TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

EXPLICIT PREDICTIVE CONTROL IN APPLICATION

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 3906T001 – Mechatronics

Author: **Kirill Lastochkin**
Supervisor: Ing. Lukáš Hubka, Ph.D.

Liberec 2015

Declaration

I hereby certify that I have been informed the Act 121/2000, the Copyright Act of the Czech Republic, namely § 60 - Schoolwork, applies to my master thesis in full scope.

I acknowledge that the Technical University of Liberec (TUL) does not infringe my copyrights by using my master thesis for TUL's internal purposes.

I am aware of my obligation to inform TUL on having used or licensed to use my master thesis; in such a case, TUL may require compensation of costs spent on creating the work at up to their actual amount.

I have written my master thesis myself using literature listed therein and consulting it with my thesis supervisor and my tutor.

Concurrently I confirm that the printed version of my master thesis is coincident with an electronic version, inserted into the IS STAG.

Date:

Signature:

ABSTRACT

In the thesis the basic principles of Model Predictive Control and its explicit form are considered. The main goal of this work was to implement explicit MPC control strategy for control of the real plant and realize it on the basis of microcontroller. There were four important steps from theory to realization. The first step was to learn the information about the topic in the field of Model Predictive Control. The second step was to make an identification of the controlled plant. The third step contained explicit predictive controller synthesis with the help of additional toolbox for Matlab, called Hybrid Toolbox. Also this step included exploration of parameter influence and comparison with popular control methods. At the last step explicit predictive controller was realized on the microcontroller STM32F100RBT6B.

KEYWORDS

Explicit MPC, regulation, Hybrid Toolbox, DC motor, STM32F100RBT6B

Table of content

LIST OF ABBREVIATIONS

MPC	Model Predictive Control
PID	proportional-integral-derivative
MC	microcontroller
$w(t)$	reference signal
$y(t)$	controlled value
$u(t)$	control action
$e(t)$	control error
N_u	control horizon
N	prediction horizon
$w(p+p\|k)$	future reference signal
$y(p+p\|k)$	predicted controlled value
S,Q	weight functions

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

At the moment the most popular method for real process control is PID control. This method is simple in realization and allows fast implementing of the control algorithm. But PID controller has its own limitations, and it is not possible to make optimal control for some processes. In complicated systems, which states are dependent on many factors and constraints, it is very hard realize control with PID, sometimes it is impossible. For effective control in such processes, modern methods are appeared.

One of these methods is explicit model predictive control, based on model predictive control. In the work this perspective control method is considered, compared with popular control methods and implemented for control of real system “DC motor – generator”. Such systems are used in the design of robotic arms, in amortization systems computation, and also for simulation of processes with low-frequency oscillations in axes, for example, flexibility accounting in high building’s elevators.

Control is realized on the basis of microcontroller. At the moment the use of microcontrollers is massive, in the last years their operating characteristics became significantly better. It allows using them for realization of complicated control algorithms.

1 DESCRIPTION OF THE CONTROL METHODS AND ESTIMATING TECHNOLOGY

1.1 Model predictive control

The method of model predictive control (MPC) allows making an optimal control in a relatively large amount of real processes and widely uses in industry. Range of application of MPC is wide enough: from simple SISO systems to complex MIMO systems with dozens of inputs and outputs. This control method was developed in the 1960s, as a response to the demands of the petrochemical industry and is actively used for process control on oil rigs and refineries [1]. At the beginning, after appearing, MPC was used for control of a slow process because the computing power needed to ensure optimal control was not enough for fast processes. Another disadvantage of MPC is the requirement to precise knowledge of model. For these reasons, most of processes are currently managed by classical PID controllers. But due to increasing of performance in microprocessor technology, MPC begins to be applied more often [2].

For now, the computing power of the industrial equipment, usually allows realizing of this control method. Today MPC is used in the chemical industry, power and paper industry [3].

1.1.1 The principles and structure of MPC

An important feature of this control method, which make a difference between it and the traditional, is possibility to consider constraints during designing of the system. Also it is playing a key role in process control. Regulators synthesized using MPC are much more efficient and optimal than obtained with using classical methods of control ones. One of the major disadvantages has been the requirement for precise knowledge of the model, due to the fact that the state-prediction of the plant is based on the processes in the model. However, practice

shows us that it is sufficient for satisfactory control to use a simplified model of the system.

The principle of MPC operation is illustrated on Fig. 1.1. As it is clear from the name of the method, the system tries to predict the future evolution to the so-called horizon of prediction (N_u, N_2). The future is built on the knowledge of the system model. MPC predicts the future values of the output signal on the basis of past input values $u(t)$, the output $y(t)$, states of the plant $x(t)$ and time t.

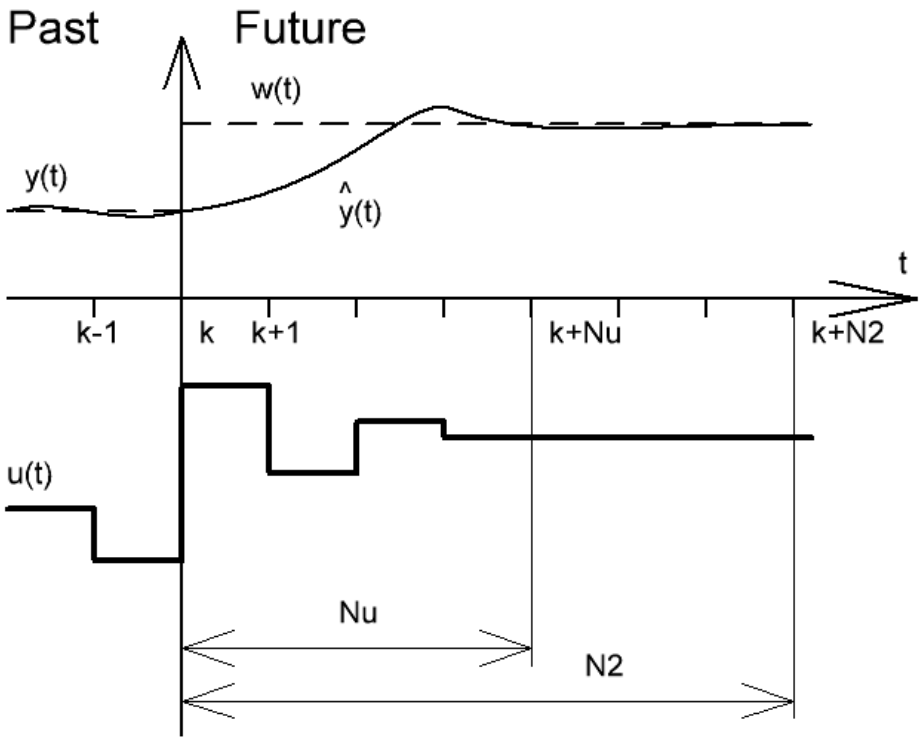

Fig. 1.1: The principle of operation MPC

In the MPC there is control action obtained by solving the optimal control problem for each sample step. The solution of this problem in each step generates a sequence of optimal control actions, but only the first one is applied to the process. On the next sample, the calculation is repeated again on the basis of the shifted time horizon, taking the most current information of the status of the

system as the new initial conditions for optimal control problem. For this reason, sometimes MPC is called “receding horizon” control.

The solution is based on a dynamic model of the process, all restrictions on the inputs and outputs are included. This is expressed by means of a linear or quadratic criterion, so that for a linear predictive model final optimization problem can be showed in a form of quadratic or linear programming. For the same hybrid model the resulting optimal control problem can be represented as a mix of linear and quadratic program. In reality, the linear form isn’t usual solution for this problem due to some solving problems.

The main difference between the MPC and the classical methods of control is that in the second case, all calculations are carried out without including of the controller into the process. For this reason, MPC has been successfully used for the control of processes in production with a dozens of manipulated parameters and constraints on them.

Controlled process is usually described by a system of difference equations

$$\begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = c(x_k, u_k) \end{cases} \tag{1.1}$$

where $x_k \in R^n$ is the state vector and $u_k \in R^m$ is the vector of input actions. For simplicity assume that $f(0,0) = 0$. The sequence of states and control inputs must satisfy the constraints

$$x_k \in X, u_k \in U \tag{1.2}$$

where $X \subseteq R^n$ and $U \subseteq R^m$ are closed sets. Assuming that the control aim is to lead the state of the plant to the target state, MPC solves the problem of conditional regulation as follows. Assuming that a complete measurement of states x is available at the current time t. Then the following optimal control problem of finding a finite horizon is solved as

$$P_N(x): \min_z \sum_{k=0}^{N-1} l(x_k, u_k) + F(x_N) \tag{1.3a}$$

$$x_{k+1} = f(x_k, u_k), k = 0, \dots, N-1 \tag{1.3b}$$

$$u_k \in U, k = 0, \dots, N_u - 1 \tag{1.3c}$$

$$x_k \in X, k = 0, \dots, N-1 \tag{1.3d}$$

$$x_N \in X_N \tag{1.3e}$$

where $z \in R^l$ is the vector of variables optimization $z = \left[u_0' \dots u_{N_u-1}'\right]'$, $l(x_k, u_k)$ and $F(x_N)$ are parameters for measuring performance, their form depends on the type of cost in linear model *(*generally *z* includes command inputs and additional optimization variables) [2]. And the choice of a finite set $X_N \subseteq X$, the final cost *F,* and the final gain *K,* ensures the stability of the closed loop circuit for MPC. At each time step *t,* x_k denotes the predicted state vector at time *t+k,* obtained by applying the input sequence $u_0, \dots, u_{k-1}$ to the model (1.1), beginning from $x_0 = x(0)$. A number *N>0* is the predicted horizon, N_u is the horizon of input action ($1 \leq N_u \leq N$). Since *N* is finite, if *f, l* and *F* are continuous, and also *U* is compact, minimum in equation (1.3a) exists. At each step *t* the solution $P_N(x)$ is found by solving a mathematical program

$$\begin{gathered} \min_z h(z, x) \\ g(z, x) \leq 0, g \in R^q \end{gathered} \tag{1.4}$$

obtained from (1.3), obeying the optimal control sequence $z^*(x(t))$, but only the first member is entered into the system (1.1). The form of functions *h*(*z,x*) and *g*(*z,x*) depends on the type of cost in linear model [2].

$$u = u_0^*(x) \tag{1.5}$$

and the optimization problem (1.3) is repeated at step $t+1$ based on the new state of $x(t+1)$ [].

Basic settings of MPC may vary for different cases, depending on the predictive models, efficiency and finite conditions used.

MPC structure is shown in Fig. 1.2.

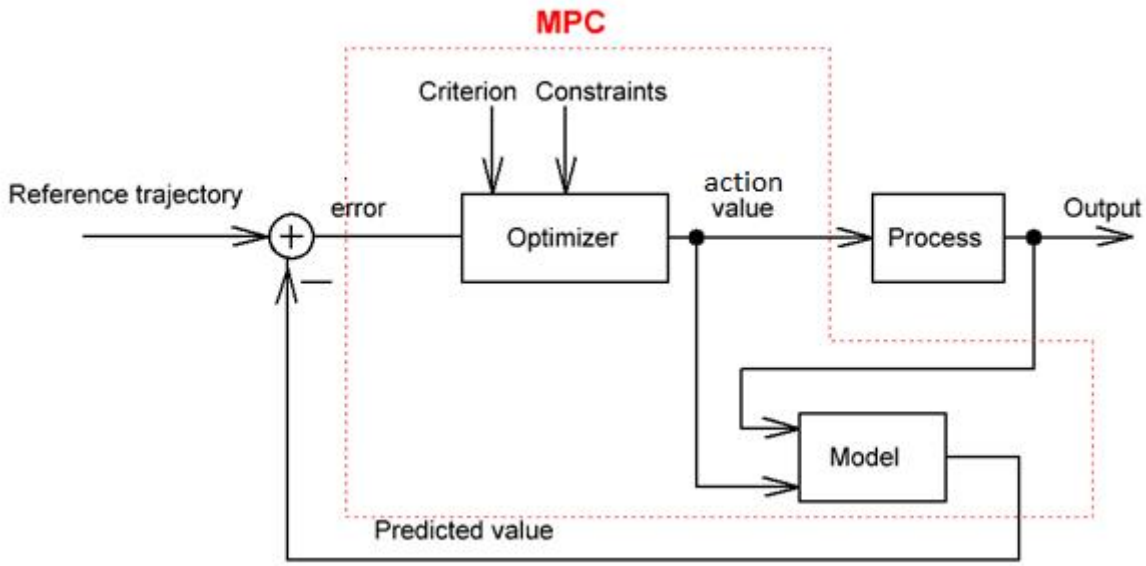

Fig. 1.2: The structure of the MPC

1.1.2 Quadratic criterion and constraints

Usually, a quadratic criterion is applied in the following form

$$J(k) = \sum_{p=N_1}^{N} \big(w(k+p|k) - y(k+p|k)\big)^T \cdot S \cdot \big(w(k+p|k) - y(k+p|k)\big) + \sum_{p=0}^{N_u-1} \Delta u(k+p|k)^T \cdot Q \cdot \Delta u(k+p|k) \tag{1.6}$$

This criterion (1.6) monitors deviations from the desired path and predicts increasing of this value on the base of previous actions. Each member is loaded with an appropriate weight function (S, Q), which influence on the final process dynamics and quality of control.

Member $y(k+p|k)$ predicts output value y in time p+k on the basis of data known at time k. Element N_1 represents the smallest prediction horizon, which is usually equal to one, greater value is used only in systems with a time-delay or with large rise time.

In addition to the criterion for optimization problem can also be given various constraints that have to be considered in the solutions found. These constraints are usually specified in the form of non-linear equations. Often constraints are implemented to the input variables, manipulated variables of state. There are two kinds of constraints: artificial, used in order to achieve nominal process parameters and technological, dependent on the physics of the process. Both of these sorts should be considered when implementing the MPC. Often constraints themselves can vary, but it is preferable to compute the system by setting them to constant values, since the introduction of additional variables in the optimization problem significantly increases the complexity of the solution.

Constraining conditions are divided into two types: hard and soft. Hard conditions are such that the system under any circumstances cannot break them. Such restrictions are often applied to the control action because of its limited resources. Soft limits allow breaking bounds for some time. The consequences of such violations are not serious, but still are undesirable.

1.2 State observer and offset-free tracking

Using of MPC means, that the model of plant ant it current states are known. The first problem is solved by methods of identification before the starting of real process. The second problem should be solved online, and because of it, it is necessary to include into the system state observer.

Also it is usual problem that model isn't same as the plant, and because of it the output of the plant reaches incorrect steady value during the working process. To avoid this problem offset-free tracking method is used.

1.2.1 Offset-free tracking

This method is rather simple and commonly used for predictive controller modifications. The main idea is to measure the error value between reference value and plant output, and after that subtract this error from reference trajectory [4].

In implementation it means, that the additional member $e(k)$ equal to this error is added to state space representation as a new state.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = [c_1\ c_2\ c_3\ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} \tag{1.7}$$

New state doesn't depend on k and is a constant offset, added to the output.

1.2.2 Observer structure description

Generally, system with observer looks like on Fig. 1.3. The main idea of observing is to add a parallel dynamic model with same parameters as the plant to the system. Observer also has an important part – correction unit. It allows making required dynamic to the state estimating. Inputs of observer are: input and output of the plant. Outputs of observer are vector of estimated states and estimated output of the plant [5].

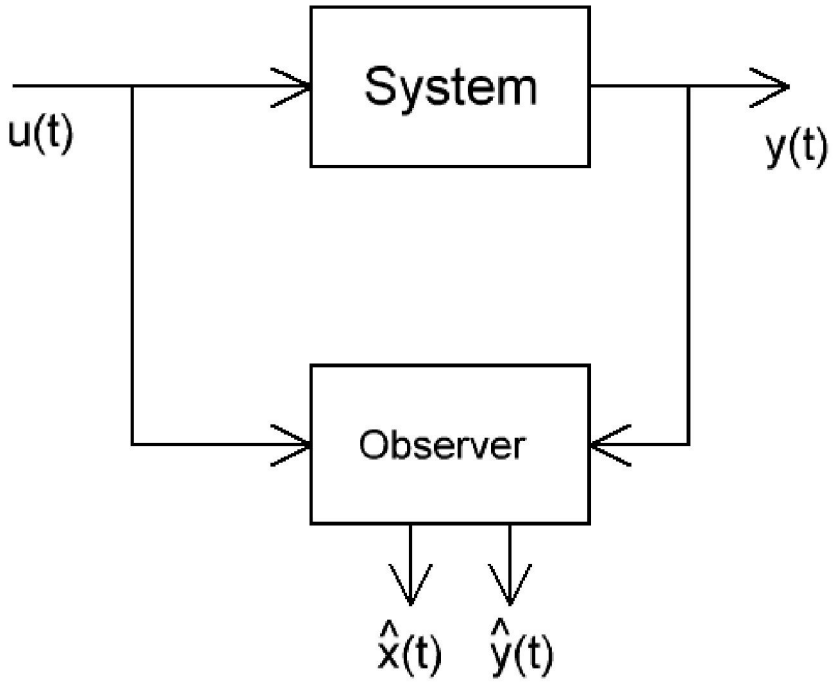

Fig. 1.3: The scheme for estimating of the state of the system

In case of matrix procedure of calculating, its scheme looks like on Fig. 1.4. Matrices A_M, B_M, C_M are equal to the matrices A, B, C of the plant, matrix of correction factors L provides convergence of error $\Delta(k)$ between outputs of estimator and plant to zero. $\bar{x}$ –estimated states vector, y_M - estimated output [5].

