in how it calculates the optimal input. The parameters and the constraints of the system are the same, but in this case the optimization is done only once, and we get the explicit form of the control law. We can imagine that explicit form of control law like a simple affine function where we know the slope and the affine term, the only variable which is changing is the variable which represents the state. There isn't only one affine function, there is one for every region. Every single region consists of the same number of the parameters, but the values of the parameters are different. If we have a one dimensional problem, than every region consist of one slope and one affine term. If we have a M dimensional problem, than every region consist of M slope and one affine term.

When we know the actual state of our system, we know the parameters of control law, so we can easily calculate the optimal input. This method is much faster than the on line method of predictive control, because there is optimization only before the beginning. During the running, there is only necessary to find the correct region where is the system. This method is useful until we don't have too much states and inputs. When we have a bigger system, with many constraints the optimizer will create a big amount of regions. The big number of regions can occur that the controller algorithm will not be able to upload to the memory of the controlled system, so the implementation will impossible. Every region increases the amount of the memory which is necessary for a good operation of the controller. If we don't have enough memory for our control algorithm, than we can't drop out some regions. The solution is little bit more difficult. If we can't upload the whole look up table to the memory it can cause that the control will not optimal or what is the worst scenario the process can lead to unstable behavior.

1.4. Open and closed loop in MPC

In this section we would like to show the difference between open and closed loop. Let's imagine the situation when we have a car, and we need to get car from the current position to the other position. There are several type of constraints, like the car has to stay on road, respect speed limits, and secure that the car will not crash into the car ahead of. The main goal of the optimization can be to minimalize the amount of fuel what will car need to get from A to B. Now the predictive horizon we can imagine in this way: every single prediction horizon equals 10 meters. So if we have a small horizon, it‘s like drive a car during the foggy forecast, if we have a longer horizon it's like drive a car in the sunny forecast. The open loop control will work in the following way: we obtain the sequence of optimal input from one measurement and every single input is implemented to the system. After when we get out of the inputs we make another measurement and the optimizer calculates again a sequence of inputs. Returning to the example with car it's like drive the car that way, you open your eyes for a second, watch what is on road, we mean the traffic, the tables, the curves, etc., and after close your eyes and drive with closed eye until you remember the path. This isn't the best attitude, because during you drive without eye contact on road, there can appear unexpected situation, like the car ahead of you will unexpectedly slow down, or some children step on road. If we don't calculate with these options we can hit somebody or crash the car. This is the reason why predictive control is used in closed loop. The basic idea of the closed loop is the same like in the open-loop, but there we implement only the first input from the sequence of inputs. We need information about the system in every single sample time. If we aren't able to measure all the states, than somehow we need to estimate the states, or use an observer, Kalmans filter. After when we have information about the system the optimizer can calculate the sequence of optimal inputs to the system. We need to realize that we get the sequence of inputs, but we implement only the first one. Now is ensured the continuous feedback to the system. The first thing what can look really useless are the other calculated inputs. If we don't use them, why do we want from the optimizer to calculate them? As I mentioned before the moving horizon is like an ability to see to the future, in the example with car it is the distance ahead of car. Imagine the situation when you drive a car and front of you is a concrete wall. If you have a small predictive horizon it is like driving the car with bad visibility conditions. If you don’t see the wall, you can’t stop the car or change the way to avoid the obstracle, but if we have a larger predictive horizon than we will able to see what should happened and changed the way or the speed of the car by using the brake. This was just an example, it doesn’t work only with the cars, it can works with the reactors, with robots, with everything what is necessary to control. On one hand we try to increase the length of the predictive horizon to get better predictions, but on the other hand we would like to decrease the length of the horizon, because it increases the complexity of the optimization problem. As I mentioned before to solve more complex optimization problem is necessary to have more time. In the explicit predictive control it shouldn’t be problem, because the optimization is done only once, but the big predictive horizon can causes that the number of regions will highly increase, what is not desired in this case. This is the worst what should happen. As we can see the length of the prediction horizon must be chosen very carefully.

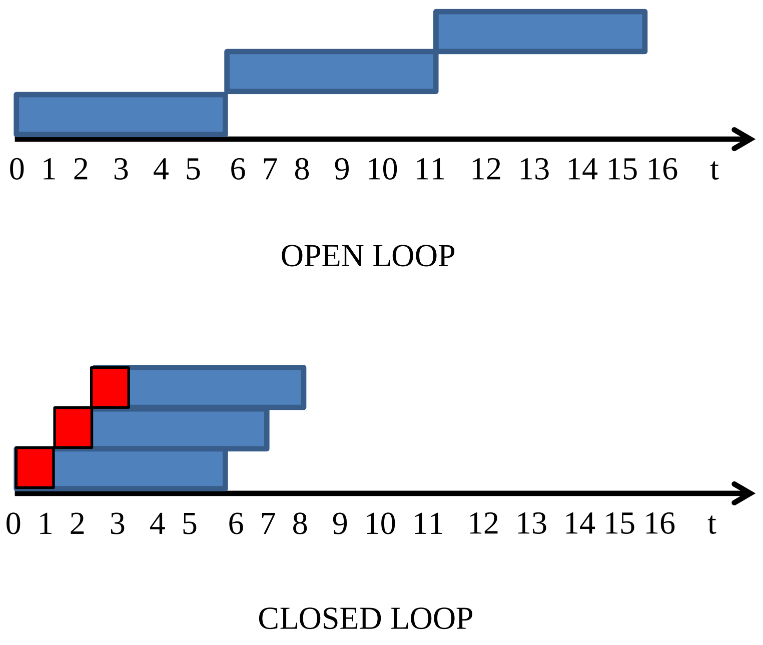
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 t
OPEN LOOP
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 t
CLOSED LOOP

Figure 4.: Open and closed loop 1

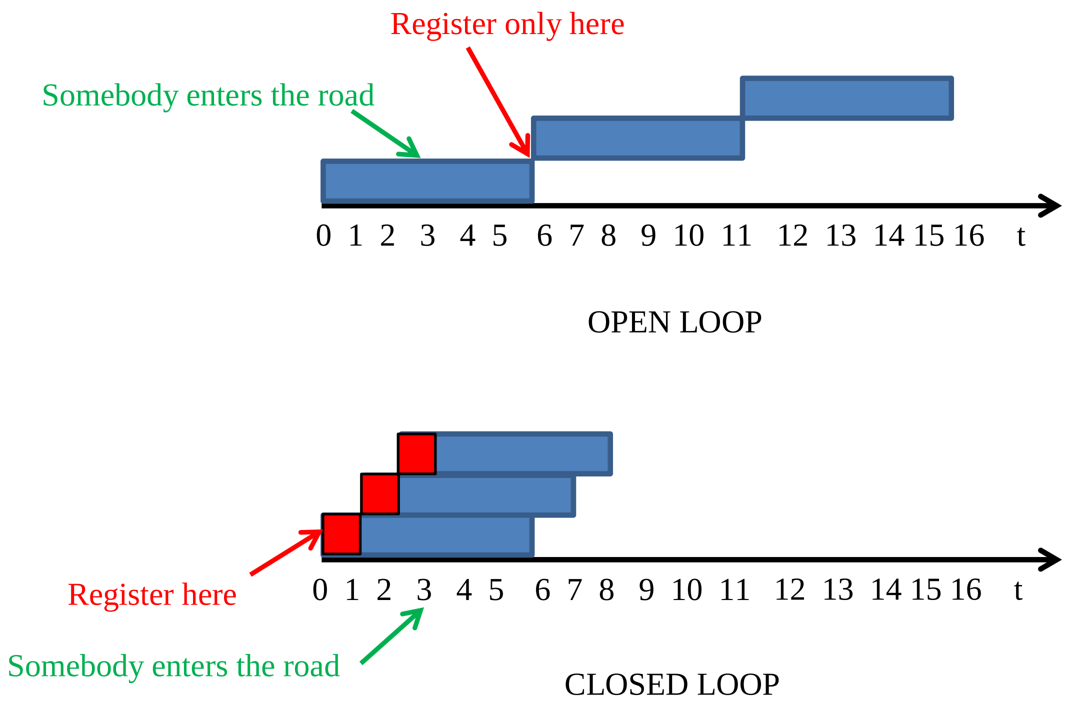

Figure 5.: Open and closed loop 2

1.5. Optimization

We will aim to use convex optimization, because if we find the extreme of the convex optimization problem, we found the global extreme. The optimization problem can be divided in to the two parts, in to the objective function and the constraints. There can be 3 types of objective function. It can be convex, concave or neither. If we would like to minimize the objective function it is good to have a convex function, if we need to maximize the objective function it is good to have a concave function.

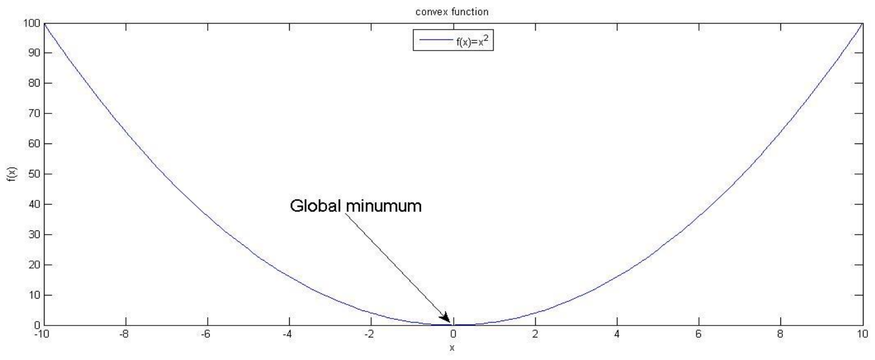

Figure 6.: Convex function

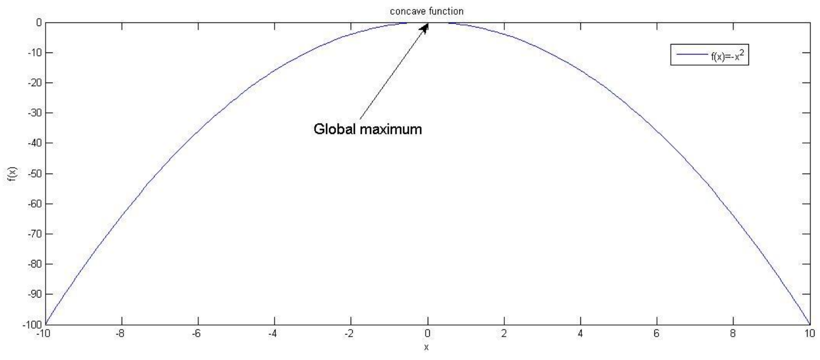

Figure 7.: Concave function

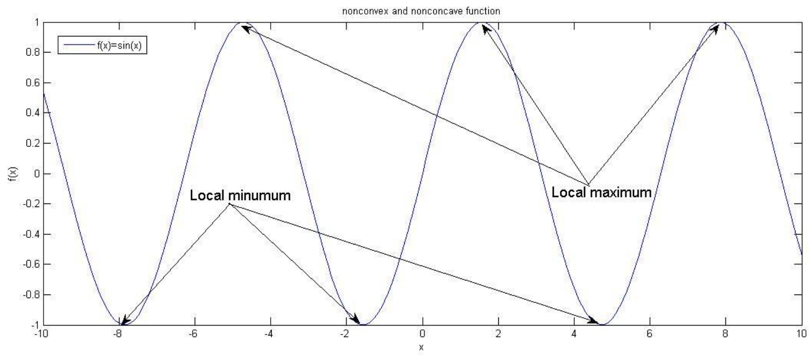

Figure 8.: Nonconvex and nonconcave function

The function is convex when the following inequality holds:

$$f(\theta x - (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \qquad (10)$$

$$\theta \in [0,1] \qquad (11)$$

The aforementioned inequality can be interpreted as follows. We are given convex function and if we connect any of 2 points by a line, and a function will under the line than we can say that the function is convex.

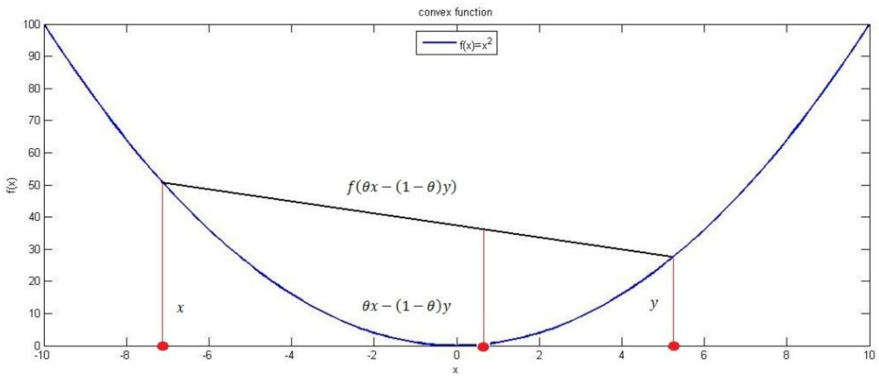

Figure 9.: Geometrical represantation of convex function

The objective function tries to minimize the distance from the setpoint. The distance, what is a scalar value, can be expressed by norms. There are many different types of norms. Ones are the following: 1, 2, ∞.

The 1 norm, which sometimes is called Manhattan norm, calculates the distance like if we are in the city and we would like to get from the A to B, but we can't go through the building, we can use only the streets. The 2 norm we can imagine like if want go from A to B but there aren't any barriers. The ∞ norm calculates only with the longest part of the way. The following figures shows the differences between these norms:

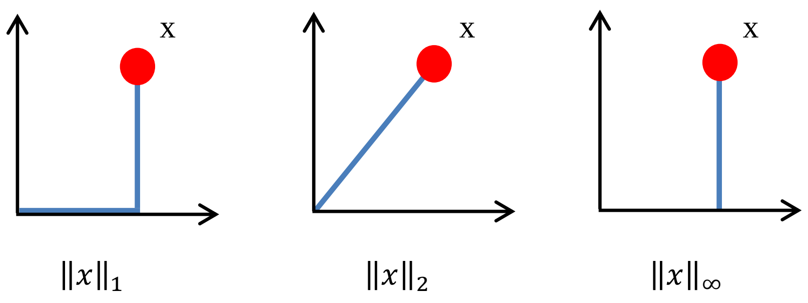

Figure 10.: Norms, distance from origo

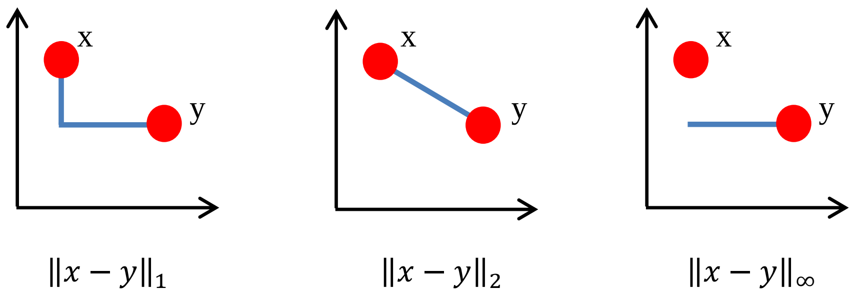

Figure 11.: Norms, distance between x and y

The mathematical formulations of these norms are the following. Variable x is a vector of all state variables.

$$x = [x_1, x_2, \ldots, x_n]^T \tag{12}$$

$$\|x\|_1 = \sum_i |x_i| \tag{13}$$

$$\|x\|_2 = \sqrt{\sum_i x_i{}^2} = \sqrt{x^T x} \tag{14}$$

$$\|x\|_\infty = max_i |x_i| \tag{15}$$

The $\|x\|_2$ is usually used to minimize the amount of energy necessary to move the system from the current state to the required state. The problem with this norm is the square root, which causes nonlinearity. Instead of $\|x\|_2$ norm is frequently used the square of that norm ($\|x\|_2^2$).

$$\|x\|_2^2 = \sum_i x_i{}^2 = x^T x \tag{16}$$

The difference between these norms is that the $\|x\|_2^2$ undervalues small and overvalues large numbers, but there is no difference in the result between them. The outcome from optimization should be the same, it doesn't depend which norm was chosen to use.

As we have the objective function, next what we need to know are the constraints of the system. In predictive control is possible to use constraints in the form of equalities and inequalities. The constraints delimit the allowed place where the optimizer can work. We can have an easy convex set or hard nonconvex set of constraints. It is really easy to show the difference between convex and nonconvex constraints by geometry.

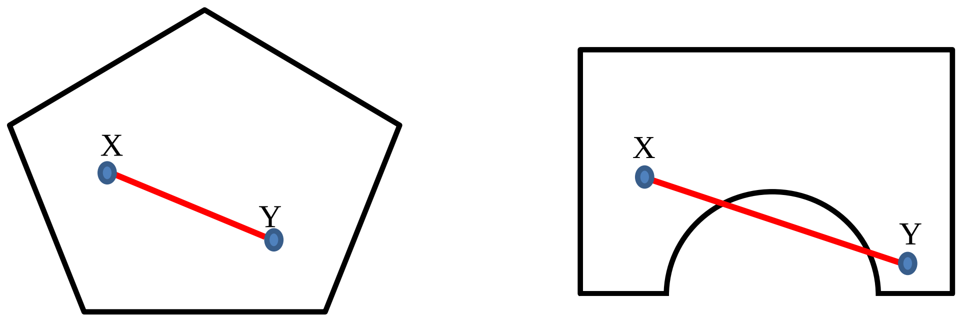

Figure 12.: Convex and nonconvex constraints

On the figure above are two 2 dimensional shapes. The first one is convex and the second one is nonconvex. The way how to decide what kind of shape do we have is to choose 2 different points which are the part of the shape. If we can connect them with a line segment and the whole line segment is a part of the shape than we have a convex shape, but this have to be valid for arbitrary two points in the shape. The mathematical formulation is the following:

We are given a shape (S) and some points which are part of this shape (x, y):

$$x, y \in S \tag{17}$$

We need an auxiliary variable (λ):

$$\lambda \in [0, 1] \tag{18}$$

Than the line between them can be formulated in following way:

$$\lambda \mathrm{x} + (1 - \lambda)\mathrm{y} \in \mathrm{S} \tag{19}$$

It must pay for every value of λ and for every point x, y. If it is valid, than we have a convex shape.

Polytopes are convex shapes, created from finite pieces of halfspaces. The halfspace indicates the allowed place. Polytops are every time convex shapes, so they are really useful in the optimization.

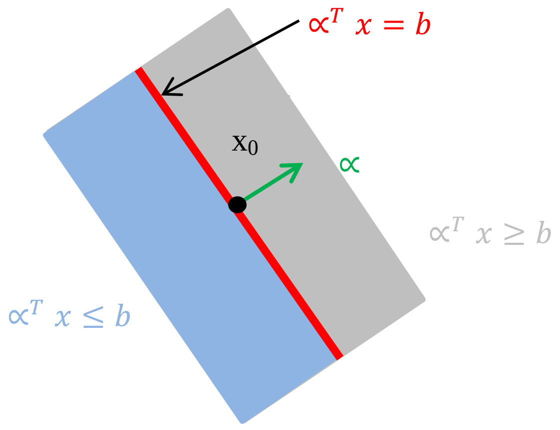

Figure 6.: Halfspace

$$\propto = [\propto_1, \propto_2, \dots, \propto_n] \qquad (20)$$

$$\beta = [\beta_1, \beta_2, \dots, \beta_n] \qquad (21)$$

Where variable α consist from the slopes and the β from the affine term

So if we have a finite number of halfspaces in form $\alpha_i^T x \leq b_i$ then the intersection of these halfspaces give a polytope.