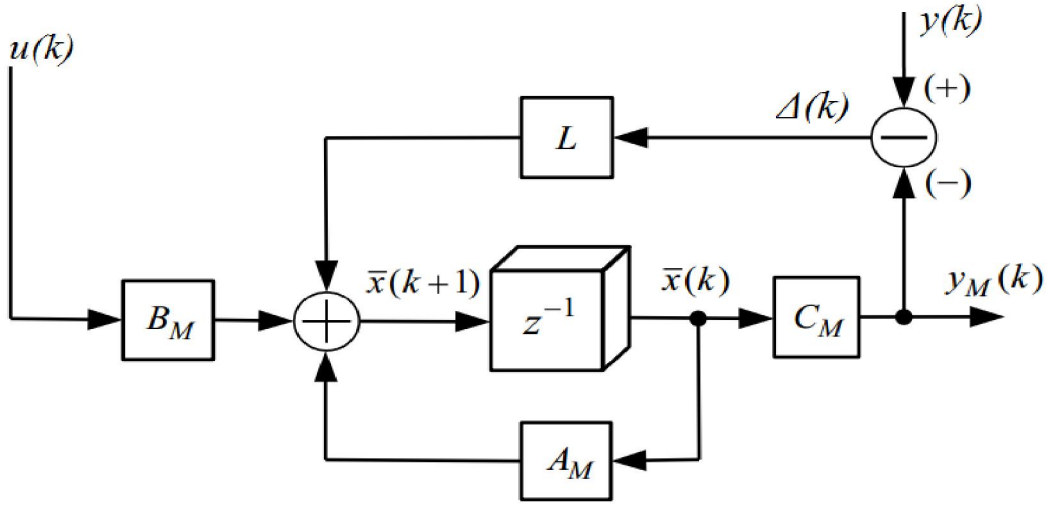

Fig. 1.4: Structure of estimator

1.2.3 Matrix procedure of synthesis

For the describing of the process system of difference equations is used (1.1), and because of it the convenient way to synthesize the observer is matrix approach.

System equations in matrix form

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases} \tag{1.8}$$

Observer equations in matrix form

$$\begin{cases} \bar{x}(k+1) = A_M\bar{x}(k) + B_M u(k) + L\Delta(k) \\ y_M(k) = C_M\bar{x}(k) \\ \Delta(k) = y(k) - y_M(k) \end{cases} \tag{1.9}$$

Let's add new variable for error between states of plant and observer

$$e(k) = x(k) - \bar{x}(k),$$

shift argument for 1 step forward

$$e(k+1) = x(k+1) - \bar{x}(k+1), \tag{1.10}$$

Adding (1.8) and (1.9) to (1.10), obtains

$$e(k+1) = (A - LC)e(k) \tag{1.11}$$

This equation is homogeneous. If observer is stable, *e(k)* goes to zero, $\bar{x}(k)$ to *x*(*k*). From (1.11) derives equation for observer calculation

$$\det[zI - A + LC] = C(z) \tag{1.12}$$

where *C*(*z*) is the desired polynomial of observer.

For plants with 1 channel and transfer function

$$F(z) = \frac{b_{n-1}z^{n-1} + \cdots + b_0}{z^n + a_{n-1}z^{n-1} + \cdots + a_0}$$

procedure of synthesis becomes easier, if introduce the model of the system in transposed observable form:

$$A = \begin{bmatrix} 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & \dots & 0 & -a_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -a_{n-1} \end{bmatrix}, B = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ b_{n-1} \end{bmatrix}, C = [0 \quad 0 \quad \dots \quad 1], L = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ \dots \\ l_{n-1} \end{bmatrix}$$

Similarly as in (1.11), obtains

$$A - LC = \begin{bmatrix} 0 & 0 & \dots & 0 & -a_0 - l_0 \\ 1 & 0 & \dots & 0 & -a_1 - l_1 \\ 0 & 1 & \dots & 0 & -a_2 - l_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -a_{n-1} - l_{n-1} \end{bmatrix} \tag{1.13}$$

Because of form of the matrix *C*, in (1.12) will be counted only last column of matrix (1.13). Therefore, final equation for correction factors looks like

$$l_i = c_i - a_i, i = \overline{0, n-1} \tag{1.14}$$

where c_i are factors of $C(z)$ [5].

After augmenting the model, according to (1.7), the observer should be recalculated. All procedure is defined in detail in [4]. In practical implementation it means, that we should calculate observer of $(n+1)^{\text{th}}$ order.

1.3 Explicit MPC

Model predictive control, as described in the preceding paragraph requires performing an optimization algorithm at each time-step *t* for the current state of the system $x(t)$, to realize the optimal control. This computation is performed in real time, and because of this requirements for computational power of controller become harder. For fast systems solution of the optimization problem (1.4) in a relatively short time becomes difficult, so classic MPC is commonly used as a way to struggle with a slow process. With the evolution of computer technology concept of slow process varies [6].

Additional constraints are used because of possible errors in the calculations, appearing of which could lead to the collapse of the entire system. Extra attention to this issue has to be treated in a safety control system.

Explicit model predictive control is designed to solve this problem on the basis of the MPC method.

1.3.1 Explicit MPC principle

The main difference from the classical MPC is that explicit MPC optimal control problem (1.4) is solved offline for the entire state space of the system $x(t) \in X$, and on the basis of this solution dependence between $u(t)$ and $x(t)$ is described explicitly (1.7), and isn't indirectly determined by optimization procedures.

$$u^* = k(x) \tag{1.14}$$

According to [2], the controller $k(x)$, defined by the equation (1.14) is similar to the classic MPC and the solution of the optimization problem is found in the form of minimizing of the quadratic criterion (1.6). For Explicit MPC, unlike MPC, it is necessary to find a solution for all possible states $x(t)$, not only for the current state of the system. Because of this fact, solutions are based on using of multiparametric quadratic programming [7]. The result of solving optimization problem is expressed in the form (1.8) and is a continuous, piecewise-linear control law.

$$u(x) = \begin{cases} F_1 x + g_1 \; for \; H_1 x \leq k_1 \\ \dots \\ F_M x + g_M \; for \; H_M x \leq k_M \end{cases} \tag{1.15}$$

where H_i and k_i are the parameters that describe a range of the state space $x(t)$ values. These areas are called critical regions; each of them has its own control law, which is expressed by a continuous linear function. In operation, the plant

goes into one of these regions, and special algorithm selects the appropriate control law [7], [8].

The process of creating regions is as follows. At first, the problem of optimization for the selected initial conditions is solved, and then some area around the critical region built. After that algorithm explores the entire region and gradually creates another new critical region. According to the results of computations is created a table of linear functions and their respective regions.

Search for particular regions is carried out in real time, and for this procedure it is necessary to know the state vector $x(t)$, which normally is immeasurable. Therefore, it is required to estimate these states. Thus, the intelligent controller has to be complemented by a state observer (Fig. 1.3). It allows determining of the current state of the system, and ensures the timely switching between control actions during the transition between the critical regions.

Although this process is intuitive clear, and there are sufficient numbers of affordable methods to estimate the states of the plant, one important limitation is complexity and volume of data tables, depending on the number of created critical regions. With the growth of usable memory for storing tables, requirements to computational power become harder. It means, that controller has to work quickly to correct selection of the current region.

1.3.2 Reducing complexity of explicit MPC

Explicit MPC complexity, as mentioned above, depends on the number of regions M (1.8), found in the process of solving the optimization problem. Their number increases significantly due to various constraints q (exponential in the worst case), introduced in the control problem (1.3), also their number influences on the dimension of the state vector $x(t)$.

One way to reduce complexity is to decrease the prediction horizon, which leads to a reduction of q. In practical applications, the explicit MPC is limited by rather simple systems (1-2 inputs, 5-10 states), but allows working at high sampling rates (up to 1 MHz) and has a simple code to be embedded in the system [9].

Another often used way to implement explicit MPC - reducing the number of regions. The analysis of regions is made for this purpose and those which contain the same control functions are joined together. During this procedure optimality of the solution is lost, so it is needed to test suboptimal solutions for compliance with the task (1.3).

Some authors [10] propose to abandon the complete separation of the state space into regions and confine dividing into sub-areas, each of which contains a set of regions. The control law is assumed to be produced online, like in MPC, but only within the desired sub-location.

1.3.3 Advantages and disadvantages

Summarizing the above, we can describe all major advantages and disadvantages of explicit MPC.

Advantages:

- Suitable for rapid control processes;
- There is a possibility to account constraints;
- Decreased requirements for computing power in comparison with MPC;
- Decreased memory requirements in comparison with MPC;
- Provides optimal control.

Disadvantages:

- Increase in the number of regions with growth constraining conditions;
- The requirement for an relatively precise knowledge of the model;
- The negative influence of the increasing number of regions on the requirements for processing power and memory.

On the basis of this information, we can talk about explicit MPC application in real processes control.

1.3.4 Implementation

After all computations performed offline, we have the solution (1.8) in the form of a table with an array of linear gain feedback. The desired coefficient is selected online by finding the critical regions $\{x: H_i x \leq k_i\}$, represented in the form of polyhedral, in which the current state vector $x(t)$ is located. This problem is called the problem of localization of the point, and it's most simple solution is consistent passing through all regions, until finding the correct one. This approach is very easy to implement as a program, but with increasing number of regions M, it becomes less and less attractive. Therefore, more advanced algorithms are used. Based on the properties of multiparametric solutions it is proposed to use algorithms that can help to avoid storing information about all regions, significantly reduce requirements to data storing and to the computational complexity of estimating desired control action.

The first algorithm involves the introduction of hyperplanes that separate regions by the method of binary search. In this case, the dependence between residence time of the desired region and their numbers becomes logarithmic.

Another way involves the limitation of the localization of the point problem by set of regions, in which the system can switch from the current state in one step

within one sample. But you need to have a correct estimation of the maximum difference between the states of the real process and a predictive model.

Ultimately, the choice between explicit MPC and MPC should be based on the requirements to the controller (speed CPU, RAM and memory data).

Common development tool is a *Hybrid Toolbox* for *Matlab*, which description we shall discuss later.

In actual production method of explicit MPC occupied a niche for control plants with a high sampling rate and the relatively small size (1-2 controllable input parameters 5-10). Most real applications of explicit MPC are related with the field of automotive and power converter [2].

2 TASK FORMULATION

The goal is to implement explicit MPC in real application. There are some subtasks:

- synthesize a controller for controlling of the system “DC motor – generator”;
- explore synthesis parameters influence on control quality;
- compare results with other control methods;
- realize synthesized controller on microcontroller;

There are different requirements to these tasks.

Requirements to control. For controller designing use *Hybrid Toolbox* for *Matlab*.

Requirements to hardware. Realize system on 1 microcontroller. System shouldn’t include PC. Design and create interface scheme between the plant and microcontroller.

Requirements to software. According to the modularity principles, write library files for every module. Make a control program, using the written modules. The plant output should be led to desired value, set in the control program.

3 DESCRIPTION AND IDENTIFICATION OF THE SYSTEM

3.1 System description

As an object of control is used system "DC Motor - Generator" which diagram is shown in Fig. 3.1. The system consists of a DC motor M connected by elastic coupling EC with the tachometer SM.

Motor M is excited by controlled voltage $u_{m.}$ Output signal is voltage from the tacho $u_{sm.}$ Controllable variable in the system is the rotational speed of the generator $n,$ measured by the tacho [11].

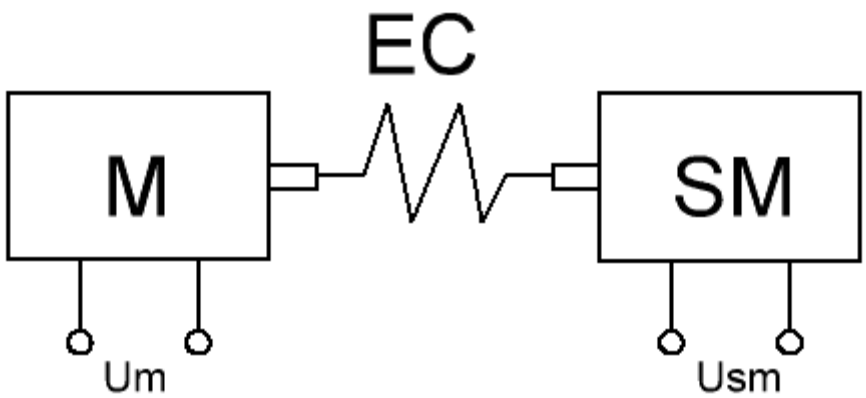

Fig. 3.1: System Diagram "DC motor - Generator"

Connection to the PC can be used for making the system identification and testing of control algorithms. A functional block diagram is shown in Fig. 3.2.

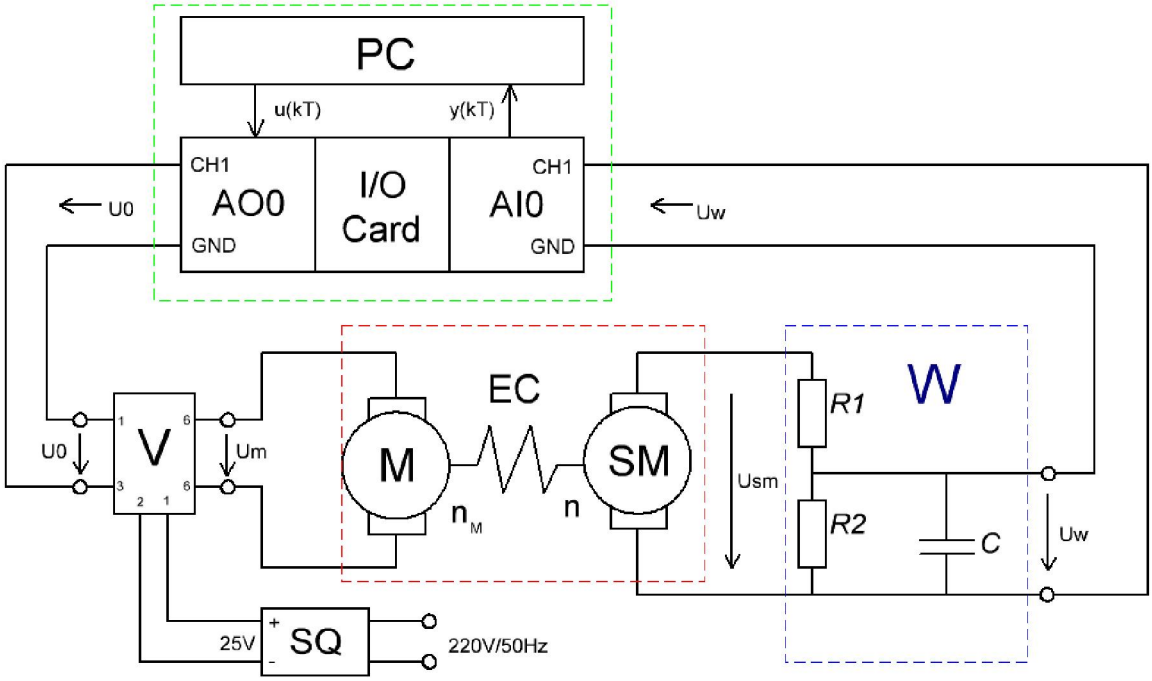

Fig. 3.2: Functional diagram of the system under control of the PC

The system "DC motor - Generator" is highlighted with a red rectangle. Other elements serve for the correct transmission of information from the PC to the object and back. The circuit consists of an amplifier input signal V, which leads to its desired range of values, the power source SQ, the output voltage decreasing circuit and filter W, and the I/O board, which provides signal conversion for the PC.

Input channel of I/O board converts the signal from converter u_w to signal $y(kT)$. Resulting range is 0 ÷ 10 V. Output channel of I/O board generates a control voltage, the value of which is converted from the signal $u(kT)$ [12].

Detailed information on the elements of the circuit is shown in tab. 3.1, the description of the signals is given in tab. 3.2.

Tab. 3.1: Description of the device

Character	Description

M	DPT DC excitation type P2TV369, 24V, 2000 rev / min
SM	Tachometer, type P2TV369, 24V, 2000 rev / min
EC	Flexible coupling
V	Signal Amplifier
W	Converter (voltage divider) and the low pass filter
SQ	Power supply, ZPA Decin, 25V / 10A
R1	Resistor 59k
R2	Resistor 41k
C	Capacitor 100 nF
I / O	I / O board PC
AO0	Analog output card
AI0	Analog input board

Tab. 3.2: Signal Description

Character	Description	Value
u(kT)	Normalized digital control signal	0 ÷ 10V
u_0	Voltage analog output I / O board	0 ÷ 10V
u_m	The output voltage of the amplifier	0 ÷ 24V
u_{sm}	Output voltage tacho	0 ÷ 24V
u_w	The output voltage of the converter	0 ÷ 10V

y(kT)	The normalized digital feedback signal	0 ÷ 10V
n_m	The engine speed	0 ÷ 2000 rev / min
n	The rotational speed of the generator	0 ÷ 2000 rev / min

Appearance of the system is shown in Fig. 3.3.