The general form of the optimization problem in the predictive control has the following form:

The objective function

$$\min c^T x \qquad OR \qquad min\ x^T Px + 2Q^T x + R \qquad (22)$$

The constraints

$$Ax \leq B \tag{23}$$

$$Gx = H \tag{24}$$

$$x \in X \tag{25}$$

$$u \in U \tag{26}$$

$$x_t = x(t) \tag{27}$$

The result of this optimization problem is the optimal input to the system.

2. Reduction of complexity

Explicit predictive control is able to reduce the cost of the controller. This type of control precalculates the law of control and saves information in a look-up table. What is necessary is to find the correct position in the table, by some mathematical operations. In case when the look-up table size is too big, then is problem to upload it to the hardware, because the controllers have limited memory. In this section we would like to show and explain the way how to reduce the size of the table.

2.1. Theory

The complexity is primarily given by the number of constraints and secondary by the number of states. Both of these are strongly connected to the length of the prediction horizon. If we reduced the number of constraints we can get simpler look-up table, but this isn’t the good way how to do it. There are few other possibilities how to reduce the complexity, for example there is a chance to simplify the actual explicit control law replaced by easier form. The explicit form of the control law is usually in the following form:

$$u^* = \chi(\mathrm{x}) \tag{28}$$

Where x consists of values of the actual state, $u^*$is the optimal input, and χ consists of the parameter of the control law. It can be everything. The simplest case is that when this is an affine equation. In this work I was working with the affine equations.

2.2. Problem Statement

We are given the system and we are able to calculate the objective function in optimum (Jlow) and in the worst case which is still able to stabilize (Jup) the system. We would like to find the function f(x), which satisfies the following inequation for all state (x):

$$\forall x \qquad J_{low}(x) \le f(x) \le J_{up}(x) \tag{29}$$

Where f(x) has the following form:

$$f(x) = max_k(\alpha_k^T x + b_k) \tag{30}$$

The index k is fixed value, which says us the number of new approximation. We would like to find these new slopes (α^T) and affine terms (β) of the new, easier objective function.

2.2.1. Objective function

The aim of this work was to understand and create a general algorithm which will able to reduce the complexity of the controller. First of all it was necessary to understand the connection between the complexity of the explicit control and the complexity of objective function. If depends one on another, if yes than in what way.

At first time we tried to recognize the difference on the simply SISO system. The control algorithm was written in Matlab. During the work we used some products of

Matlab, and few Toolboxes, like Yalmip, etc. There were some prepared functions, so we didn’t have to create every function that we used in this work.

At first is necessary to define the system with we would like to work.

State-space representation:

$$A = 0.8 \qquad B = 1 \qquad C = 1 \qquad D = 0$$

Constraints for state:

$$-5 \leq x \leq 5$$

Constraints for input:

$$-1 \leq u \leq 1$$

Predictive horizon:

$$N = 5$$

Weighting matrix for state:

$$Q = 1$$

Weighting matrix for input:

$$R = 1$$

Chosen norm:

$$1\ norm$$

This information in Matlab is saved in the next form:

```
clear sysStruct probStruct

sysStruct.A = 0.8;
sysStruct.B = 1;
sysStruct.C = 1;
sysStruct.D = 0;
sysStruct.umax = 1;
sysStruct.umin = -1;
sysStruct.xmax = 5;
sysStruct.xmin = -5;

probStruct.Q = 1;
probStruct.R = 1;
probStruct.norm = 1;
probStruct.N = 5;

nx = mpt_sysStructInfo(sysStruct);
ctrl = mpt_control(sysStruct, probStruct)
```

The state-space representation of the system, the matrices A, B, C, D, are saved in the variables sysSturct.A, sysStruct.B, etc. As you can see, this is a stable system. Here we can find constraints for a maximum and minimum value for input and similarly for the state. Q and R are the weighting matrices. By these matrices we can tune the objective function of the optimization. If the matrix R will have larger value than we say that it is an important or expensive input, so try to use it only when it is really necessary. Not only the size of these matrices are important, larger weight have the rate between the weight matrixes. For example if we set the value 1 for both of them or set the value 100 both of them the result will the same, because the rate is still the same. In this case one norm was used and the length of prediction horizon was declared in N.
There exists a function in Matlab which is able to create from this information the explicit form of control law. I mean we get the number of the regions and the parameters of these regions. This function is prepare_data. If we would like to plot how does the objective function looks like there is another function in Matlab, the plot_pwa. The argument of this function is Jlow or Jup, which are variables created by the function prepare_data. What we get is the form of the objective function.

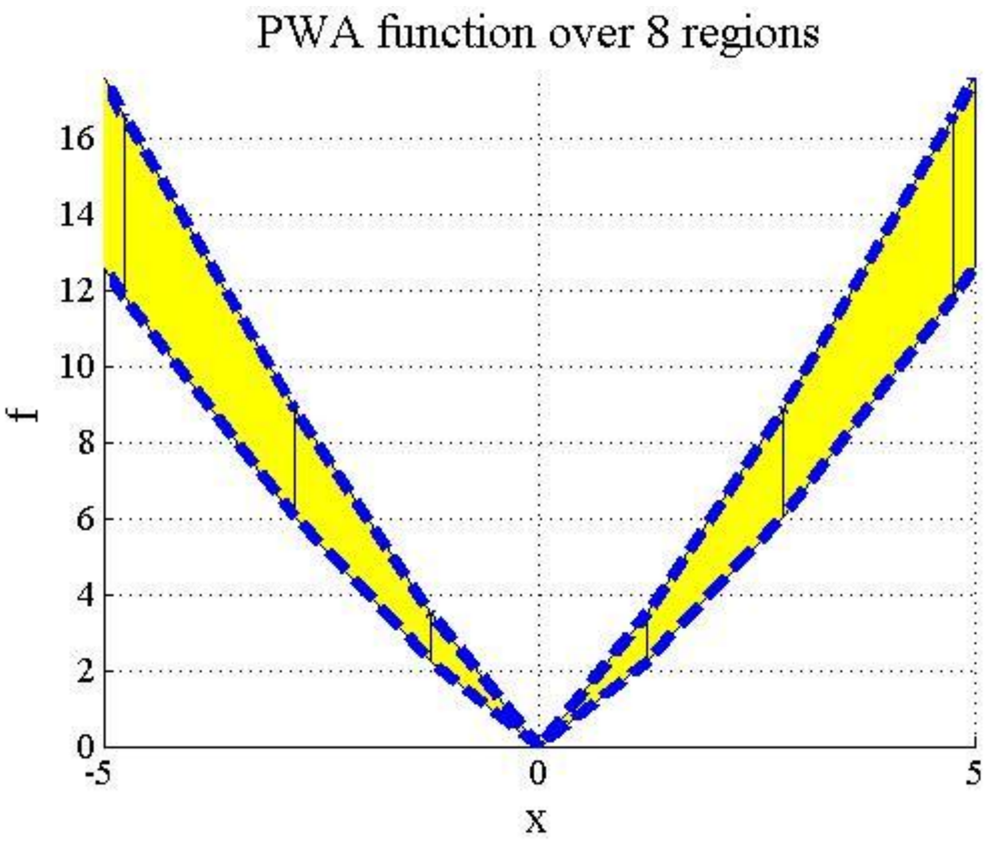

Figure 14.: Objective funtion of stable systém

The lower blue function (Jlow) represents the optimal values of the objective function. The upper function (Jup) is the worst case scenario which is still able to produce the stable feedback. The Jup is calculated in the next way:

$$Jup = Jlow + \|x\|_p + \|u\|_p \tag{31}$$

So the upper bound is calculated from the optimal lower bound (Jlow) and from the addition of the norm of the state and the norm of the optimal input in current state. This is the way how we get the allowed space, where we can find our easier form of control law. This system can be controlled by 8 regions as you can see on the figure 13. Every region consists of the linear parameters from the slope and from the affine term.

They are in the following way:

$$J\_up(x) = \begin{cases} C\_up^T{}_1 x + D\_up_1 & if\ x \in R_1 \quad (32) \\ C\_up^T{}_2 x + D\_up_2 & if\ x \in R_2 \quad (33) \\ C\_up^T{}_i x + D\_up_i & if\ x \in R_i \quad (34) \end{cases}$$

$$J\_low(x) = \begin{cases} C\_low^T{}_1 x + D\_low_1 & if\ x \in R_1 \quad (35) \\ C\_low^T{}_2 x + D\_low_2 & if\ x \in R_2 \quad (36) \\ C\_low^T{}_i x + D\_low_i & if\ x \in R_i \quad (37) \end{cases}$$

So every region has its own parameters. In the same time only one region is active. This is the biggest disadvantage of this method. The high number of regions can occur that it will impossible to implement in real plant. The goal is to reduce the number of these regions, but it is necessary to rest this amount of regions which still can stabilize the process. The first criterion of the control law is the stability and the next one is the quality of control. There are 2 methods how to reduce the number of regions. The first one is the simpler from the mathematical part of the problem, because we need only choose some points from the allowed space. When we have a one dimensional optimization problem it isn't really hard to choose good points for approximation, but in case when we have a many optimization variables than it can be really difficult. If we choose all points from the lower function, from the Jlow it will not efficient, because the lower function is the most complex case. There is no concrete method how to choose good points, but it is necessary to choose some from the allowed space, so the value of these points must be bigger than the lower function (Jlow) and less than the upper function (Jup). We can choose one point from every region, but if we do this the result what we get will not simpler than the nominal problem. So at first what is good to do if we have one dimensional system is study the graph of the objective function. For example if we have the situation what is on the 13th figure. We are able to reduce the number of regions from the original 8 to the 2. This is the best case, but if can reduced the number of regions about only 50 % is really good result.

On the next 3 figures are few possibilities how to choose point for the approximation to reduce the number of the regions. In the first 2 cases there was enough only 2 regions, but in the last case I showed the situation when we have 4 regions. If the new created line between the upper and the lower function is not similar with the lower function, the result is that the solution is not optimal only suboptimal, but still able to guarantee a stability of the control. But if the new line is the same like a lower function than it is optimal, but there is no way how to reduce the number of regions. So there is a compromise between the complexity (number of regions) and the quality of control. We need to know if the less quality is still satisfactory. If not than we need to change the number of regions. The suboptimal solution is worse than optimal, but if the optimal solution is impossible to implement, than is necessary to think in this way, to get the only the suboptimal solution, but less with less complex look-up table.

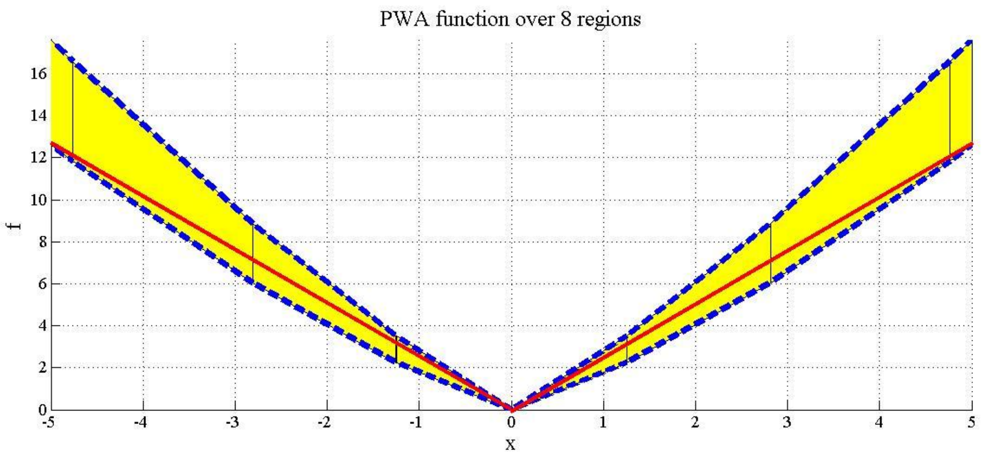

Figure 15.: Reduced complexity 1

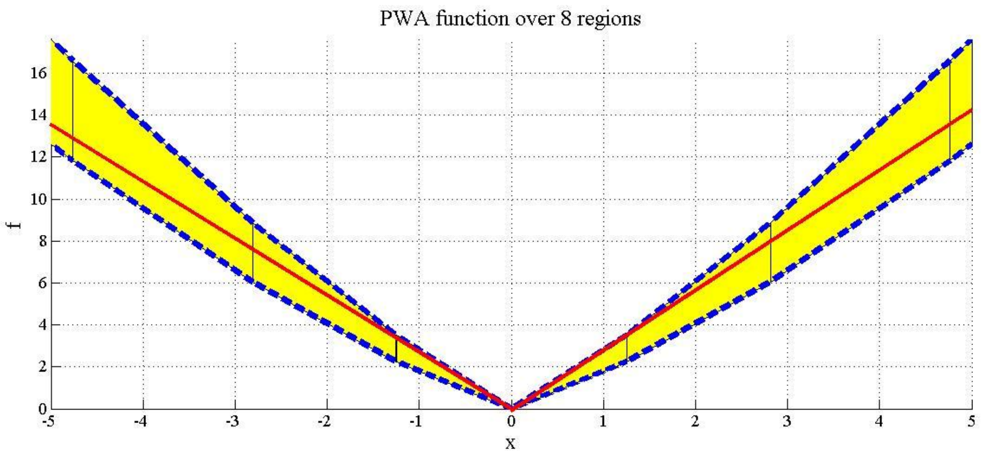

Figure 16.: Reduced complexity 2

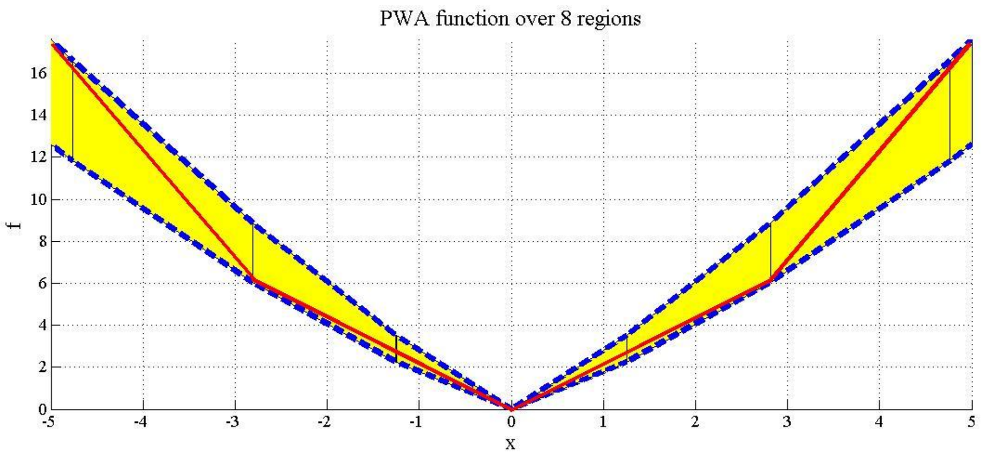

Figure 17.: Reduced complexity 3

The other method, what I used in this work is more complicated from the mathematical part, but it still works when we have a difficult not only one dimensional optimization problem.

In this method Matlab, Yalmip and MPT Toolbox was used. The beginning is the same like in the first approximation method.

Let have a convex objective function, which was divided to finite number of regions. Every region has its own parameter, not only for the objective function for the control law too. Usually in real life the form of the objective function is symmetrical. If we recognize that fact we are able to formulate the optimization problem like a symmetrical problem, so our problem will be easier for the solver and for the computer too. The solvers which I used during this work don’t work in the symmetrical principle. It doesn’t have to be bad, because if we want we can compare which solution is better for us, and choose the better one. Than get the symmetrical values of the better one.

At first time I would like to explain the mathematical part of the optimization.

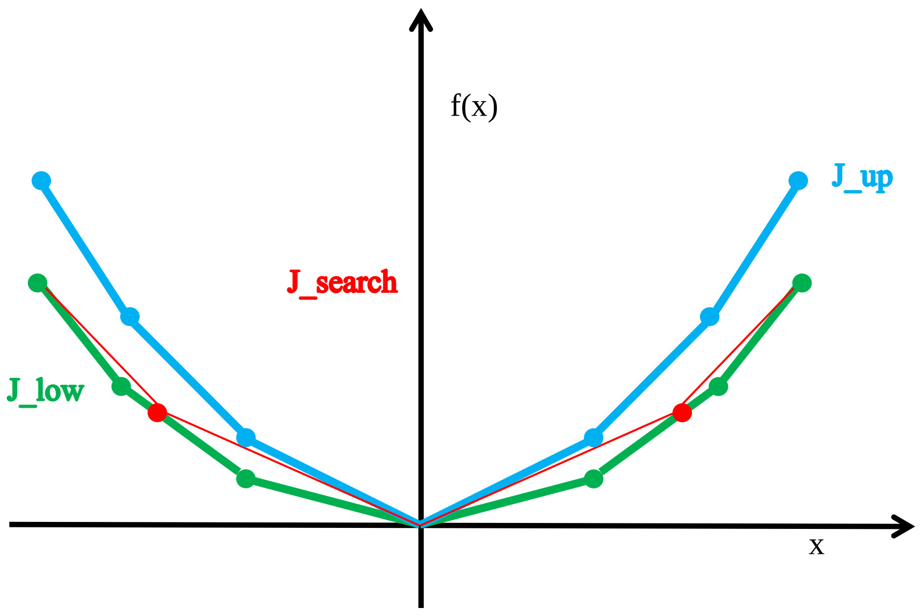

Figure 18.: J functions

The main goal is to find the red J_search function, which must be between the functions J_up and J_low. These functions were created in Matlab by the function prepare_data. The variables J_low and J_up are cell arrays. They contain the following variables: R, C and D.

The variable R represents the number and the characteristic of the regions. By the function extreme we are able to get the size and the vertexes of the regions.