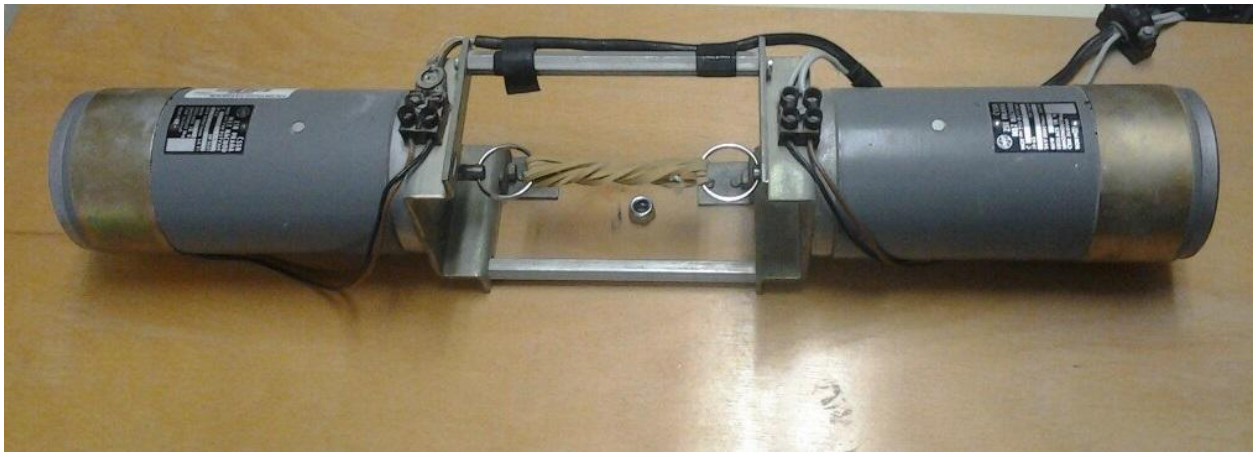

Fig. 3.3: System "DC motor - Generator"

Systems with such dynamic properties are widely used in designing of robotic arms, in coupling between motor and instrument. Also it is possible to make a simulation of processes with low-frequency oscillations in axes, for example, counting of ropes flexibility in elevator's systems in high buildings.

3.2 Identification

Effective management of the real process is based on the knowledge of its behavior and properties that can be expressed in the form of a mathematical model. Control object in this case is a system of several elements, mathematical models which are unknown to us. At the same time, the use of explicit MPC requires knowledge of the object model, which leads to the need to identify it.

Identification of the plant is based on the static and dynamic characteristics, which can be obtained by direct measurement. The task of identifying - accurately and completely describe the system.

3.2.1 Static characteristic

Static characteristic of the object displays the dependence between its output and the input in steady state. Measuring circuit in *Matlab* is given in Fig. 3.4. To apply and receive signals directly from the object library *Simulink Real-Time Windows Target* for *Matlab* is used.

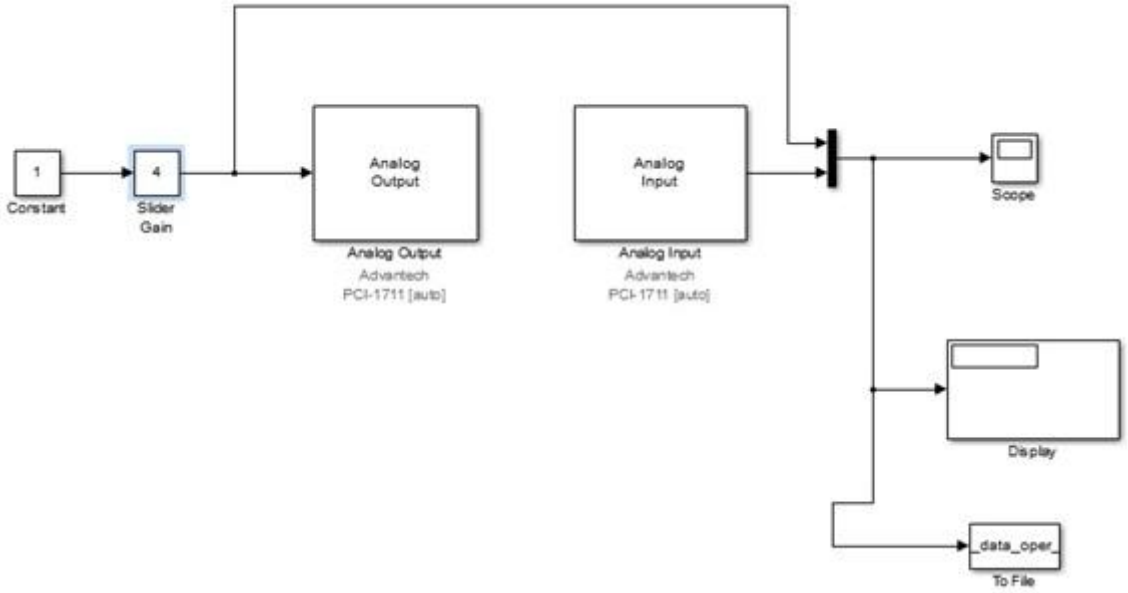

Fig. 3.4: Measurement circuit

The range of values of the input exposure: 0 ÷ 10V. Step sampling time Δt = 0.01 s. The measurement results are shown in Fig. 3.5.

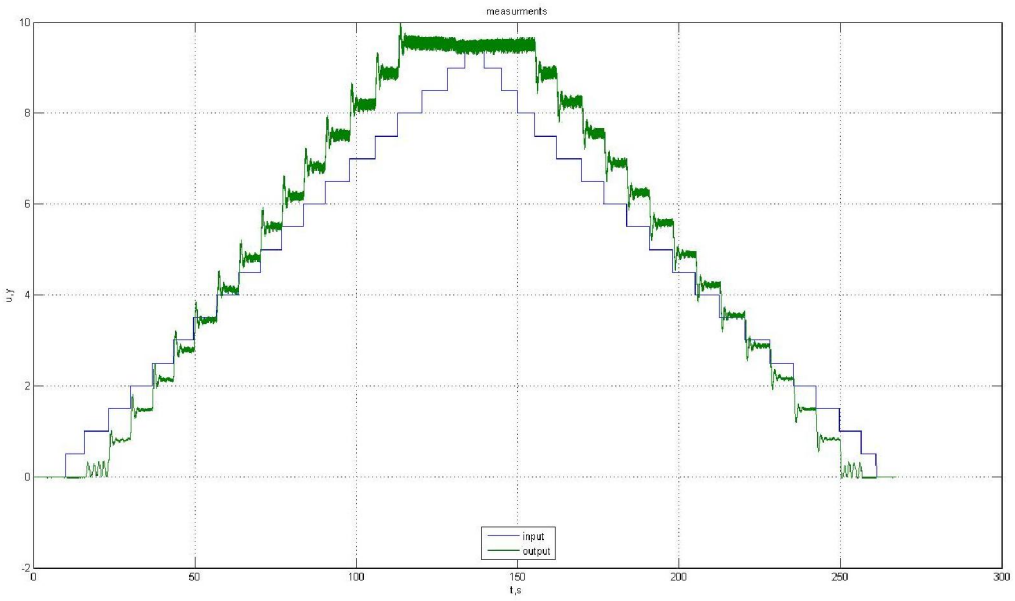

Fig. 3.5: Graphs u(t), y(t)

At the initial stage ($U = 0 \div 1$V) output variable *y(t)* fluctuates around 0 V. Closer to the upper limit of the range of input values the output variable reaches saturation (~ 9V) and does not respond to increase of the input signal.

For each step of the input exposure is measured steady-state value. To determine the steady-state value averaging filter is used for the last 20 values before the next step of the input action. On the basis of these measurements is based static characteristic *y(u).* The result is shown in Fig.3.6.

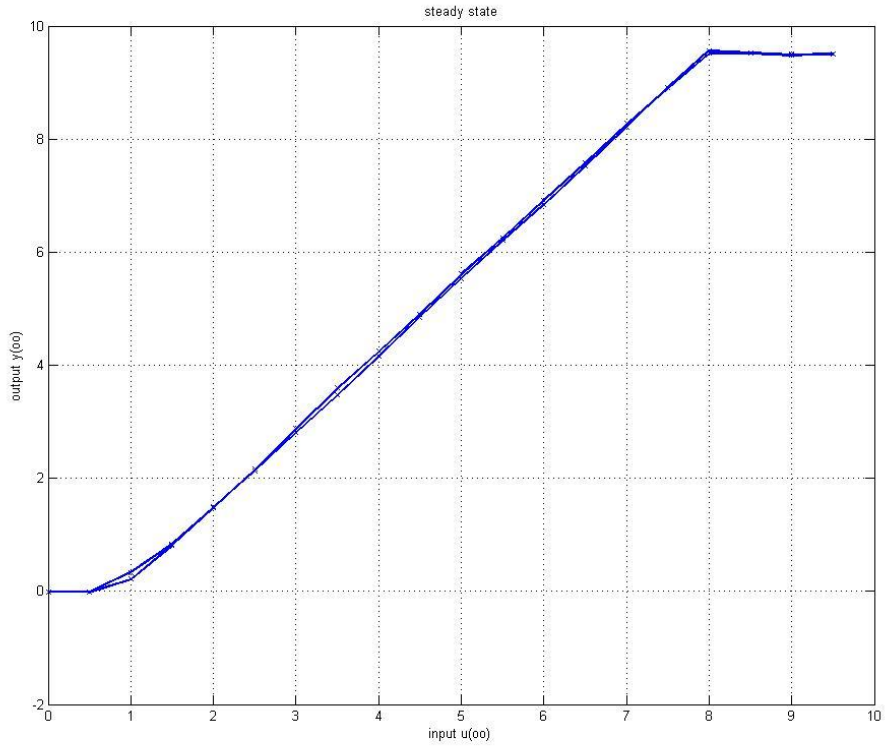

Fig. 3.6: Static characteristic of the plant

As can be seen from the static characteristics, the object seems to be linear in the range of input actions $U = 1 \div 8$ V. In the range of $U = 0 \div 1$ V there is not enough power to overcome the starting torque of the motor. In the range $U = 8 \div 10$ V object is in the saturation. There is a small hysteresis loop caused by noise and inaccurate estimates of steady-state value.

3.2.2 Dynamic behavior

Second, the most important part of the identification is to provide a mathematical model that describes the dynamic properties of an object. In this case it is useful to present model as a transfer function.

On the basis of the data (Fig. 3.5) it is possible to obtain a set of transfer functions, each of which corresponds to its transition process. By analyzing the received data, the nominal transfer function is selected, as well as the designated limits, which may vary on its parameters.

Getting of the most appropriate transition process of the transfer function is performed by minimizing the cost function represented by a quadratic criterion (3.1)

$$J = \min_{x} \sum_{i=1}^{N} (y_i - y_{Mi}(x))^2 \tag{3.1}$$

where y_i is measured value, y_{Mi} is value of current model transfer function [13].

After analyzing the physical description of the object, the estimated transfer function has the form (3.2).

$$F(p) = \frac{k}{(Tp+1)(T_0^2 p^2 + 2T_0 \varepsilon p + 1)} \tag{3.2}$$

During identification were received 26 transfer functions. For an analysis of each of them were built responses to the unit step (Fig. 3.8), the distribution of the roots (Fig. 3.9), as well as an example identified transfer function (Fig. 3.7).

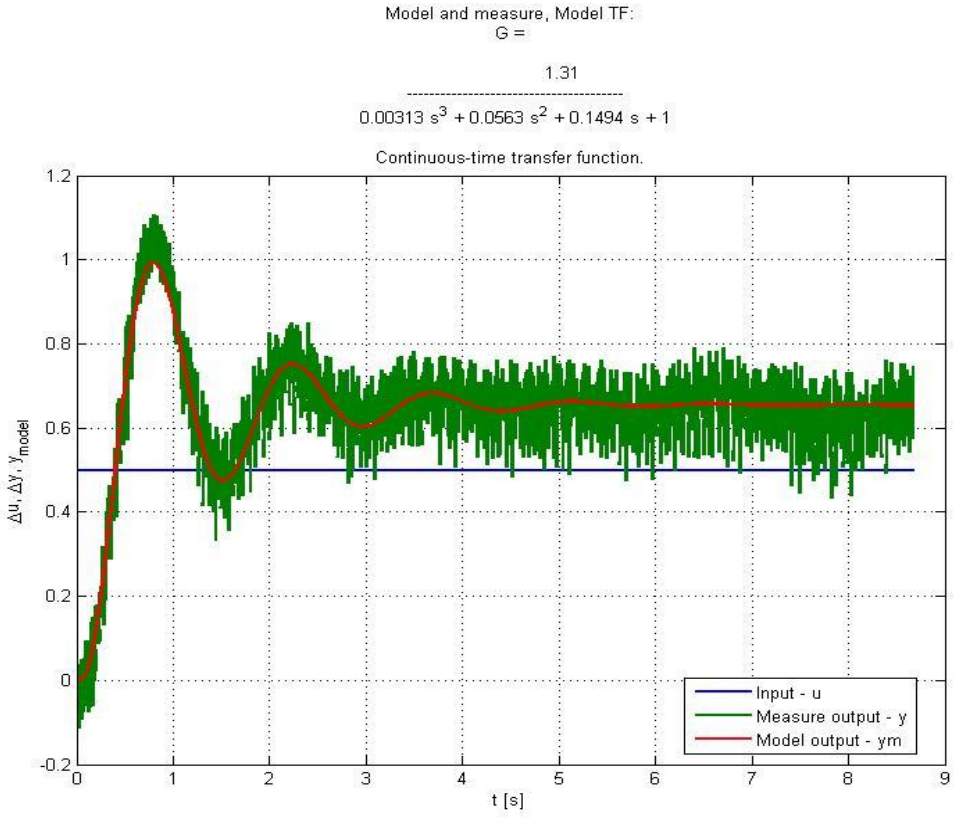

Fig. 3.7: An example of transfer function identification

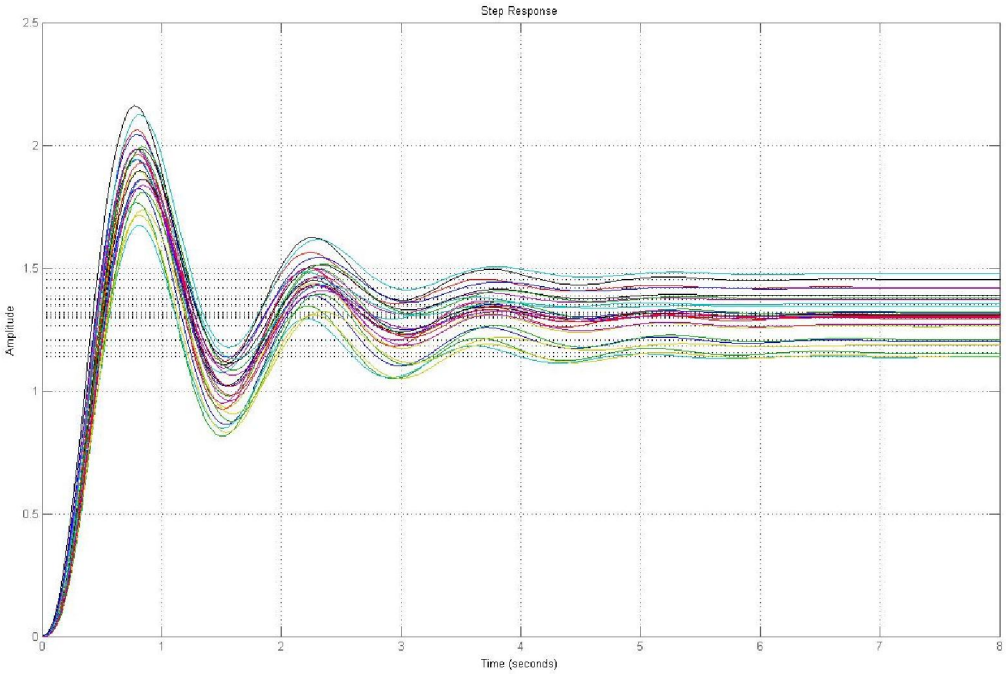

Fig. 3.8: Responses to a step exposure

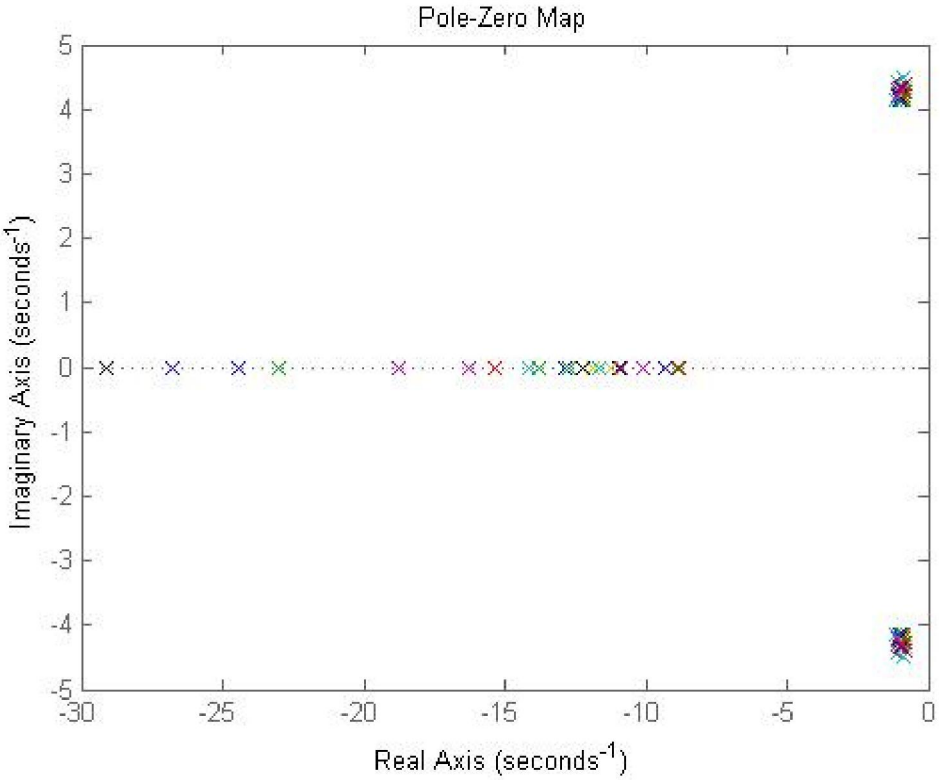

Fig. 3.9: Distribution of roots in the phase plane

Set of responses to a unit step (Fig. 3. 8) clearly shows the similarity of the dynamic properties of the systems, as well as the difference between their gains. Portrait root (Fig. 3. 9) allows to suggest that the oscillatory unit in the plant has a small variation of parameters (complex-conjugate roots are concentrated in one place), while the spread of parameter of an aperiodic unit is large enough. To obtain a complete picture of all analyzed parameters derived transfer functions, minimum and maximum values of the coefficients of the transfer functions in the form (3.2) are presented in tab. 3.3.