The number of regions:

```
n_R = length(Jlow.R)
```

The size of the regions:

```
extreme(Jlow.R(i))
```

As I already mentioned before there is an ability to divide the whole optimization into two part. Then is enough to calculate from the minimum independent value to the zero. If we would like to do this, we can save time, our algorithm will work faster, because it must work with less optimization variables, and less constraints. The function prepare_data doesn’t create the regions in this sequence what we need. We need to sort these data. This is the way how to do it in Matlab. By the function extreme I got all the extreme of the function. I saved them to the variables, used a sort and after comparison the sorted values with the original values and save it information like the position. After that I had the information of the position I was able to sort the original data by this information of position. Now is enough to say that work only with the regions which are on the left side of the coordinate origin.

```
MIN=[];MAX=[];

for i=1:n_R
MIN=[MIN min(extreme(Jlow.R(i)))];
MAX=[MAX max(extreme(Jlow.R(i)))];
end

%% ranking of regions
TEMP = [];

for i=1:n_R
TEMP= [TEMP extreme(Jlow.R(i))];
end

TEMP2=[];
```

```
for i=1:size(TEMP,2)
    TEMP2(1,i)=max(extreme(Jlow.R(i)));
    TEMP2(2,i)=min(extreme(Jlow.R(i)));
End

SORT=sort(TEMP2(2,:));

POSITION=[];
for i=1:n_R
POSITION=[POSITION find(SORT(i)==TEMP2(2,:))];
end


for i=1:n_R
    C_low{i}=Jlow.C{POSITION(i)};
    D_low{i}=Jlow.D{POSITION(i)};
    C_up{i}=Jup.C{POSITION(i)};
    D_up{i}=Jup.D{POSITION(i)};
    Regions{i}=extreme(Jlow.R(POSITION(i)));
end
```

These new variables (C_low, D_low, C_up, D_ up) include the original data but in form which is better for us. After this we can begin with the formulation of the optimization problem. The first thing what we need to do is get some approximation which are in the allowed space. The way how I formulated this optimization problem is the following:

We know that every points of the approximation must be under the upper function J_up and must have bigger value than the lower function J_low.

$$\forall x \qquad J_{low}(x) \leq J_{search}(x) \leq J_{up}(x) \tag{38}$$

The first part of this inequality is easy to guarantee. If we need to guarantee that all the values of the approximated function will under the upper function we can formulated it in this form:

$$\forall i, \forall\, x \in R_i \colon \max(\alpha_k^T x + \beta_k) \leq c_{V,i}^T x + d_{V,i} \tag{39}$$

This equation says that the maximum value of the function from the approximation have a lower value than the upper limit (J_up). Index *k* is number of the functions which we need for the good approximation, index *i* is index for the regions, and *V* is the vertexes of the regions. This equation can be transformed into the easier form. The inequalities say that the maximal value must be lower than some other kind of value, and this is the same when we say it must be bigger than any value which can take the function. If we realize that we can rewrite the function before:

$$\forall i, \forall\, x \in R_i\colon\ \alpha_k^T x + \beta_k \le c_{V,i}^T x + d_{V,i} \tag{40}$$

We used convex optimization so if we need to connect two points it will never have nonconvex function. In the worst case it should be a line between these points.

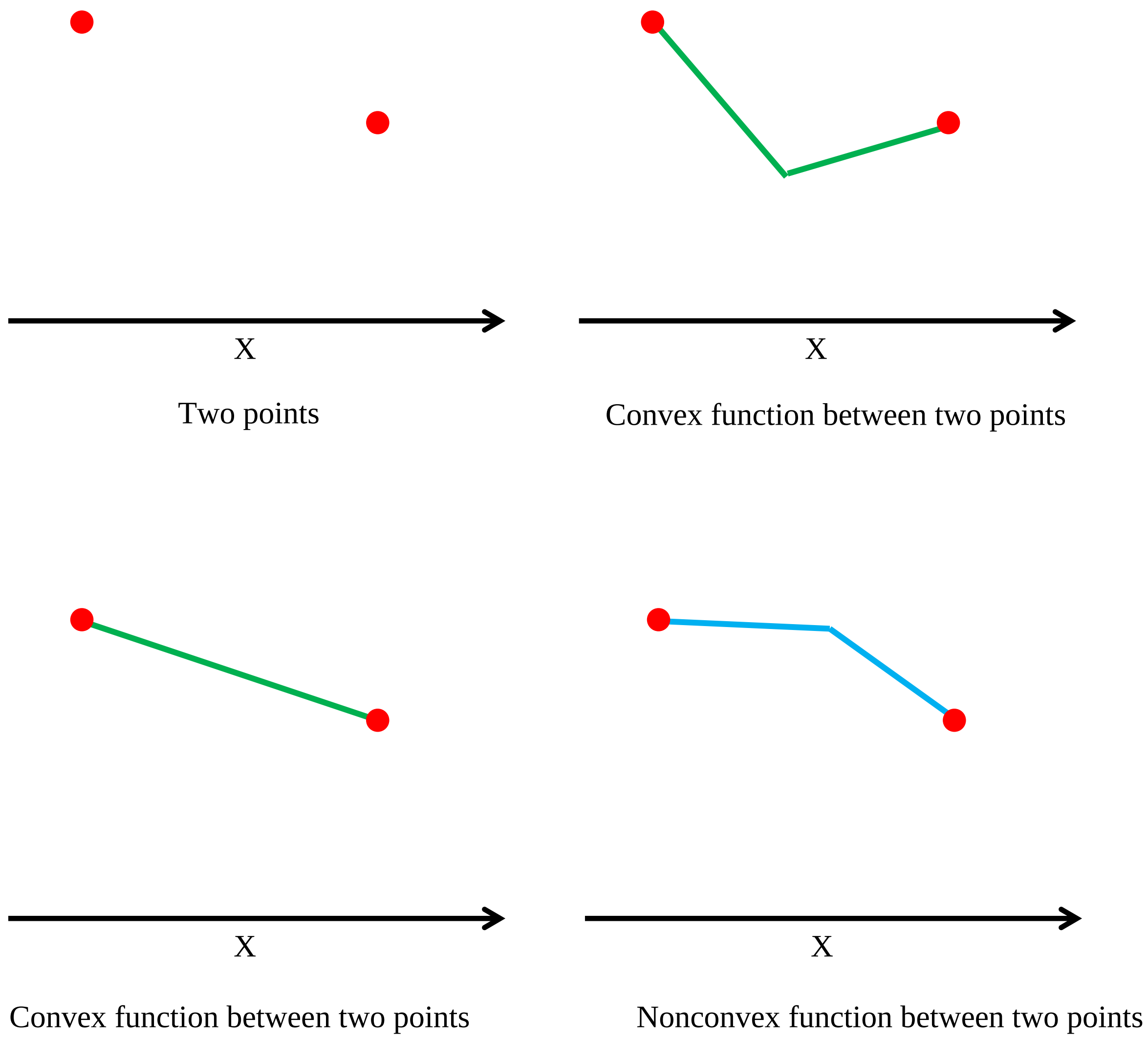
X
Two points
X
Convex function between two points
X
Convex function between two points
X
Nonconvex function between two points

Figure 19.: Convex and nonconvex functions between 2 points

The mathematical definition for the convex function, as I mentioned in the section 1.5 Optimization, can be expressed by the following inequation:

$$f(\theta x - (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \tag{41}$$

$$\theta \epsilon [0,1] \tag{42}$$

From the figure 18. convex and nonconvex functions between 2 points and from this equation we are able to reduce the complexity of this problem. We don’t have to calculate and control every point on the connector between 2 points. It is enough to control only in the boundaries of the regions if the constraints are valid. This is the way how to ensure that the approximation will under the upper function.

The next task is it to ensure that the approximation will over the lower function. The mathematical formulation is the following:

$$\forall x \qquad J_{low}(x) \leq J_{search}(x) \tag{43}$$

$$\forall i, \forall\, x \in R_i\colon \max(\alpha_k^T x + \beta_k) \geq c_{V,i}^T x + d_{V,i} \tag{44}$$

Bound the function from below is harder than from above. When we bond the function from above we need to check only the vertexes of the regions. In case if we have a one dimensional problem there are only 2 points. As I mentioned before we work with convex functions, and if we realize that how is possible to connect two points, than we should realize that the condition which says the approximation must be above the lower function don’t have to be complied in this case if we investigate only the vertexes of the regions. The only way how to solve the problem was create the algorithm which can certificate that the desired function will above the lower function. So the control algorithm works in the next way:

1. the first part of algorithm will declare what kind of system do we have
2. tries to find an easier form of the objective function, by calculating new α and β

3. is it necessary to certificate if these new α and β satisfies conditions
4. if yes we have a solution
5. if no than the point where the system doesn't satisfies is saved
6. everything starts from the point 1., the only difference is that than the algorithm will calculate with this point which was saved

On the following figures are the objective functions of unstable system which are obtained from the control algorithm. The only difference between the stable and unstable system is on the state-space matrix A, everything other are the same. In case of unstable system the value of the A=1,1.

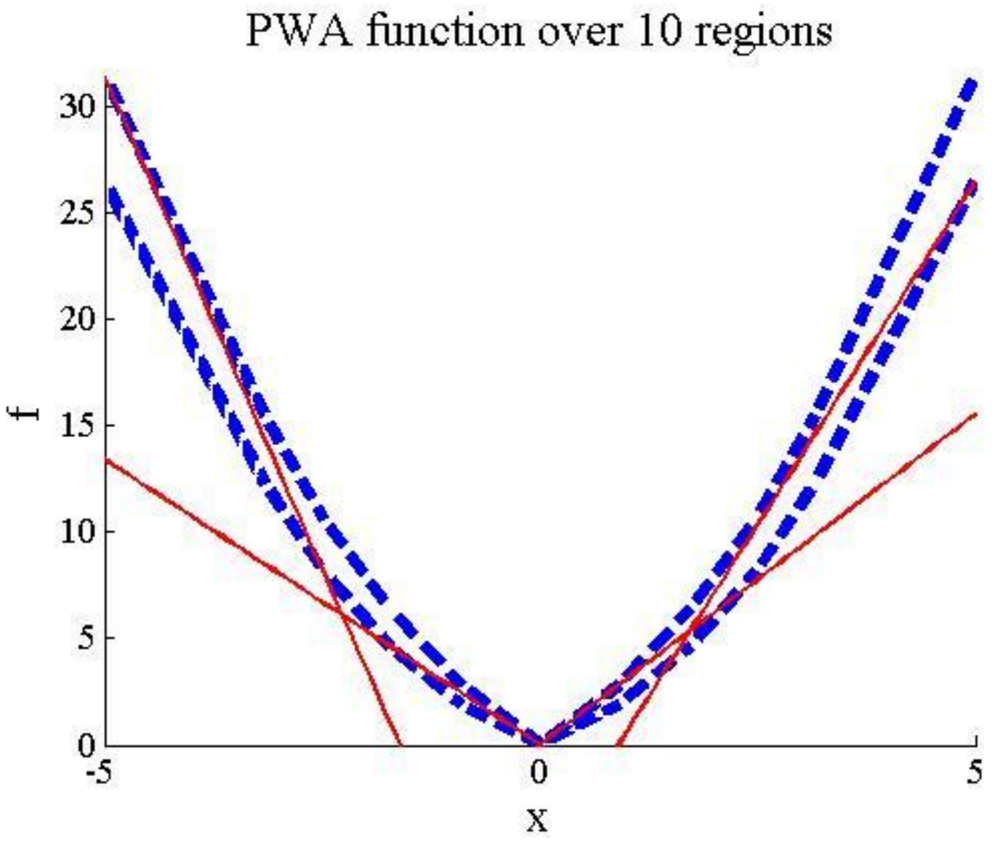

Figure 20.: Objective function from algorithm 1

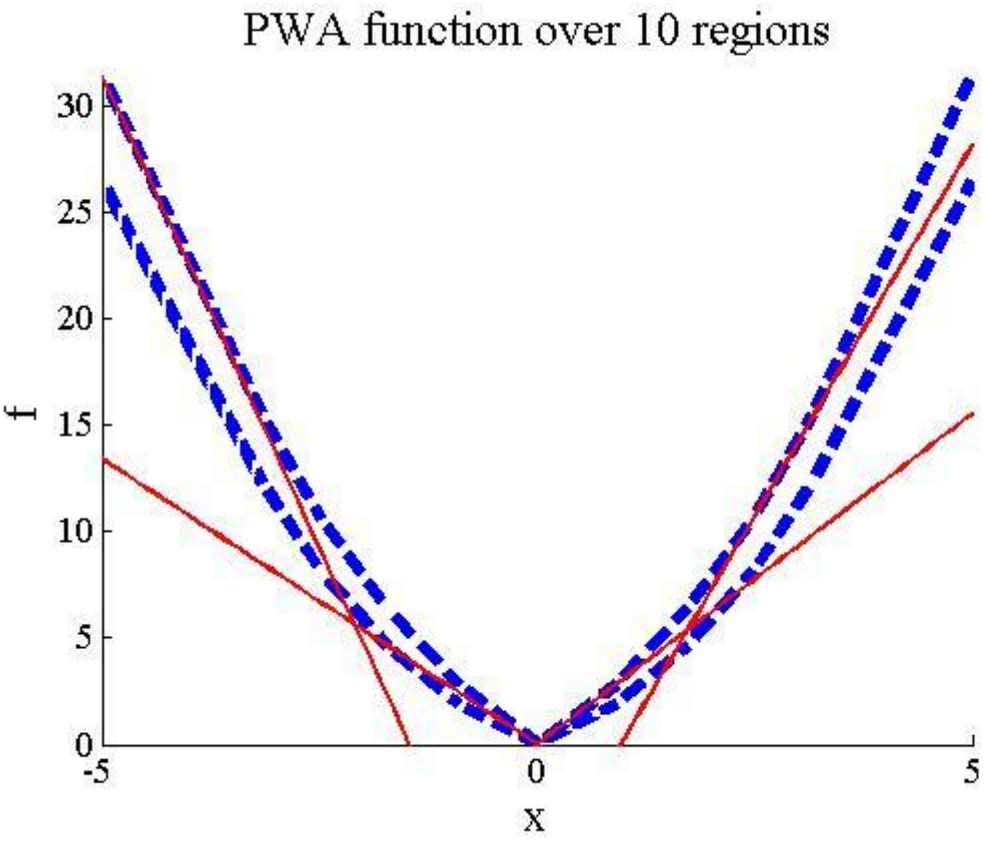

Figure 21.: Objective function from algorithm 2

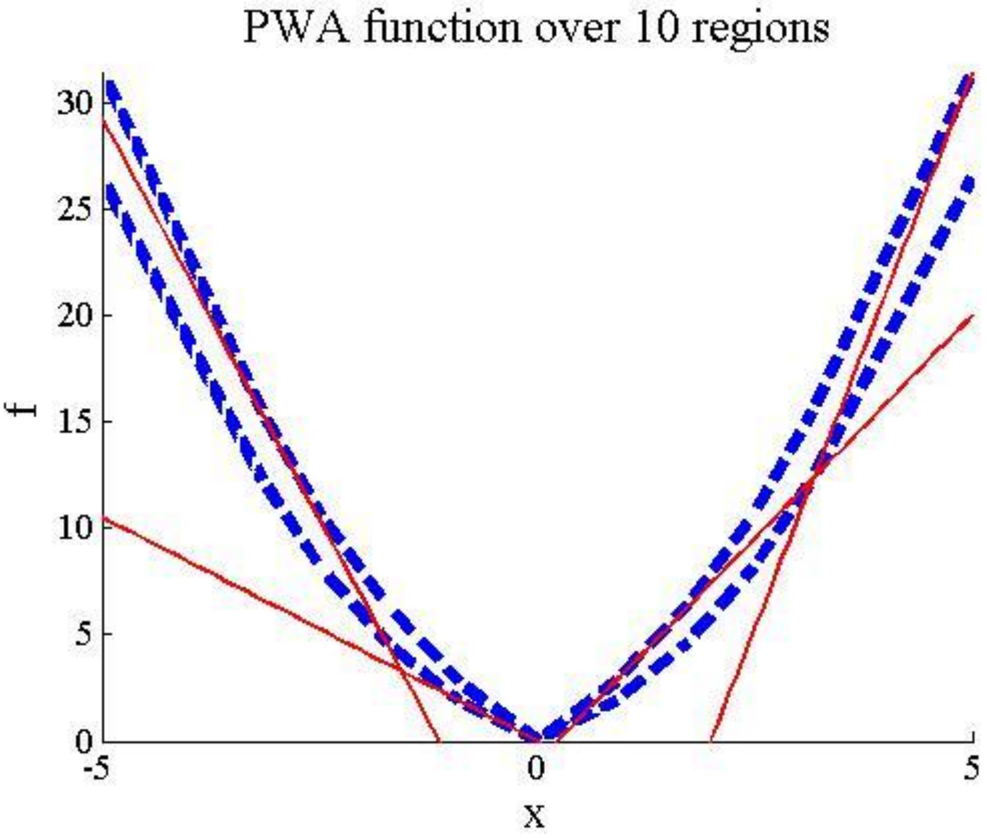

Figure 22.: Objective function from the algorithm 3

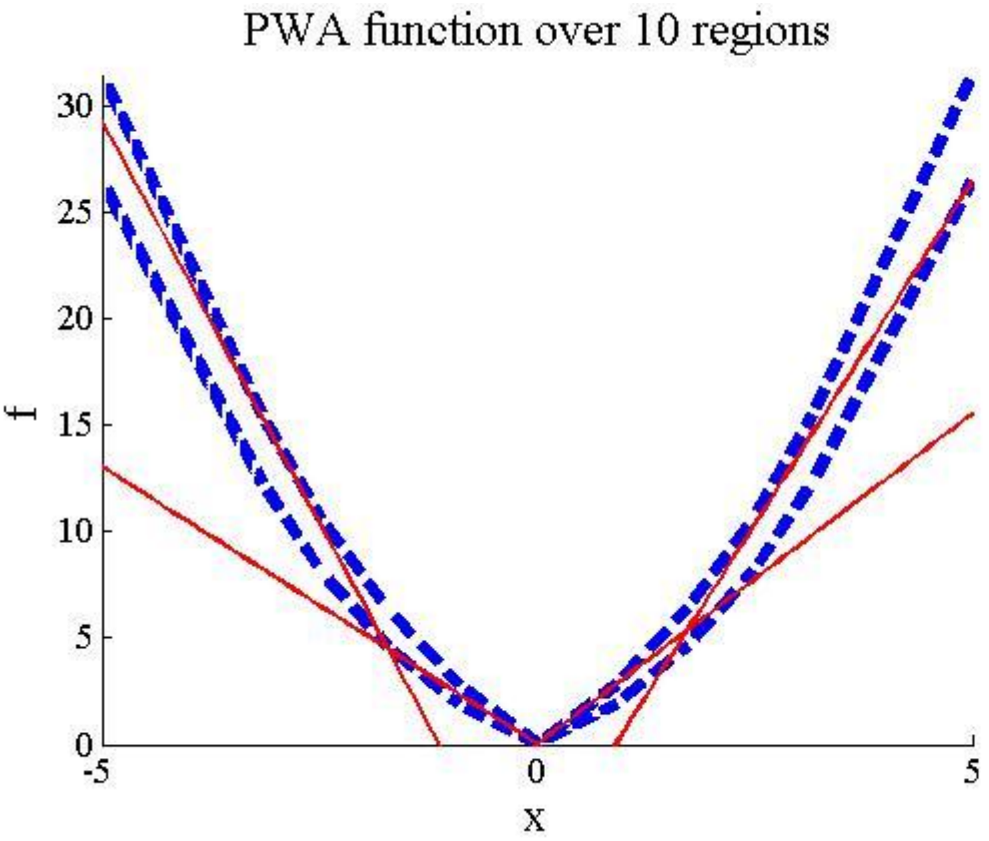

Figure 23.: Objective function from algorithm 4

As you can see the control algorithm needs 4 steps to find the correct values of the variables α and β. The main goal of the certification algorithm is to reduce the distance and find the point where this distance has the maximal value. The algorithm consists of two loops, where the first one is responsible for calculating the variables α and β, and the second one controlled if this parameter satisfies all the conditions. As you can see on the figure 19-22 during the optimization I used all the regions. This is the reason why these graphs aren't symmetrical.