Tab. 3.3: Extreme values of the coefficients

Parameter	Minimum	Maximum
b_0	1.1376	1.4763
a_3	0.0018	0.0059
a_2	0.0561	0.0689
a_1	0.1355	0.2124

According to the averaged values of the poles was obtained final transfer function

$$F(p) = \frac{1.305}{0.00394p^3 + 0.0689p^2 + 0.163p + 1} \tag{3.3}$$

For obtained transfer function was drawn its step response with all other responses (Fig. 3.10).

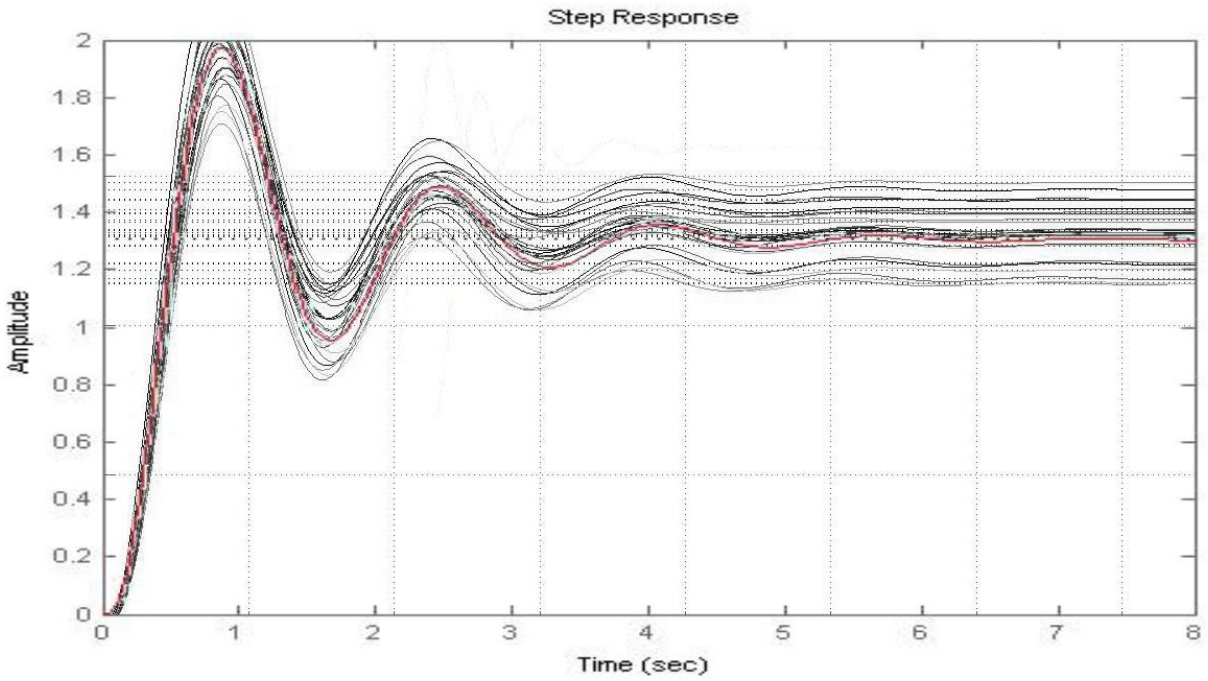

Fig. 3.10: Step-response of obtained transfer function

It is possible to use this transfer function as a model of the plant, but it is relatively robust estimation of it. If the process in the plant goes only near some working point, it is advisable to make an additional identification and derive more precise transfer function.

3.2.3 Additional identification

As an working point was chosen point $(u,y) = (4,4)$ with deviation of input action $\Delta u = \pm 1$ V. Identification goes like in previous chapter, but for all steps at one moment. It allows deriving one common transfer function at once.

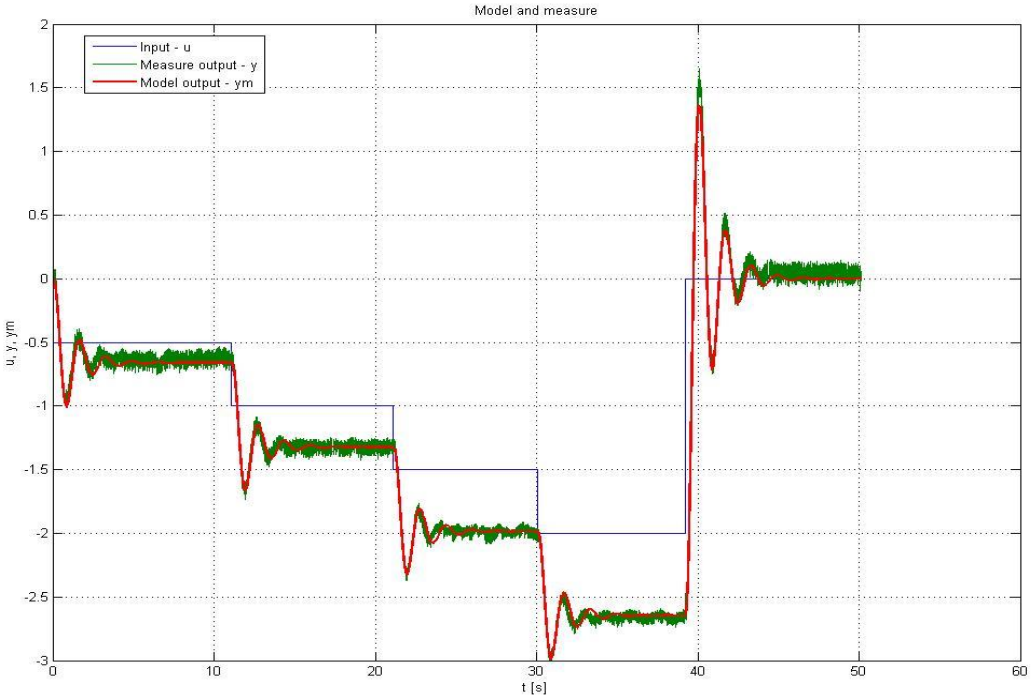

Fig. 3.11: Identification results near working point

After additional identification was defined new transfer function

$$F(p) = \frac{1.323}{0.003068p^3 + 0.06758p^2 + 0.1495p + 1} \tag{3.4}$$

For better understanding it is useful to compare poles of (3.3) and (3.4). Poles are introduced in tab. 3.4.

Tab. 3.4: Poles of transfer functions

Poles of (3.3)	Poles of (3.4)
-15.89	-20.42
-0.799 ± 3.91i	-0.802 ± 3.91i

Dynamic is nearly the same, but open-loop gain relatively different. To make an optimal regulation for other working points it is good to recognize its own transfer function.

4 MODELING OF SYSTEM WITH CONTROLLER

4.1 Hybrid Toolbox library

For implementation of explicit MPC in *Matlab Simulink* in 2003 *Hybrid Toolbox* library was created. It is shareware product; its author is Alberto Bemporad, professor of Institute for Advanced Studies Lucca.

This library gives wide opportunities for modeling, simulation and analysis of hybrid systems. There is set of instruments for designing of explicit MPC controllers. Also it is possible to use some C-functions for making embedded programs [14].

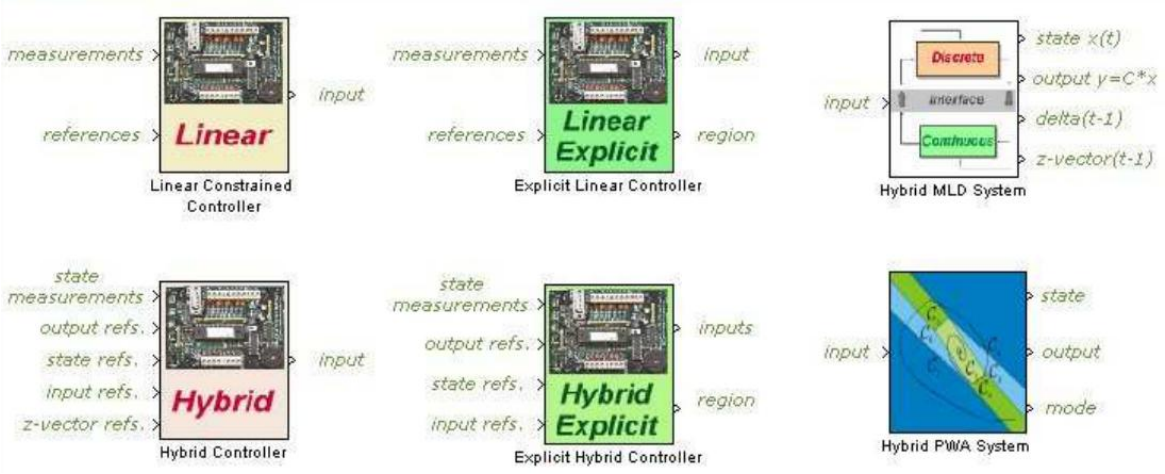

Fig. 4.1: Elements in Hybrid Toolbox

For making an explicit controller it is necessary to work with two functions – *lincon* and *expcon*. First of them synthesize a linear controller on the basis of quadratic optimization. The second one converts this controller into explicit form, and it is necessary to set all constraints for this conversion. Also there is one more useful function *hwrite*. It writes all matrices of controller in the special file.

Also *Hybrid Toolbox* provides the region-searching algorithm in the form of C-function. It uses matrices from file, written by *hwrite* function. This algorithm is introduced in Fig. 4.2.

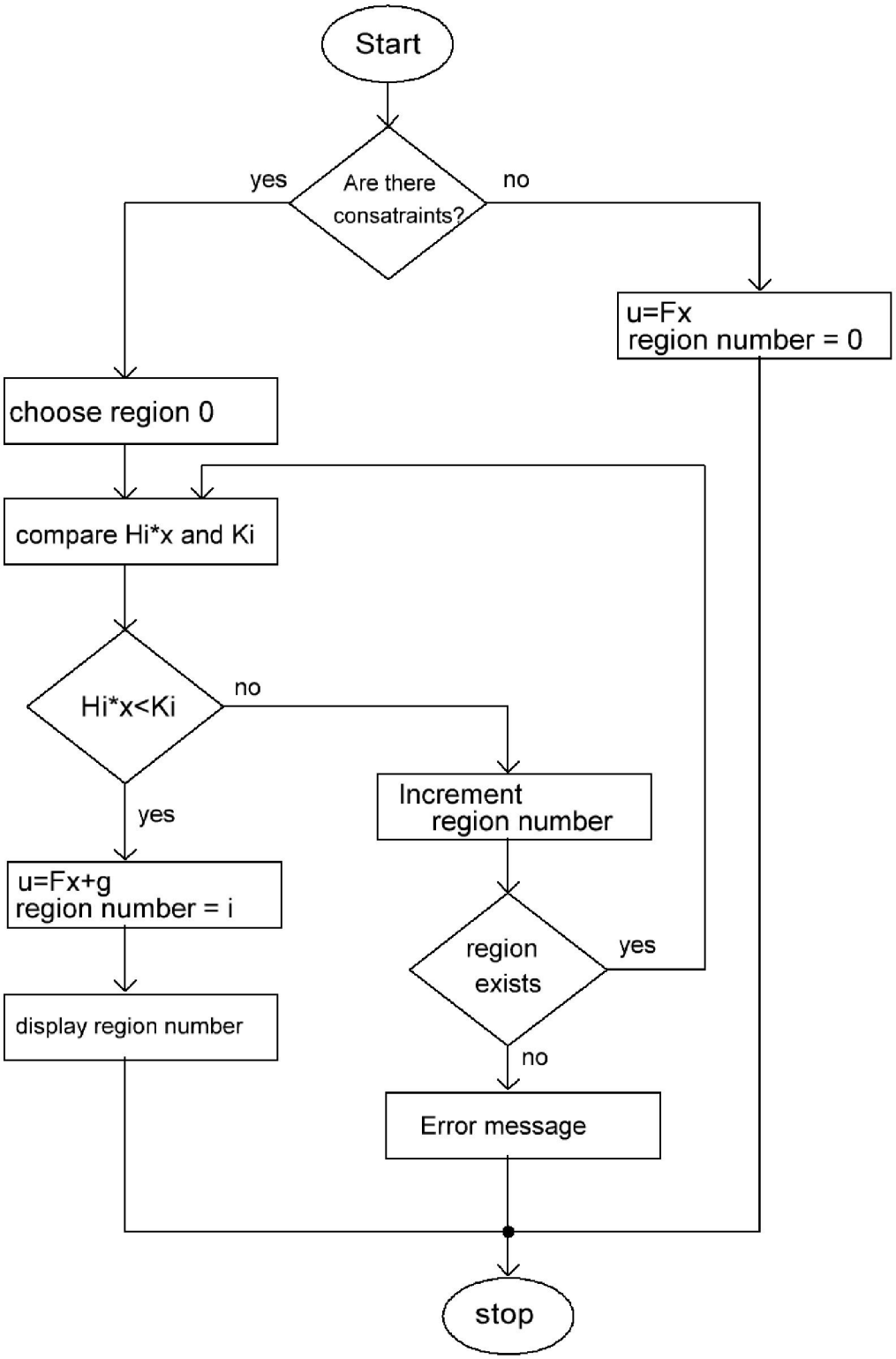

Fig. 4.2: Searching algorithm

It is very simple method for searching the current region and it is good for controllers with not so high number of detected regions.

4.2 Designing of model with controller

Designing process is setting up of some parameters and using of *Hybrid Toolbox* functions for synthesis. Model in *Matlab Simulink* looks like in Fig. 4.3.

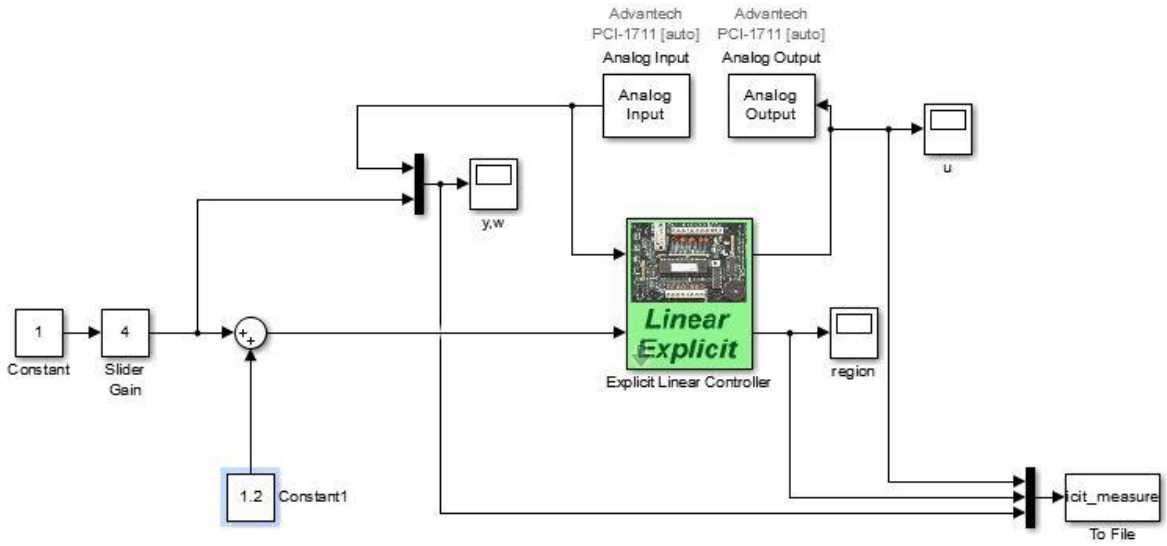

Fig. 4.3: Model with controller

Inputs and outputs of the real process are elements of *Real-Time Windows Target* library, connected with I/O circuit. For correct work of unit "Explicit Linear Controller" it is necessary to derive the explicit control law. The best way to do it is to write script in *Matlab*. The final script *explicitMPC.m* includes setting of all parameters (tab 4.1), controller synthesis, and simulation of the model (Fig. 4.3). Such parameters as sampling time, control horizon and prediction horizon will be explored in Chapter 4.4. Tab. 4.1 contains the optimal parameters found during exploration. The initial values were defined according to [13].