2.2.2. Control law

Now when we have an easier form of the objective function we can approach the control law construction. The control law has the following form:

$$u_{suboptimal\ contol}(x) = \begin{cases} {\alpha_{control}}^T{}_1 x + \beta_{control_1} & if\ x \in R_1 \quad (45) \\ {\alpha_{control}}^T{}_2 x + \beta_{control_2} & if\ x \in R_2 \quad (46) \\ {\alpha_{control}}^T{}_i x + \beta_{control_i} & if\ x \in R_i \quad (47) \end{cases}$$

The new suboptimal input can be easily calculated. We have all the necessary information what we need. In the beginning we need to find the new boundaries for the new objective function, the vertexes of the new regions. I solve this problem in the following way:

We know that each part of the objective function, every single region can be described by the affine function.

$$y_i = \alpha_i x + \beta_i \tag{48}$$

Where α is the slope and β is the affine term. We know that 2 affine functions have one common point and this point is the boundary between the regions. The way how I calculate this point is the following:

$$y_1 = \alpha_1 x + \beta_1 \tag{49}$$

$$y_2 = \alpha_2 x + \beta_2 \tag{50}$$

We know that the value of y_1 and y_2 is the same in one concrete x.

$$y_1 = y_2 \tag{51}$$

$$\alpha_1 x + \beta_1 = \alpha_2 x + \beta_2 \tag{52}$$

$$x = \frac{\beta_2 - \beta_1}{\alpha_1 - \alpha_2} \tag{53}$$

We know the new boundaries of the regions, the vertexes of the regions. The next what we need is to calculate the optimal input in these special points. The general form of the calculation is the following:

$$J(x) = min \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \tag{54}$$

$$s.t.: x_{k+1} = Ax_k + Bu_k \tag{55}$$

$$x_k \in X \tag{56}$$

$$u_k \in U \tag{57}$$

This optimization calculates the optimal input to the system taking into account the constraints and the dynamic behavior of the system. The new value of the state is calculated from the equation of the state equation. There are constraints on the state and inputs too, so the optimization must respect these restrictions.

$$\tilde{u}(x) = U(V_i)\begin{bmatrix} V_i \\ 1 \end{bmatrix}^{-1}\begin{pmatrix} x \\ 1 \end{pmatrix} \tag{58}$$

V_i is the vertex of the regions

U(V_i) is the optimal input in the vertex

x is the actual state

The graph of the control law should have this form:

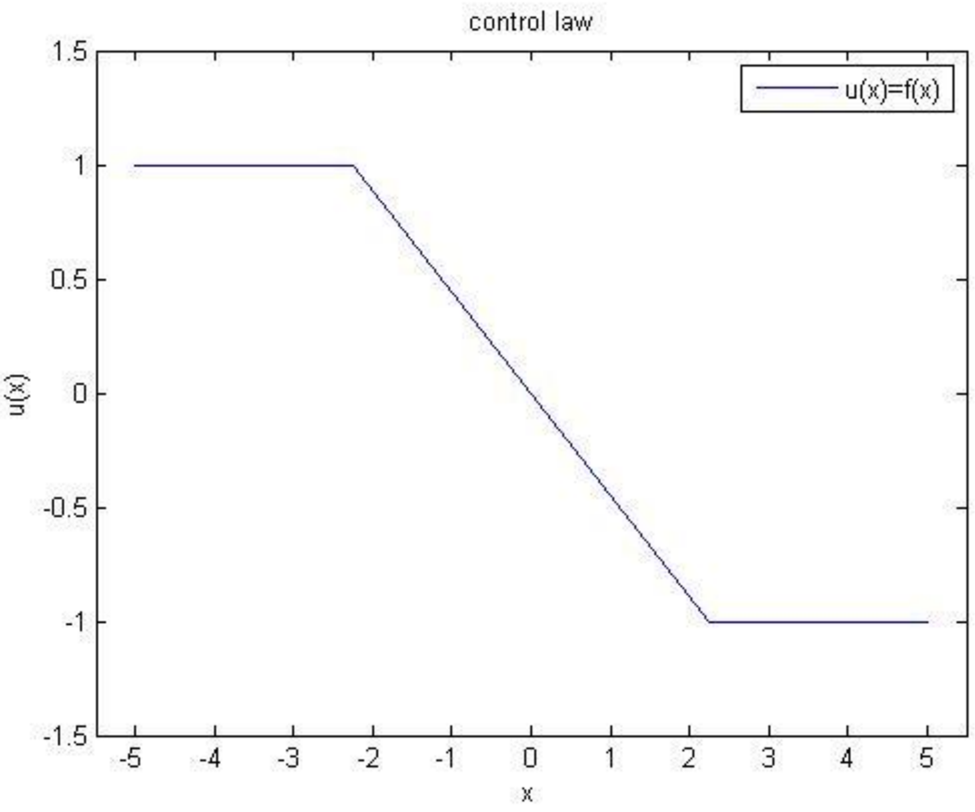

Figure 24.: New control law

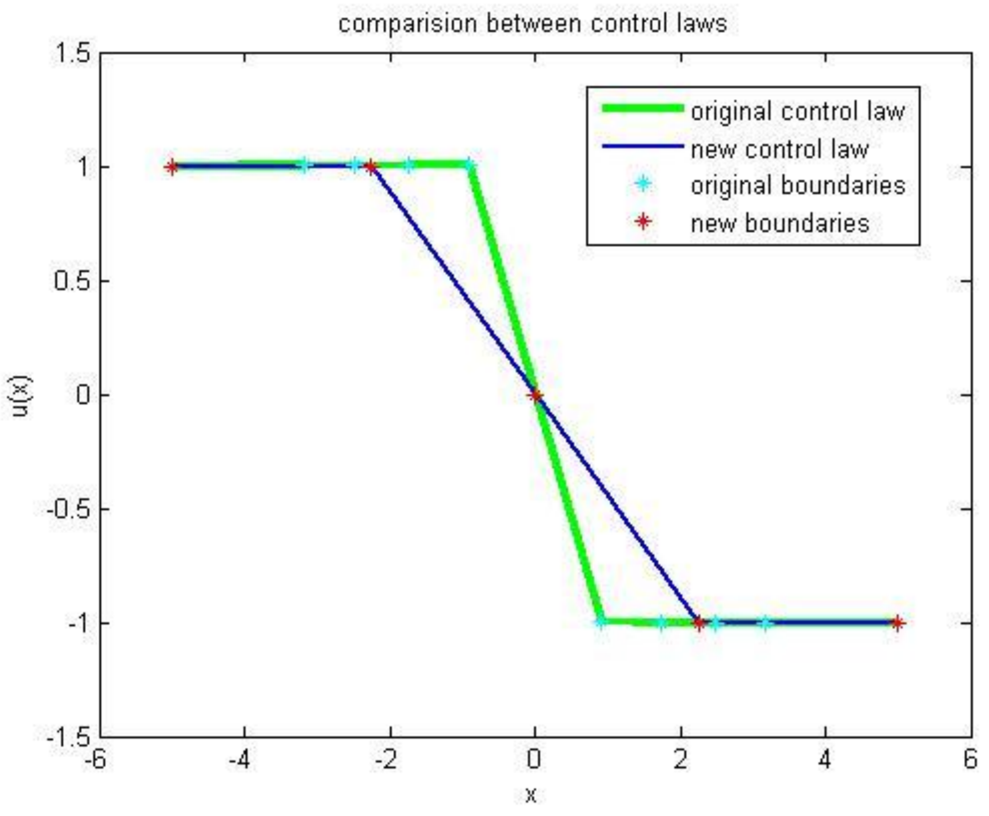

Figure 25.: Comparision between control laws

After I had this control law I was able to control my system. The control law consists of 4 regions. On the figure of the control law it seems like there are only 3 regions. The second and the third regions have similar absolute values. At first I tested the original explicit controller (blue line) and compared with the new explicit controller (green line).

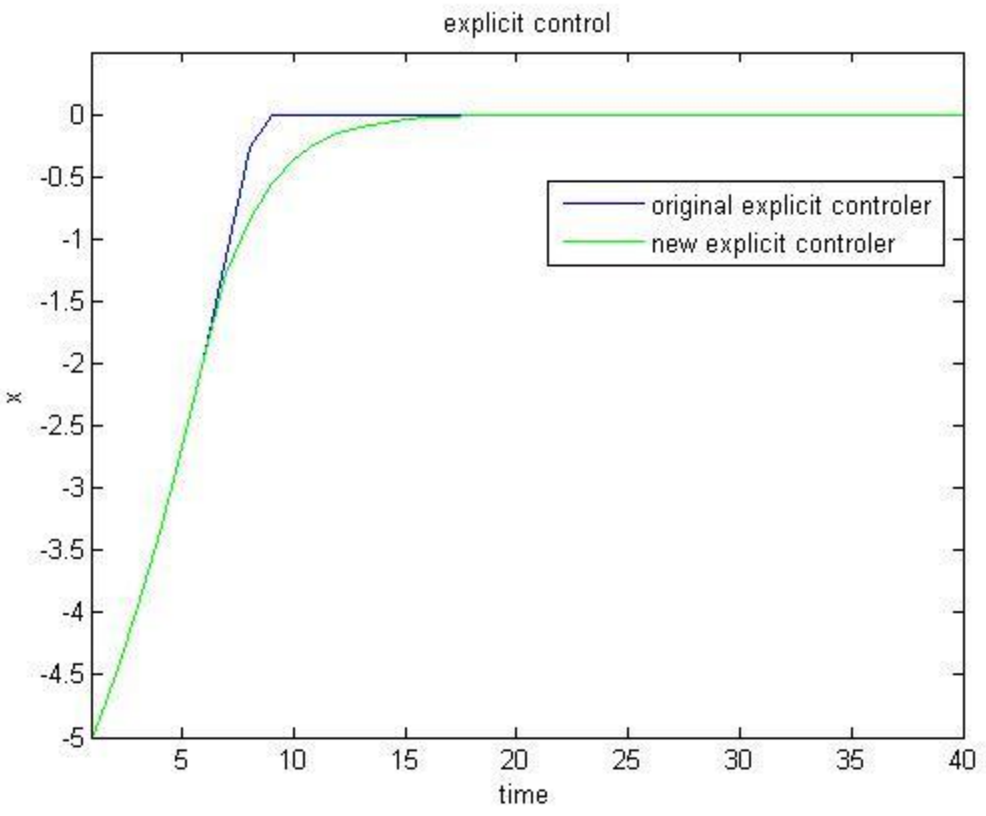

Figure 26.: Control by original and new controller

As you can see on the graph of the control both controllers are able to control the chosen system. The original controller is able to get system faster to the steady state than the easier new controller. The difference between these controllers isn't huge, so if we can't implement the original controller, which has 10 regions so it needs space in the memory for the 20 variables, we can implement the new controller which has only 4 regions and need space only for 8 variables. In this case the easier form of controller reduces the required space in memory by 60%. This is the way how to reduce the cost of the controller by eliminating the number of parameters thanks to eliminating the regions.

3. Conclusion

In this work I studied the problem of predictive control. The aim goal was to search if exist any possibilities to reduce the cost of the explicit predictive control. The work is divided into few sections. In the first section I showed the basic difference between the predictive control and the other kind of control, which are used in the industry. In the next section I explained the difference between the online and the explicit predictive control, and how the method how can be reduced the complexity of the look up table, which is the basic of the explicit predictive control. I created a control algorithm in Matlab, which can find an easier form of the control law.

4. Resumé

Úvod

Prediktívne riadenie je jednou z moderných metód v riadení procesov. Prvé myšlienky boli publikované už v sedemdesiatych rokoch minulého storočia, ale kvôli nedostatočnej výpočtovej kapacity vtedajších počítačov implementácia v priemysle nebola možná. Prvé úspechy prišli až deväťdesiatych rokoch minulého storočia, kedy sa podarilo implementovať na reálnom zariadení. Prvé odvetvie v priemysle ktorom sa ujal tento princíp riadenia bolo práve petrochemické. Veľké časové konštanty a pomalá dynamika riadených procesoch umožnila úspešnú implementáciu. Časom ako sa výpočtový výkon počítačov sa zväčšoval vznikol dostatočný výpočtový základ aj pre riadenie procesov s rýchlejšou dynamikou. Takto sa tento moderný princíp riadenie sa ujal napríklad v mechanike alebo v robotike. Táto metóda môže byť použitá všade kde je potrebné zohľadniť ohraničenia pri výpočte optimálneho akčného zásahu. Najvšeobecnejšia myšlienka optimalizácie má nasledovnú formuláciu: vyťažiť maximum (teplo, výnos, ...) pri minimálnom vstupe (energia, práca, ...).

Základná myšlienka prediktívneho riadenia

Výhodu prediktívneho riadenia, oproti ostatným prístupom je, že pri výpočte optimálneho akčného zásahu je schopný pracovať s ohraničeniami a to vo forme rovností a nerovností. Túto schopnosť nemá v sebe PID regulátor ale ani regulátor LQR. Celá myšlienka je založená na výpočte predikcií. Predikcie sú informácie o budúcom správaní sa systému, pri poznaní dynamiky, ohraničení a počiatočnej podmienky systému. Čím je náš matematický model systému presnejší tým lepšie vieme odhadnúť správanie sa systému v budúcnosti. Kvôli tomu aby sa zabezpečila stabilita systému prediktívne riadenie sa implementuje v uzavretej slučke. Pri otvorenej slučke sa vygenerovaná sekvencia akčných zásahov implementuje celá. Čo môže spôsobiť, že regulátor nebude môcť reagovať na nepredvídateľné situácie. To je dôvod prečo sa používa uzavretá slučka, v ktorom v každom perióde vzorkovania sa vypočíta sekvencia akčných zásahov, ale do systému sa implementuje iba prvá hodnota akčného zásahu. Tie ostatné hodnoty sa nepoužijú na riadenie systému. Na prvý pohľad to môže vyzerať ako nezmysel, lebo zaťažujeme počítač s riešením zložitého optimalizačného problému, a po úspešnom riešení použijeme iba prvú hodnotu riešenia. Tie ostatné hodnoty slúžia na výpočet predikcii v algoritme. Našim cieľom je čo najviac dovidieť do budúcnosti, tým pádom budeme vyžadovať čo najväčší predikčný horizont, ale na druhej strane sme obmedzený výpočtovým výkonom. Čím je náš problém zložitejší, väčší tým silnejší procesor potrebujeme alebo viac času na výpočet. Musíme vedieť garantovať, že optimalizácia sa vypočíta v každom perióde vzorkovania za kratší čas ako je samotné vzorkovanie. V prípade ak by sa to nepodarilo zabezpečiť, mohli by sme prísť o informácie zo systému.

Metódy prediktívneho riadenia

Existujú dve rozličné prístupy prediktívneho riadenia. Jedna je ON-LINE metóda a druhá je EXPLICITná metóda. Obe metódy môžu byť použité na riadenie systémov približne s rovnakou kvalitou riadenia. Rozdiel medzi nimi spočíva v potrebnom čase na výpočet optimálneho akčného zásahu. Prvá metóda sa používa vtedy keď máme dostatočne dlhý čas na výpočet optimalizácie. V tomto prípade sa optimalizácia vykoná v každom kroku. V prípade, keď máme za úlohu riadiť systém s veľmi rýchlou dynamikou ako napríklad mechanické kyvadlá, kde je čas vzorkovania rádovo v milisekundách on-line metóda neprichádza do úvahy. Explicitné prediktívne riadenie je vhodné na riadenie takýchto systémov s rýchlou dynamikou. Najväčší rozdiel medzi dvoma metódami je, že explicitnom riadení sa optimalizácia vykonáva iba raz, ešte pred samotným riadením. Zvykne sa to volať aj off-line metóda lebo optimalizácia je vykonaná ešte pred začatím riadenia. To čo získame z optimalizácie je vyhľadávacia tabuľka, ktorá pozostáva z koeficientov účelovej funkcie a koeficientov zákona riadenia. Tato tabuľka sa rozdeliť do regiónov a každý región obsahuje už spomenuté parametre. Na základe jednoduchých matematických operácii sa dá zistiť v ktorom stave sa systém nachádza, to znamená že vieme, ktorý región je aktívny. V danom okamihu iba jeden región môže byť aktívny. Keď poznáme index aktívneho regiónu vieme aj koeficienty zákona riadenia a vieme vypočítať potrebný akčný zásah. V práci som sa zaoberal iba s lineárnymi funkciami. Aj účelová funkcia bola po častiach lineárna aj zákon riadenia bol lineárny. Táto metóda sa dá použiť iba v prípade, keď máme menej ako 5 stavov.

Redukcia zložitosti

V prípade, že optimalizácia na základe našich požiadaviek vypočíta parametre regiónov, ale množstvo regiónov je tak veľké, že fyzikálne sa to nedá implementovať na riadenie lebo pamäť regulátora je silne limitovaná. V takom prípade je potrebné zredukovať potrebnú pamäť. V prípade, že by sme vynechali iba také množstvo regiónov, ktoré je potrebné na to aby sa vyhľadávacia tabuľka dala nahrať na hardware, môžeme prísť v prvom rade o optimálnosť v druhom rade aj o stabilitu. Toto riešenie nie je akceptovateľné. Redukcia zložitosti sa dá dosiahnuť aj so zjednodušením účelovej funkcie. V prípade ak sme schopný nájsť jednoduchší tvar účelovej funkcie aj za tú cenu, že naše riešenie bude iba suboptimálne, ale dokáže stabilizovať systém, je stále lepšie riešenie ako žiadne riešenie. V práci som opísal postup ako získať také jednoduchšie riešenie.

Záver

Diplomová práca bola zameraná na prediktívne riadenie. V rámci práce som porovnal prediktívne riadenie so štandardnými formami riadenia. Vysvetlil základnú myšlienku a porovnal explicitnú metódu a on-line metódu riadenia. V práci je opísaný spôsob na redukciu zložitosti explicitného prediktívneho riadenia. Na konci práce som porovnal pôvodný aj zjednodušený regulátor. Výsledok som spracoval graficky.

5. Bibliography

M. Kvasnica. Habilatačná práca, Rýchle a pamäťovo efektívne prediktívne riadenie hydridných systémov, 2011.

M. Kvasnica. Prezentation from the subject MPC

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

Jan M. Maciejowski. Predictive Control with Constraints. Prentice Hall, 2002.

J. Lo ̈fberg. YALMIP, 2004. Available from http://users.isy. liu.se/johanl/yalmip/.

M. Kvasnica, P. Grieder, and M. Baotic ́. Multi-Parametric Toolbox (MPT), 2004. Available from http://control.ee.ethz.ch/~ ̃mpt/.

S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004.