Tab. 4.1: Parameters setting for synthesis of controller

Parameter	Value	Note
T_s	0.01	Sampling time

Cost.S	10	Weights on output
Cost.T	0.1	Weights on input incrementing
N	10	Prediction horizon
N_u	2	Control horizon
Limits	0÷10	Limits of input
Range.x	-100÷100	Size of state space
Range.y	-10÷10	Limits of output
reltol	10^{-6}	Algorithm stop parameter
qpsolver	‘qpact’	Way of solving the quadratic program

Exploration of influence such parameters as length of prediction horizon, length of control horizon, sampling time will be introduced in next chapters.

4.3 Offset of static characteristic

Because of non-linearity in the process (Fig. 3.6), its linear part is shifted relative to the point $y = 0$, $u = 0$, and output has a difference with the reference action (Fig. 4.4). The problem is solved according to subchapter 1.2.1.

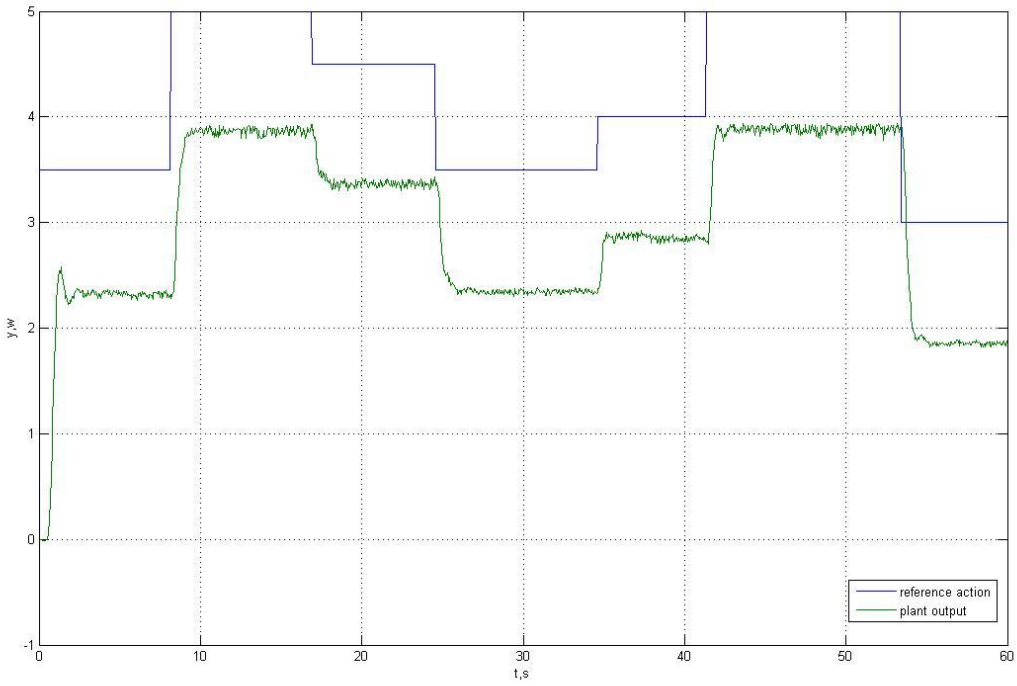

Fig. 4.4: Measurement with offset

4.4 Exploration of parameters influence

To derive the most qualitative system control, it is necessary to explore influence of different parameters on the processes in the system.

4.4.1 Sampling time

The plant is connected with digital system, and it is necessary to work with discrete signal. Chosen sampling time influence on the work of whole system. Fig. 4.5 demonstrates closed loop step-response for different sampling times.

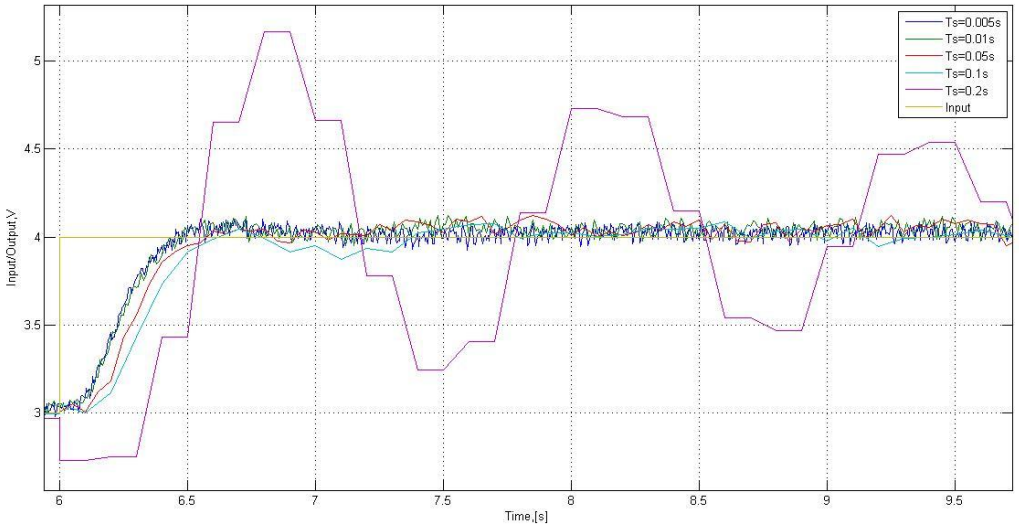

Fig. 4.5: Step-response of closed loop with explicit controller with different sampling times

If the value of sampling time is less than $\Delta t \leq 0.01$ s, there are no significant changings in the process. If the value of sampling time is more than $\Delta t \geq 0.1$ s, the quality of process decreases. So, for good results, it is enough to choose sampling time in range $0.01 \text{ s} \leq \Delta t \leq 0.1$ s.

4.4.2 Prediction horizon

This parameter sets the length of time segment, for which algorithm makes the prediction. Step-responses with different values of N are introduced in Fig. 4.6. Changing of this parameter also leads to number of regions changing. These results are introduced in Tab. 4.2.

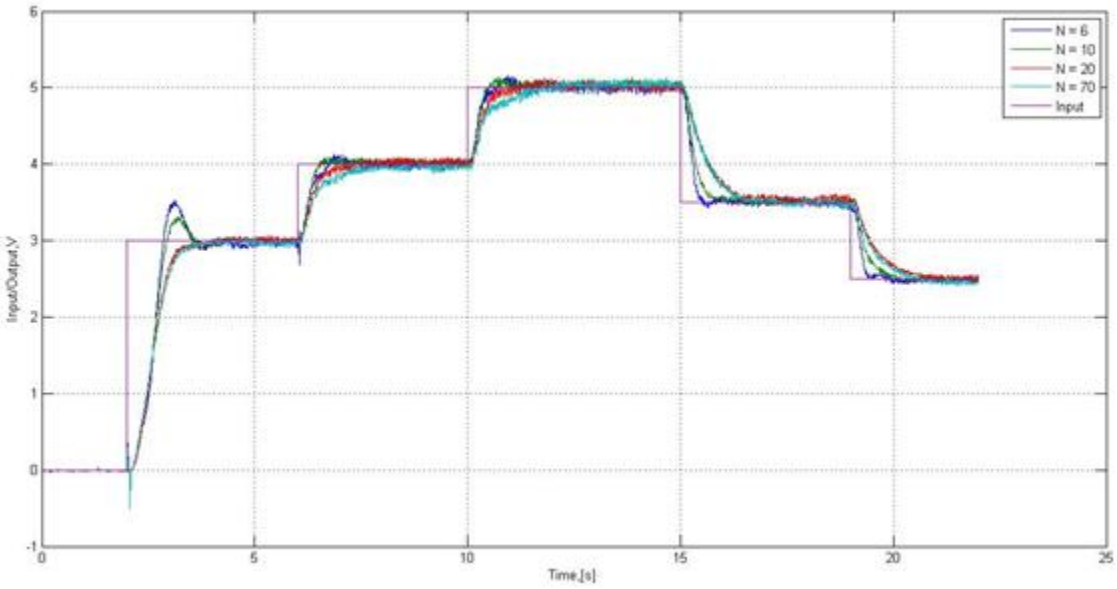

Fig. 4.6: Step-responses of closed loop with different prediction horizons

Tab. 4.2: Number of regions for different prediction horizons

Prediction horizon *N*	Number of regions
6	7
10	7
20	5
70	5

With increasing of prediction horizon the number of regions decreases. It leads to reducing controller complexity and to increasing of process rise-time, especially for negative steps. Overshooting in this case goes to zero, also during the start of all process. To choose the right value for this parameter is necessary according to the requirements.

4.4.3 Control horizon

This parameter sets the number of steps in calculated control sequence. Fig. 4.7-4.8 show step responses with different control horizons. Changing of this

parameter also leads to number of regions changing. These results are introduced in Tab. 4.3.

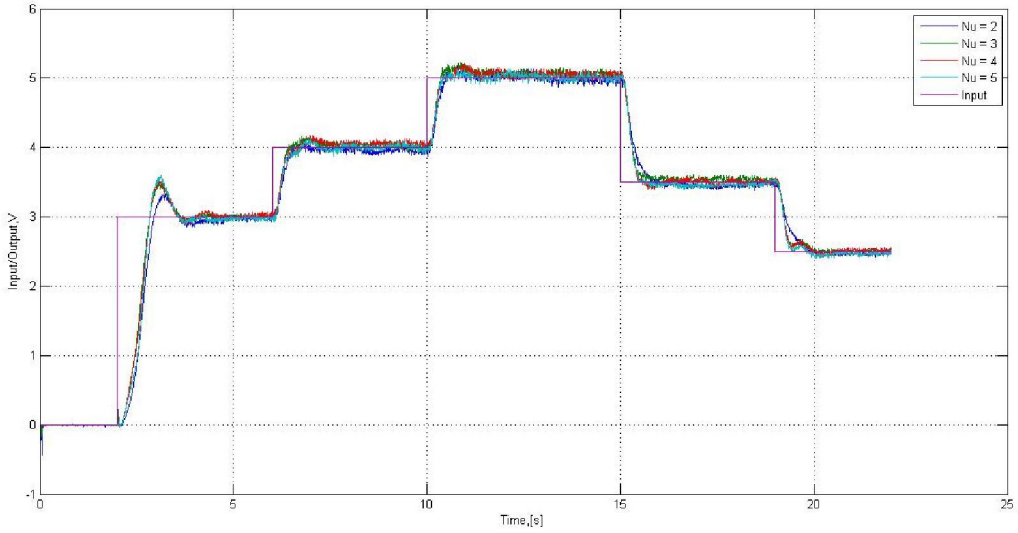

Fig. 4.7: Reaction of system on input sequence with different control horizons

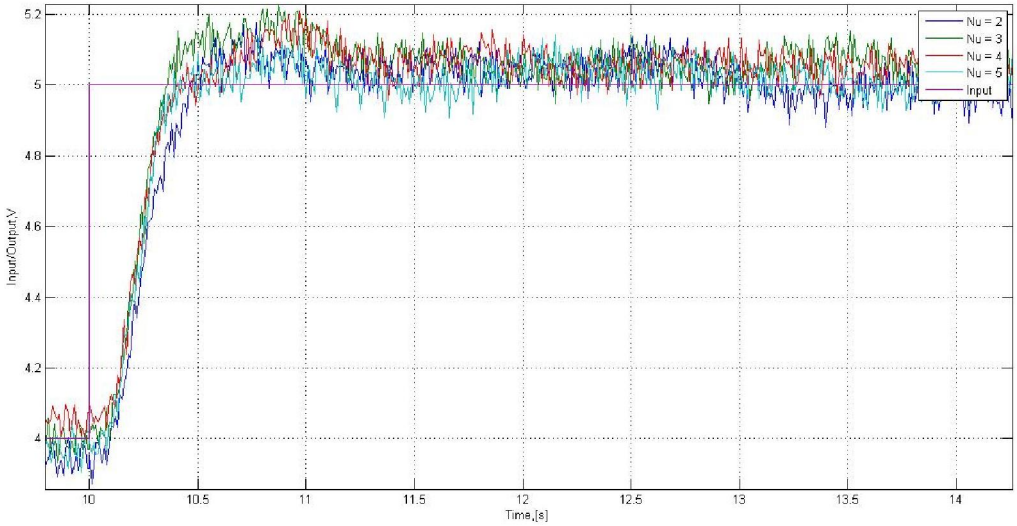

Fig. 4.8: Step-response of closed loop with different control horizon

Tab. 4.3: Number of regions for different control horizons

Control horizon *Nu*	Number of regions
2	7

3	19
4	52
5	103

Increasing of control horizon doesn't lead to significant changes in process. But the number of critical regions increases; it leads for additional memory and computational power consumption of controller. Because of that, it is necessary to choose the lowest possible length of control horizon.

Final parameter selection is introduced in Tab. 4.1.

4.5 Modeling results

After setting up of all parameters and including the offset to the scheme, next dependencies were received: input/output of time (Fig. 4.9), control action of time (Fig. 4.10). Results of partition of the state space on regions and step-response are also introduced (Fig 4.11 and Fig. 4.12). Changing of state variables during the control process is introduced in Fig. 4.13.

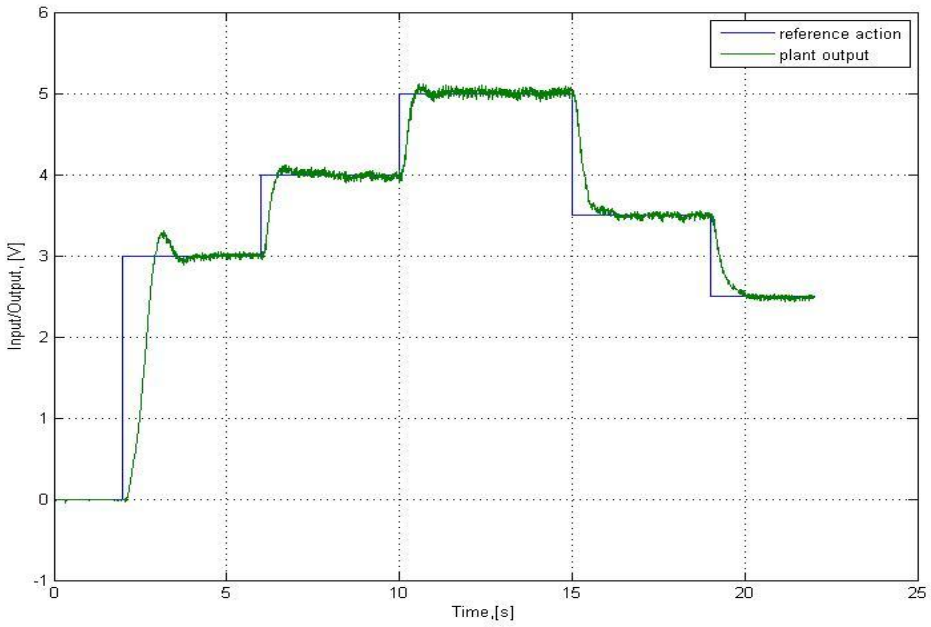

Fig. 4.9: Dependence between input/output and time

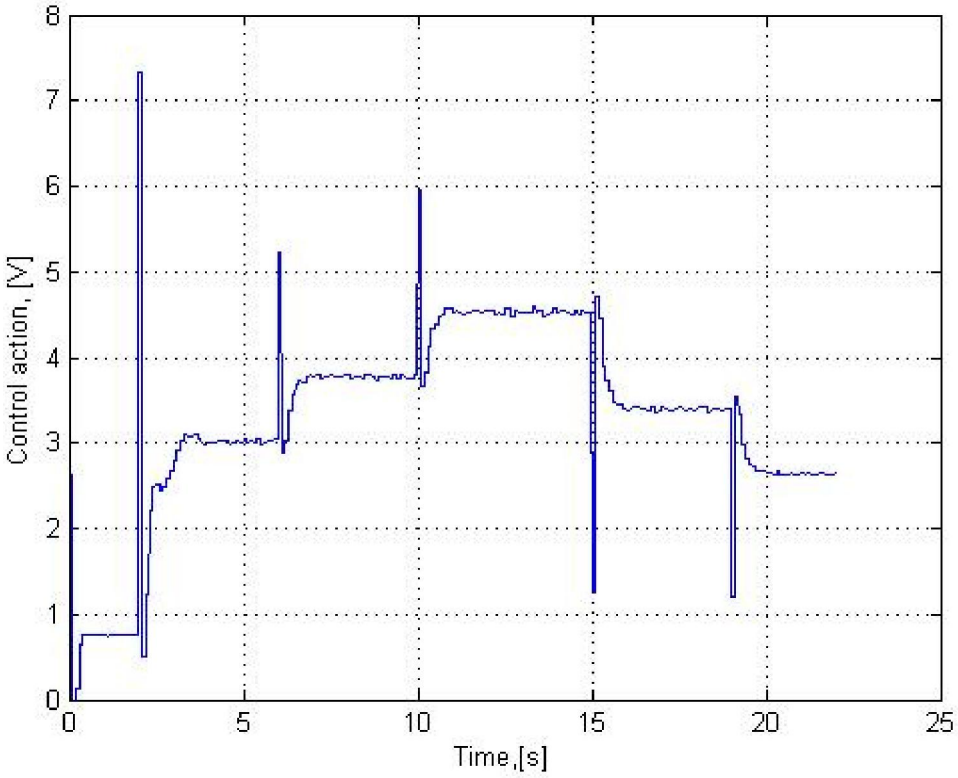

Fig. 4.10: Dependence between control action and time

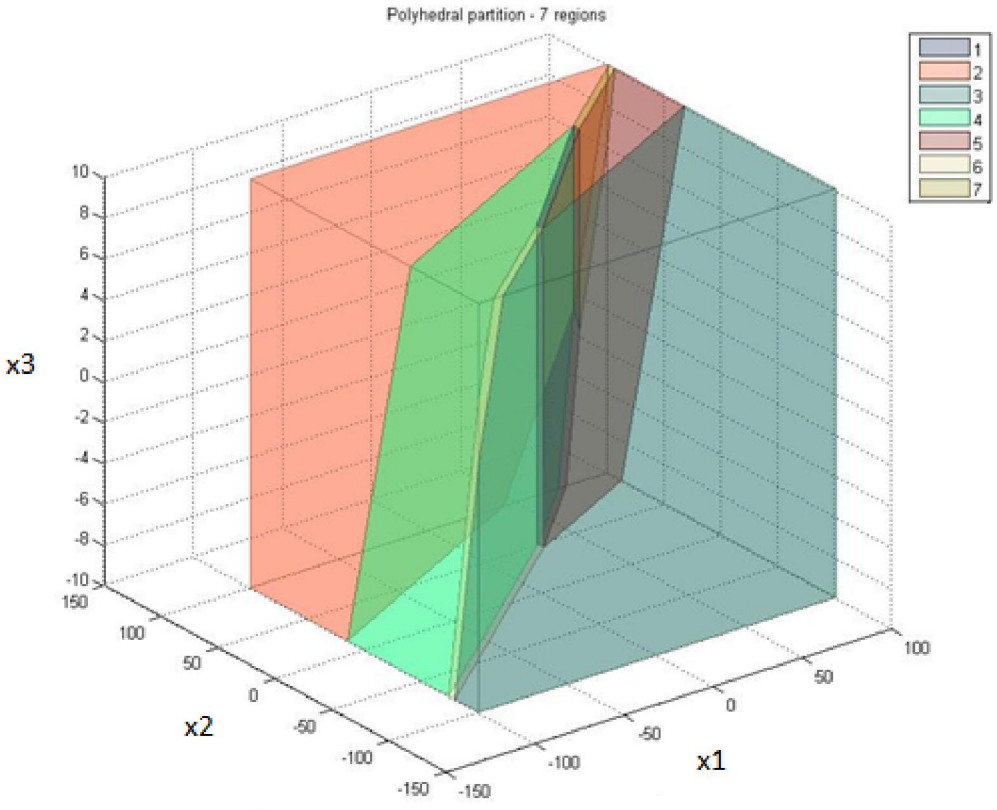

Fig. 4.11: Partition on regions

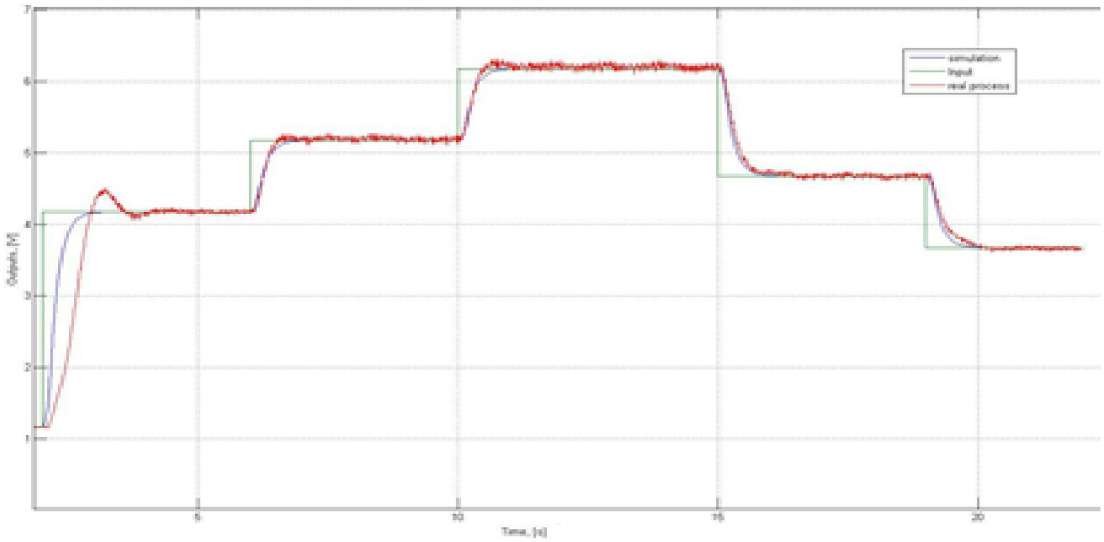

Fig. 4.12: Step-response of closed loop with explicit controller

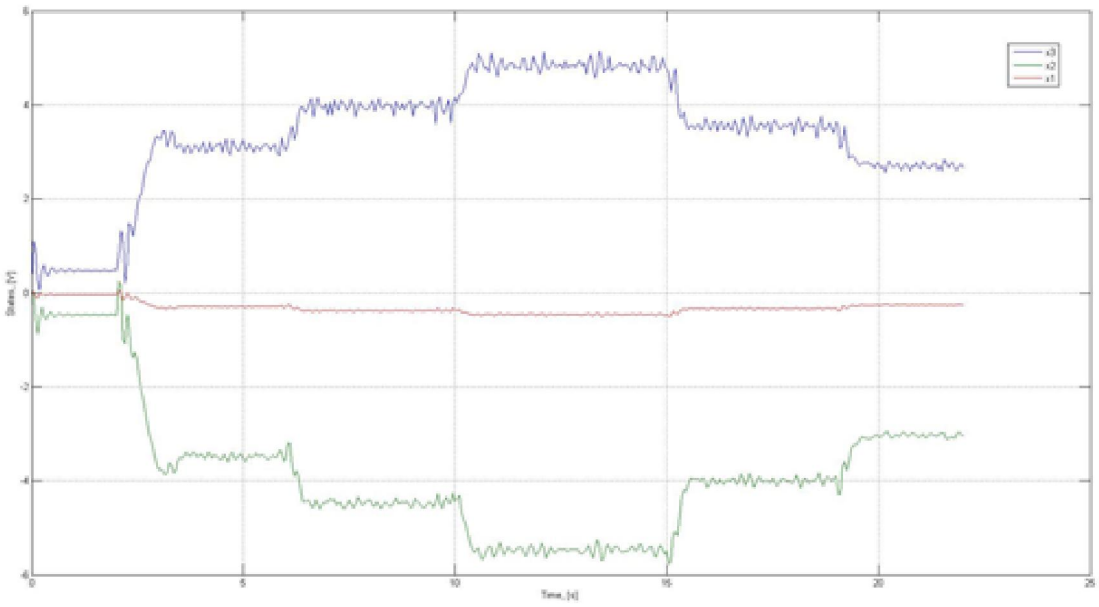

Fig. 4.13: State variables in time

From Fig. 4.9 and Fig. 4.12 one can see that system successfully reacts to steps of input action and leads the output to the reference value. Rise time is t_r = ~0.8 s, overshoot is σ = ~5%.

On the first step system reacts worse because of starting torque of the motor. In Fig. 4.11 one can see, that during the synthesis process 7 critical regions were detected in the state space.

4.6 Comparison with other control methods

To speak about possibility of explicit MPC implementation, it is necessary to compare it with some other control methods. For comparison were chosen two popular methods: PID control and robust control.

4.6.1 PID controller

At the moment, this method is one of the most usable control methods. But in systems of high order or with significant non-linearity, PID control loses its optimality. Sometimes control with it is impossible [5]. System "DC motor – generator" is close to linear, and comparison with PID control is good for explicit MPC checking.

PID controller is control law, based on three members: error of output, integral of this error and its first time-derivative. Each member has its own coefficient and purpose. P-member tries to keep the output closer to reference trajectory. I-member eliminates the static error in the system. D-member is used for better quality of system's dynamic characteristics [5].

Description of PID controller

$$u(t) = k_p \cdot e(t) + k_i \int_0^t e(\tau)d\tau + k_d \cdot \frac{de(t)}{dt} \tag{4.2}$$

where *e(t)* is difference between plant output and reference action. For derivation real derivative unit is used [5].

Synthesis of PID controller was made in *Matlab* with standard methods of auto tuning. Final values of coefficients are introduced in Tab. 4.4.

Tab. 4.4: Parameters of PID controller

Parameter	Value

k_p	0.115
k_i	1.15
k_d	0.09
Filter constant N	100

Matlab model is introduced in Fig. 4.14. Results of comparison are introduced in Fig. 4.15-16.

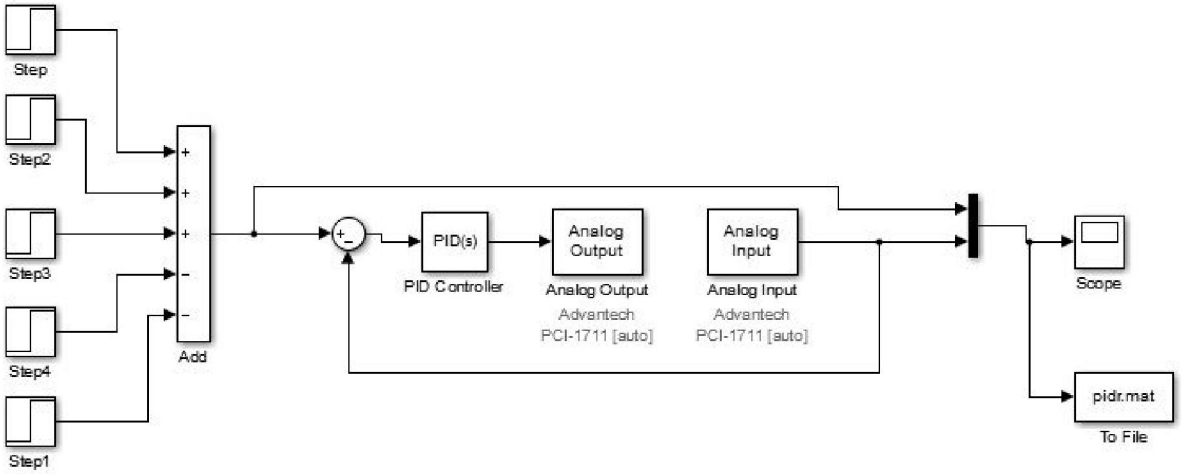

Fig. 4.14: Model with PID controller

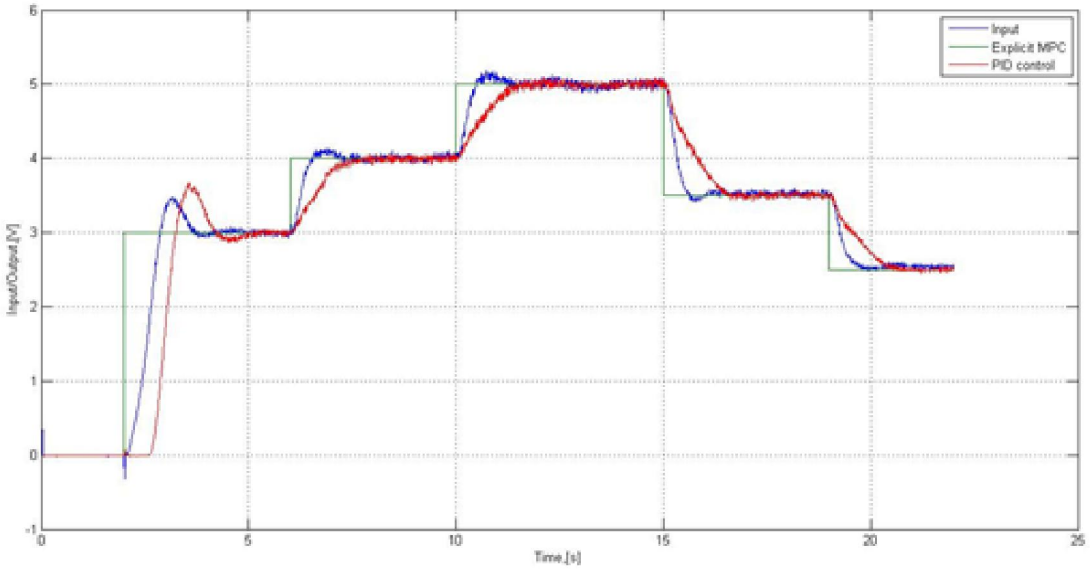

Fig. 4.15: Reaction of the system on input sequence

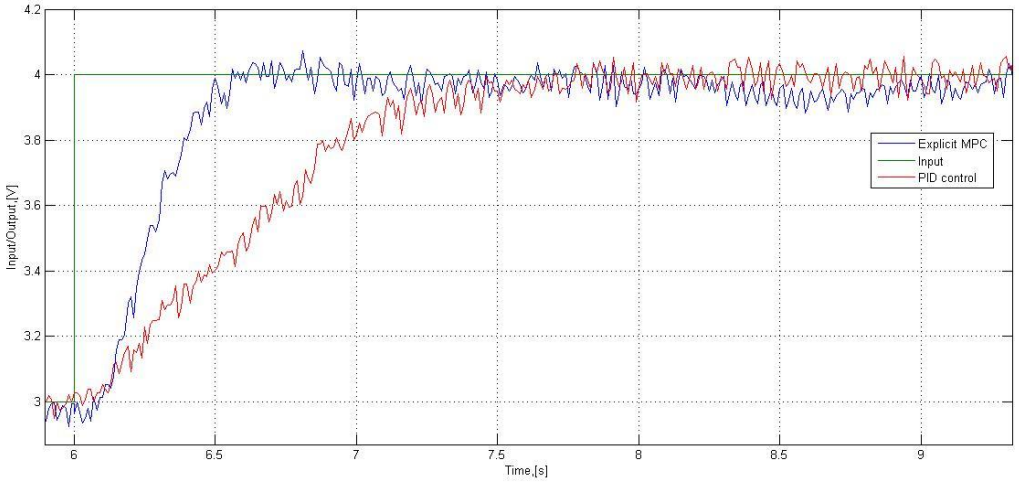

Fig. 4.16: Step-response of closed loop

Rise-time for PID control is t_{PID} = 1.6 s, for explicit MPC is t_{eMPC} = 0.8 s. Comparing results, one can see, that for this system explicit MPC is better than PID control.

4.6.2 Robust controller

The aim of robust control is qualitative control of systems with changing or unknown parameters, non-linearity operating under noises [15]. There are some methods to synthesize a robust controller. For comparison with explicit MPC controller was chosen H_2-synthesis. This controller provides stability of closed loop, works with sensitivity of the system. Transfer function of controller with this method is derived precisely; it allows making the controller without any problems. Disadvantage of this method is high number of iterations during the synthesis procedure [15].

The controller synthesis was made with *Matlab* function *h2syn*. For its implementation it is necessary to use augmented model of plant (Fig. 4.17). Filter transfer functions W_1, W_2, W_3 have to be calculated.

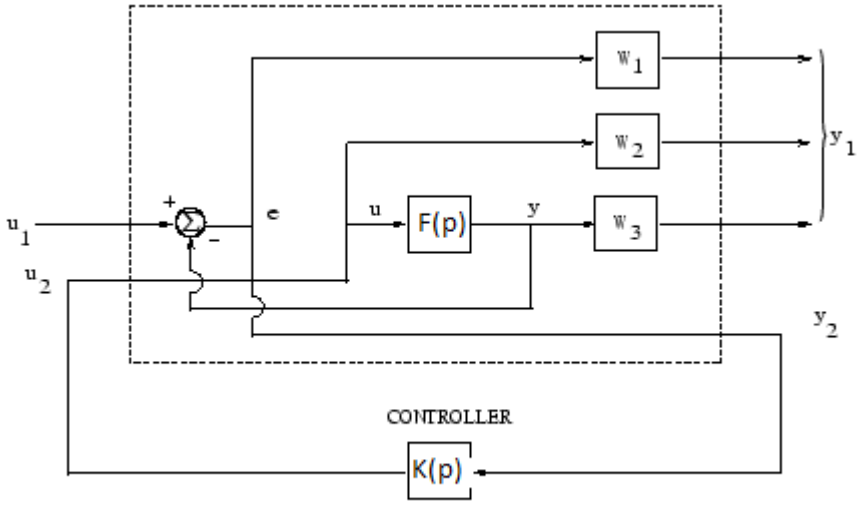

Fig. 4.17: Augmented model

Transfer functions of filters are

$$W_1(p) = \frac{1}{p + 0.001}$$

$$W_2(p) = 0.1\frac{1 + 0.005p}{1 + \frac{0.005p}{10000}}$$

$$W_3(p) = 0.8$$

As a result of the synthesis, transfer function of controller was derived

$$F_r(p) = \frac{0.001p^4 + 2000p^3 + 4.405 \cdot 10^4 p^2 + 9.745 \cdot 10^4 p + 6.51 \cdot 10^5}{p^5 + 230.6p^4 + 6404p^3 + 5.669 \cdot 10^4 p^2 + 3.194 \cdot 10^5 p + 31.94}$$

Matlab model is introduced in Fig. 4.18. Results of comparison are introduced in Fig. 4.19-20.

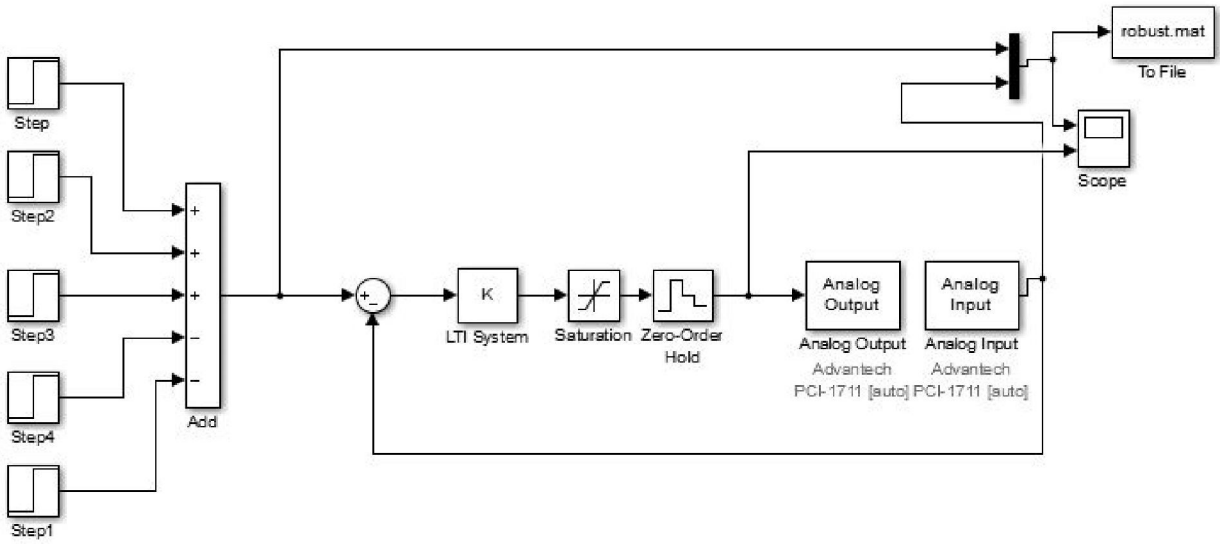

Fig. 4.18: Model with robust controller

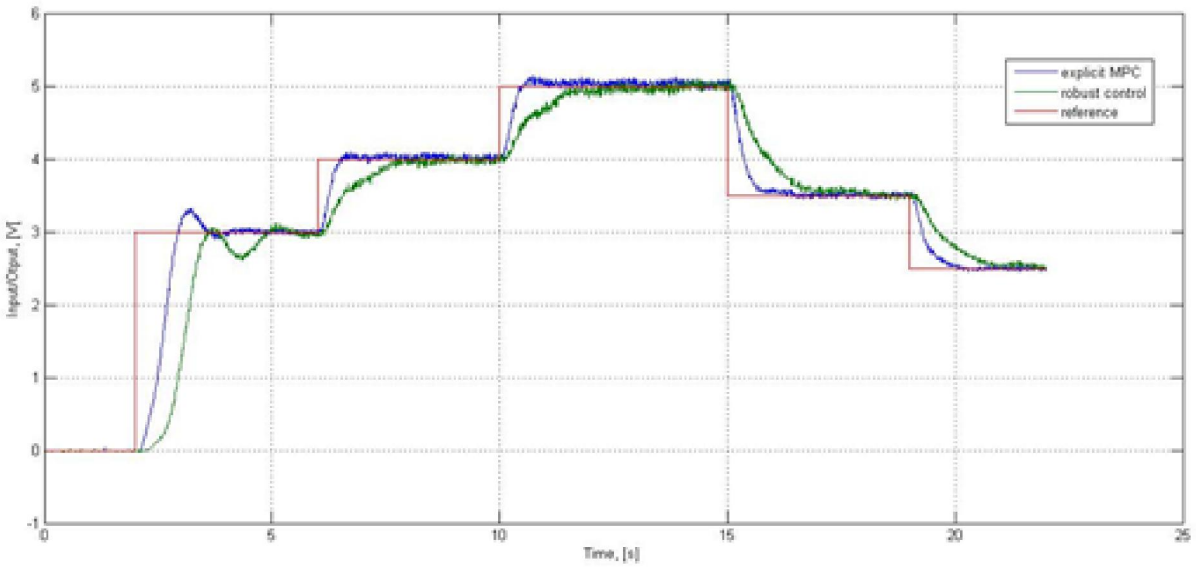

Fig. 4.19: Reaction on input sequence

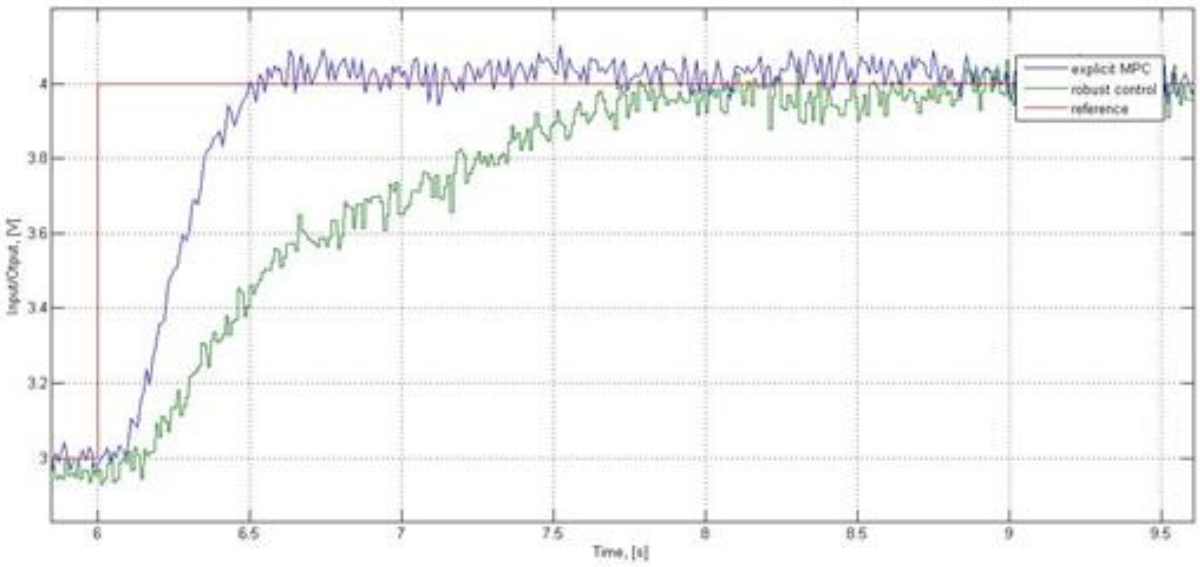

Fig. 4.20: Step-response of closed loop

Rise-time for robust control is t_{rob} = 2.5 s, for explicit MPC is t_{eMPC} = 0.8 s. Comparing results, one can see, that for this system explicit MPC is better than PID control.

4.7 State estimation

To realize explicit MPC on microcontroller, it is necessary to measure or estimate states of the plant. Measurement is impossible for given plant, so, it is necessary to calculate state observer.

For calculation the method from chapter 1.2 is used. For continuous transfer function (3.5) and sampling time, derived in Tab. 4.4, discrete transfer function is calculated. This function is

$$F(z) = \frac{0.0439z^2 + 0.1088z + 0.01486}{z^3 - 1.836z^2 + 1.073z - 0.1105} \tag{4.4}$$

For 1-channel plants it is recommended [5] to use matrix procedure of observer's synthesis. The transposed form is used. Matrix equation (1.8) is

$$\begin{cases} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.1105 \\ 1 & 0 & -1.073 \\ 0 & 1 & 1.836 \end{bmatrix} \cdot \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0.01486 \\ 0.1088 \\ 0.0439 \end{bmatrix} u(k) + \begin{bmatrix} l_0 \\ l_1 \\ l_2 \end{bmatrix} \Delta(k) \\ y_M(k) = [0 \quad 0 \quad 1] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} \\ \Delta(k) = y(k) - y_M(k) \end{cases} \tag{4.5}$$

Factors of correction matrix are derived by formulae (1.13). Let's define the desired poles according to [5]: $z_1 = 0.161, z_{2,3} = -0.162 \pm 0.774i$. In this case the desired polynomial is

$$C(z) = z^3 + 0.164z^2 + 0.573z - 0.1005 \tag{4.6}$$

So, from (4.5) and (1.13) matrix of correction factors is derived

$$\begin{bmatrix} l_0 \\ l_1 \\ l_2 \end{bmatrix} = \begin{bmatrix} 0.01 \\ -0.5 \\ 2 \end{bmatrix}$$

Calculated state observer is included into the system scheme (Fig. 4.4). New scheme has a view like in Fig. 4.21. Two tests were made for checking correctness of observer working. First is standard – the same input signal goes to the observer and to the plant, which has different initial conditions. Results are introduced in Fig. 4.22. The second test is to use the output of observer as input to the controller. It allows verify the working of whole system with observer. Results are introduced in Fig. 4.23.

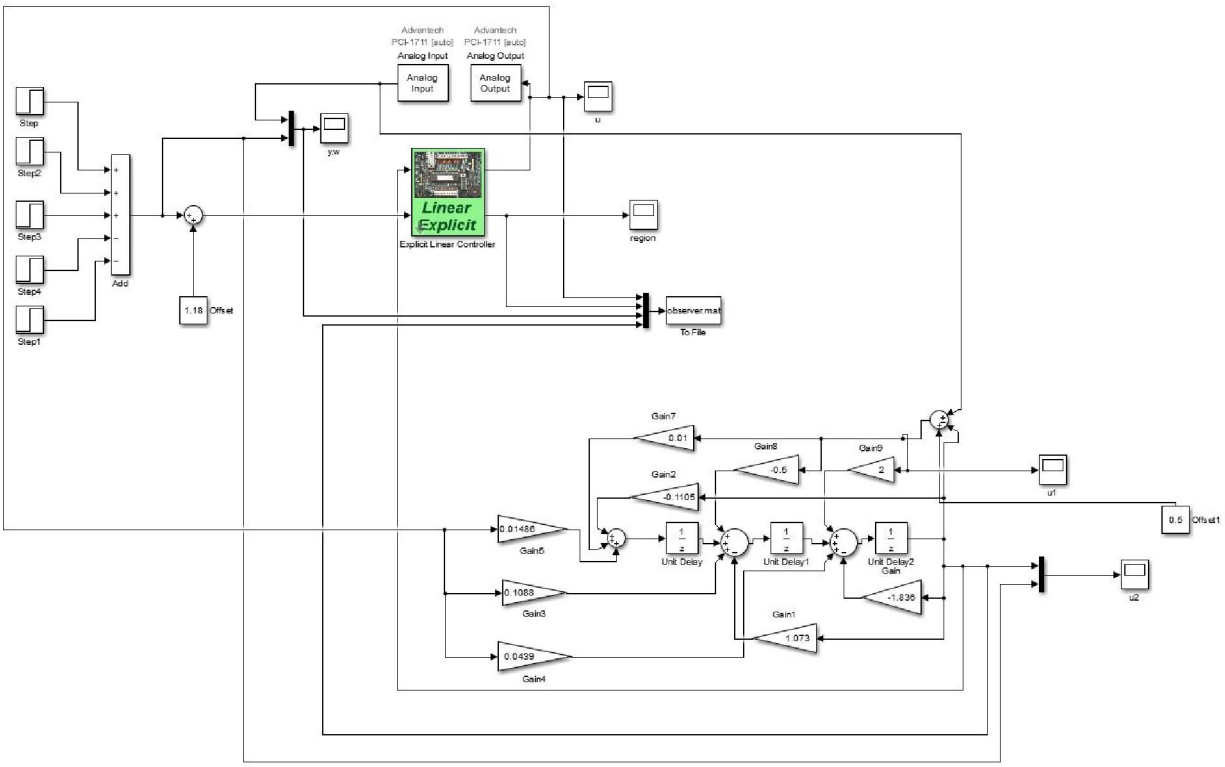

Fig. 4.21: Scheme of system with observer

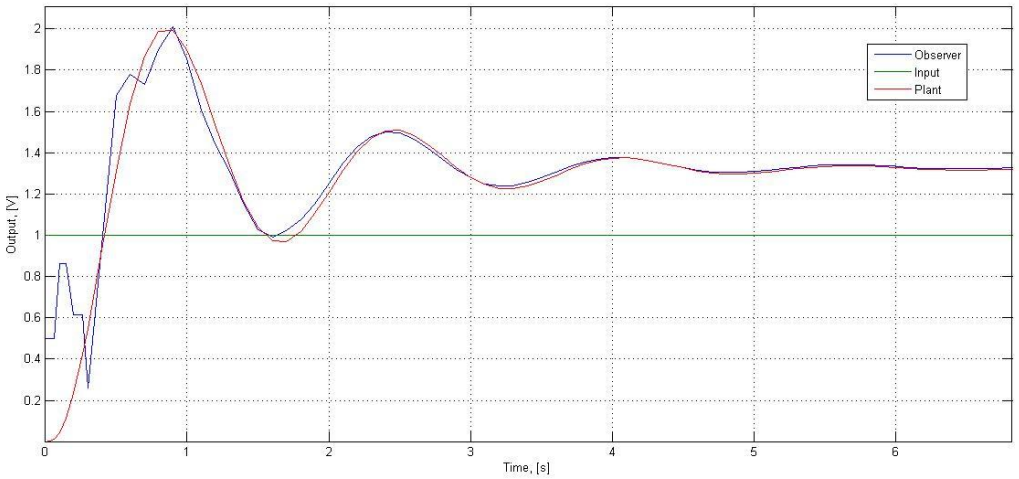

Fig. 4.22: Step-response of the observer and the plant

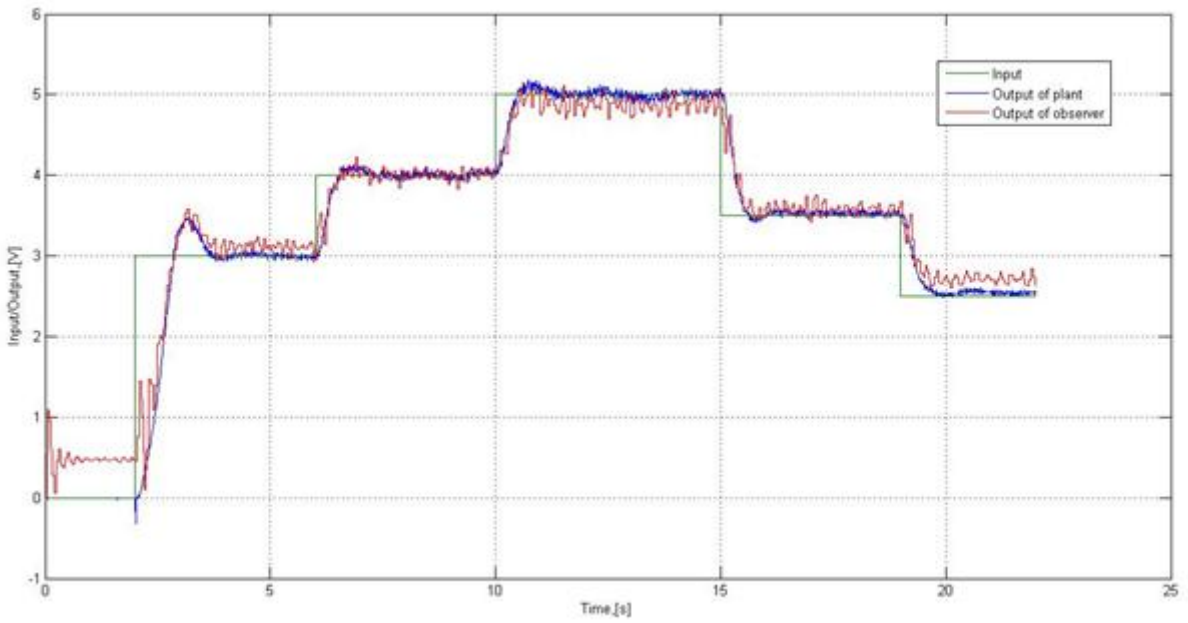

Fig. 4.23: Comparison of plant and observer reactions on input sequence

Processes in the plant and in the observer are relatively same. Deriving of states goes correctly, therefore, control doesn't loose in quality. Synthesized observer can be used for state estimation of this plant.

5 IMPLEMENTATION ON MICROCONTROLLER

5.1 Choosing of microcontroller

To implement synthesized controller on microcontroller (MC), it is necessary to consider some requirements for it. Main parameters of the system that affect the selection of MC are: operating speed, memory size, existence of ADC and DAC inside.

Because before implementing the control algorithm the amount of needed program memory is unknown, it is useful to choose MC with relatively large memory size. System sampling time is 0.01 s, and it should be enough to make all steps of control algorithm. Considering these requirements, MC STM32F100RBT6B BY ST Microeletronics [16] was chosen for the task. It is good choice, because this MC has evaluation board with embedded programmer ST-link, that uses USB-interface. This MC is chosen for setting and debugging of the process control, but there are some easier version of MC, made by ST Microelectronics. They can replace chosen one without serious changings, if STM32F100RBT6B is too mighty for the task.

The evaluation board for STM32F100RBT6B is STM32VLDISCOVERY [17]. Power supply for it is +5V or +3.3V. The board has user button, reset button, two light-emitting diodes, and quartz resonator [16].

Microcontroller STM32F100RBT6B has next characteristics:

- Core: ARM 32-bit Cortex-M3 CPU;
- Maximal frequency: 24 MHz;
- Memory: 128 Kb – flash, 8 Kb – SRAM
- Periphery: 12 timers, 12-bit ADC and DAC, I/O interfaces SPI, USART and I2C
- 3 16-bit I/O ports

Electrical parameters:

- Supply voltage: +3.3 or +5.0V
- Maximal supply current: 150 mA
- Temperature range: -40 ÷ +85 ^{o}C
- Maximal output current for one port: 25 mA
- Maximal total output current: 25 mA
- Maximal supply power: 444 mW

5.2 Description of hardware elements

The MC is made on the basis of CPU ARM 32-bit Cortex-M3, developed for implementation in progressive systems with low energy consumption. Core includes also system of interruptions, system timer, debugging system. Address space is divided to flash program memory, static memory, and peripheral devices. Core has several buses for parallel operations, controller of interruptions with 240 vectors described. Connection with periphery is made by matrix of high-speed buses.

For connection of MC with the plant it is necessary to develop the interface scheme. Microcontroller STM32F100RBT6B has embedded ADC and DAC, they will be used for this interface scheme. It is necessary to lead their ranges to ranges of input/output signals of the plant (tab. 3.2). For reducing of the range voltage divider can be used. To extend the range operational amplifier LM258A is used. It has next basic characteristics [18]:

- Slew rate: 0.3 V/us
- Supply voltage: 3 ÷ 32 V
- Input voltage range: -0.3 ÷ 32 V
- Output voltage range: 0 ÷ U_{cc}-1.5 V
- Output current: up to 30 mA

Also it is necessary to use capacitors for filtration.

5.3 Development of interface scheme

5.3.1 Power supply

For correct working of the system it is necessary to use stable +5V voltage supply for MC and not less than +11.5V voltage supply for operational amplifier. The task of autonomous work isn't set, because of it MC can use USB-connection for voltage supplying. For amplifier the supplying voltage for DC motor (+24 V) is used.

5.3.2 Input signal interface scheme

Input signal is signal running from the plant to the MC. Necessary information for calculations:

$U_{out.plant} = 0 \div 10$ V;

$U_{in.ADC} = 0 \div 3$ V;

$I_{in.ADC} = 0 \div 5$ mA;

In all calculations the maximal value is the upper value from the range for given parameter.

Range of the plant output is wider, than ADC range. The reducing of the range scheme is needed. Scheme for calculations is introduced on Fig. 5.1.

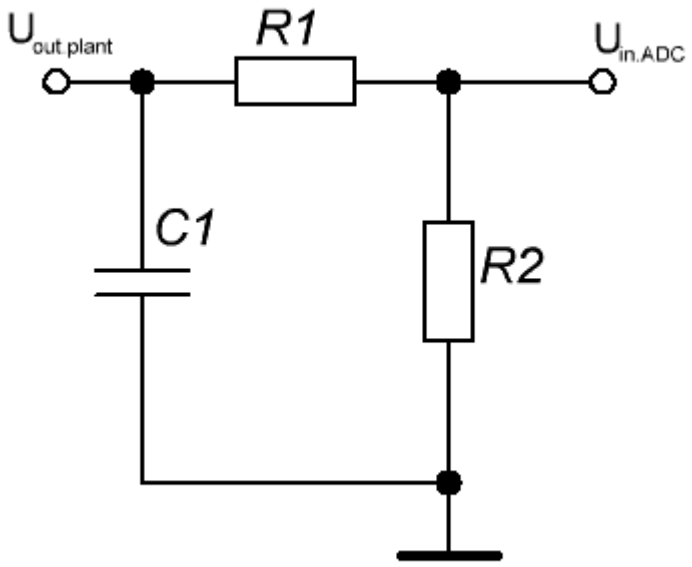

Fig. 5.1: Scheme for calculations of reducing range

Between supply and ground capacitor with C_1 = 0.1 uF is included. It is for high-frequency noise filtering.

To lead the range of the plant output to the ADC range the voltage divider is used.

$$U_{in.ADC.max} = \frac{R_2}{R_1 + R_2} U_{plant.max} \tag{5.1}$$

$$I_{in.ADC.max} \cdot R_1 \geq (U_{plant.max} - U_{in.ADC.max}) \tag{5.2}$$

From (5.2)

$$R_1 \geq \frac{U_{plant.max} - U_{in.ADC.max}}{I_{in.ADC.max}} = \frac{10V - 3V}{5mA} = 1.4\ kOhm$$

Let's accept R_1 = 2.4 kOhm.

$$\frac{U_{in.ADC.max}}{U_{plant.max}} = \frac{R_2}{R_1 + R_2} = 0.3;$$

So then $R_2 = 1\ kOhm$.

Real voltage divider:

$$\frac{R_2}{R_1 + R_2} U_{plant.max} = 0.294 \cdot 10V = 2.94V < U_{in.ADC.max}$$

Calculated divider is usable for this task.

Real interface scheme, ready for implementation, is introduced in Fig. 5.2.

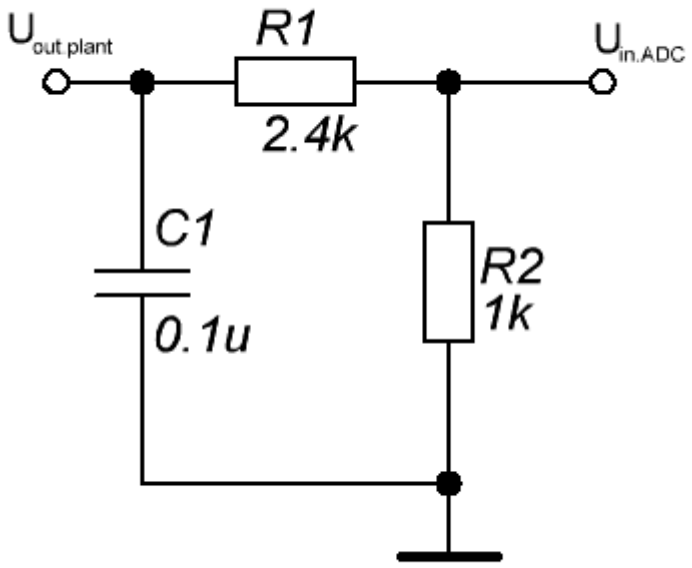

Fig. 5.2: Scheme for implementation of reducing range

5.3.3 Output signal interface scheme

Output signal is signal running from the MC to the plant. Necessary information for calculations:

$U_{in.plant} = 0 \div 10$ V;

$U_{out.DAC} = 0 \div 2.93$ V;

In all calculations the maximal value is the upper value from the range for given parameter.

Range of the plant output is narrower, than DAC range. The extending of the range scheme is needed. Non-inverting circuit is used. Scheme for calculations is introduced on Fig. 5.3.

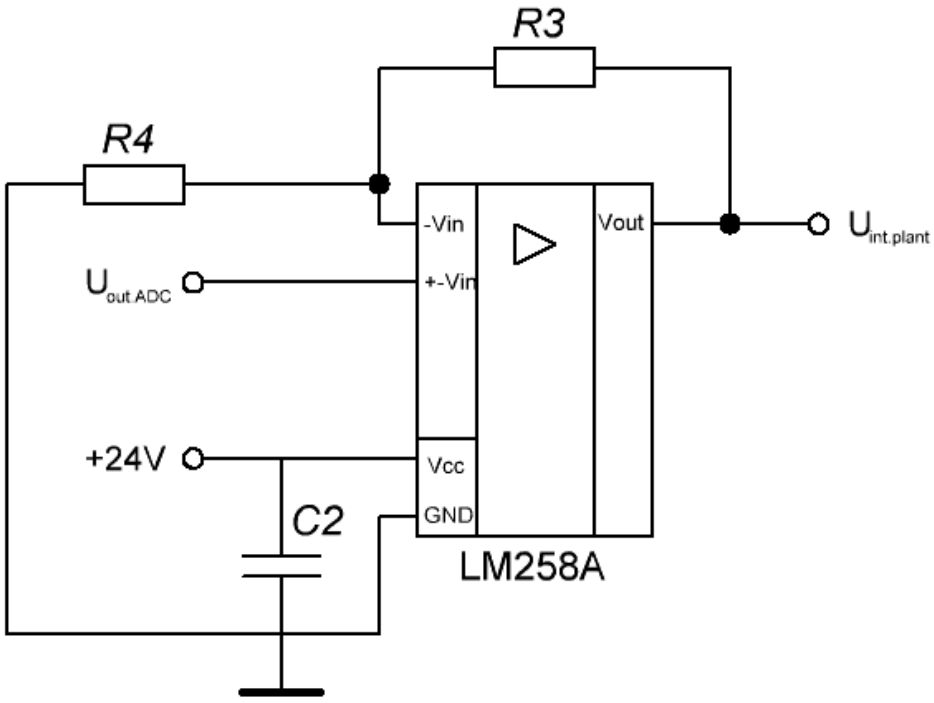

Fig. 5.3: Scheme for calculation of extending range

Between supply and ground capacitor with C_2 = 0.1 uF is included. It is for high-frequency noise filtering.

To lead the range of the plant input to the DAC range parameters of non-inverting amplifier are calculated.

$$U_{in.plant} = U_{out.DAC} \cdot (1 + \frac{R_3}{R_4}) \tag{5.3}$$

From (5.3) is got

$$\frac{R_3}{R_4} = \frac{U_{in.plant}}{U_{out.DAC}} - 1 = 2.413$$

Let's accept R_3 = 2.4 kOhm, R_4 = 1 kOhm

Real interface scheme, ready for implementation, is introduced in Fig. 5.4.

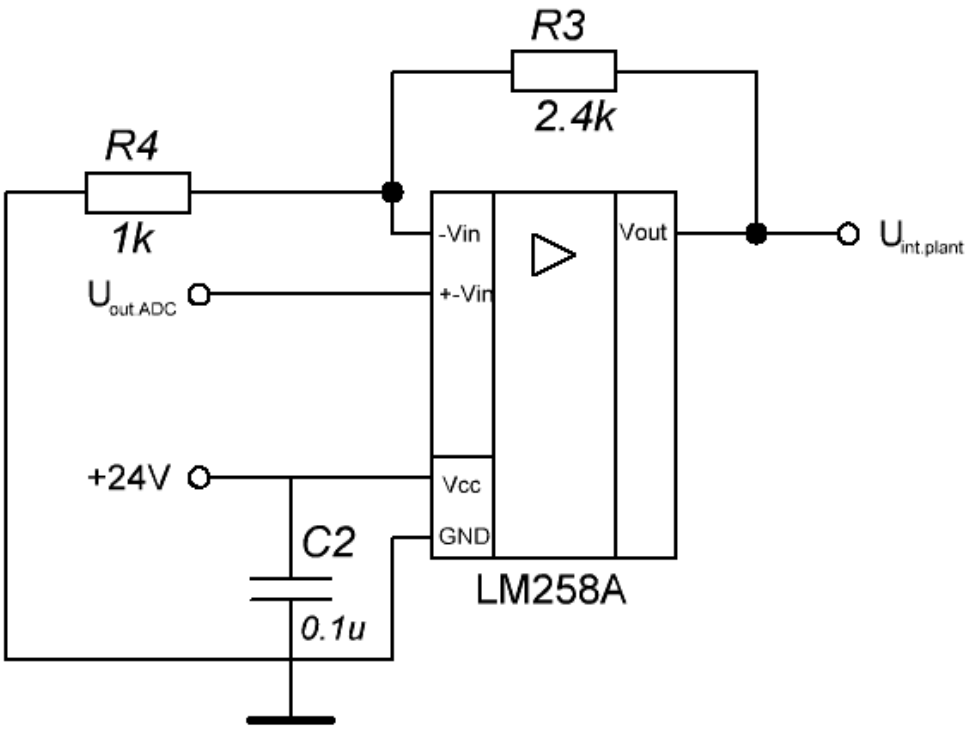

Fig. 5.4: Scheme for implementation of extending range

5.4 Development of libraries

5.4.1 Module of internal ADC

In header file description of ADC class is located. There are some internal variables: *IsNewConvertion* (indicates the end of conversion), *ConversionData* (digital data from the last conversion).

The next methods are described in the class: *ADC_Class*() (constructor of the class), *Init*() (initialization function), *Start*() (run ADC), *Stop*() (stop ADC working).

Executing file contains realization of methods and interruptions handler.

ADC_Class() – class constructor, makes zeroing of internal variables.

void *Init*(void) – sets up internal registers of MC. Realization of this function contains several blocks: clocking of periphery (port A, DAC, timer TIM2, DMA controller DMA1), setting of port A pin as an analog input, permitting of interruptions, setting of TIM2 to frequency (conversion every 0.01 s), setting of DMA (bus between ADC registers and variable *ConversionData*), setting of ADC [19], [20].

void *Start*(void) – start function. It makes calibration of ADC, starts it, starts timer TIM2.

void *Stop* (void) – stop function. It stops ADC and timer TIM2.

void *ADC1_IRQHandler*() – interruption handler. It sets flag *IsNewConvertion* to 1, if the new conversion ends.

5.4.2 Module of internal DAC

In header file description of DAC class is located. There is one internal variable: *OutputValue* (variable for storing the output value).

The next methods are described in the class: *DAC_Class*() (class constructor), *Init*() (initialization function), *SetOutput*() (function of setting output to desired value).

Executing file contains realization of methods and interruptions handler.

DAC_Class() – class constructor, makes zeroing of internal variables.

void *Init* (void) – sets up internal registers of MC. During the function working, it sets up next things: activation of PA4 pin of port A as an analog output, clocking of port A and DAC, choosing of software trigger to value transmitting to DAC, starting of DAC.

void *SetOutput* (int value) – function of setting output to desired value. Write the function parameter value to output DAC registr and sets output trigger to 1. Also it saves this value to internal variable *OutputValue*.

5.5 Development of control program

Control program is divided on several parts. The first part is initialization and starting of ADC and DAC. Also in this part all necessary variables are introduced.

The second part – main program loop. It is executed till the turning off the power. The main function of this part is to receive the signal, handle it and transmit to the plant the appropriate control action.

More detailed description of the second part working. Every 0.01 second ADC reads the input signal, digitalizes it and saves to variable *ConversionData*. After setting the flag *IsNewConversion* to 1, handling starts. On the basis of *expconobs* function and equation (4.5) vector of current state is estimated and the control action is calculated. Later on this control action is led to appropriate range and goes to DAC. Current state vector becomes the past state vector. Flag *IsNewConversion* goes to 0.

Control program has comments for each action. Reference action is set inside the program by setting the value of parameter *r[0]*.

5.6 Regulation with microcontroller

This part of work is connected with physical connection of all system parts. According to the Fig. 5.2 and Fig. 5.4 pins of MC were connected with input and output wires of the plant. After that the control program including all written libraries was loaded into the MC. For this operation was used *Eclipse IDE* for C/C++ developers. The detailed instructions about installation of this development environment are in appendix on CD ROM.

There is one important thing in the process of loading the control program into the MC. During the loading all registers of MC can get random values. Because of it unexpected rotations of DC motor can appear. In practice these rotations are very strong. It is necessary to connect wires from MC to the plant after loading the program, but not earlier.

The result of controlling the plant with MC confirmed that results, obtained with *Matlab*, are same with MC regulation. The results are introduced in Fig. 5.5.

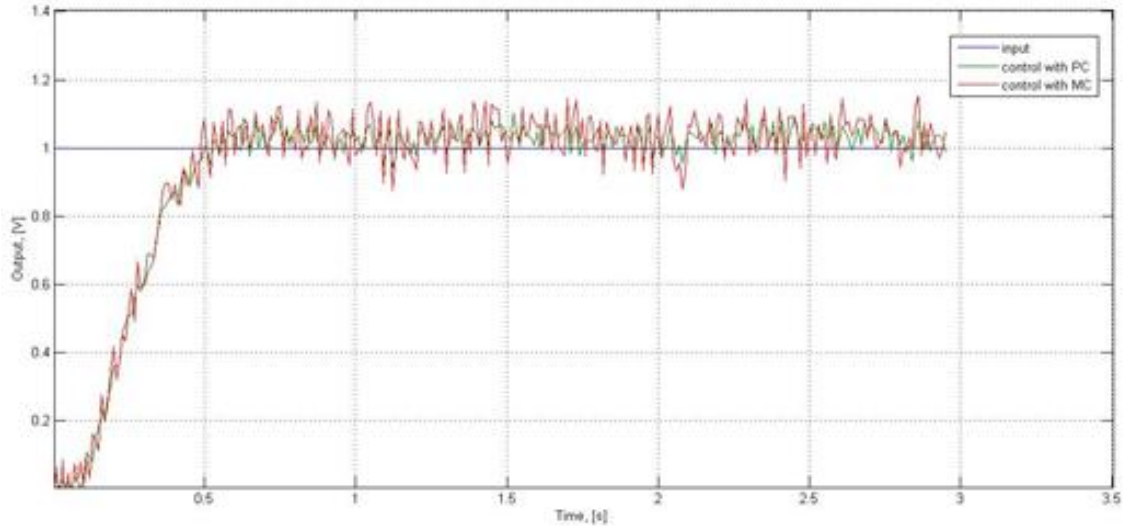

Fig. 5.5: Step-response of closed loop with MC

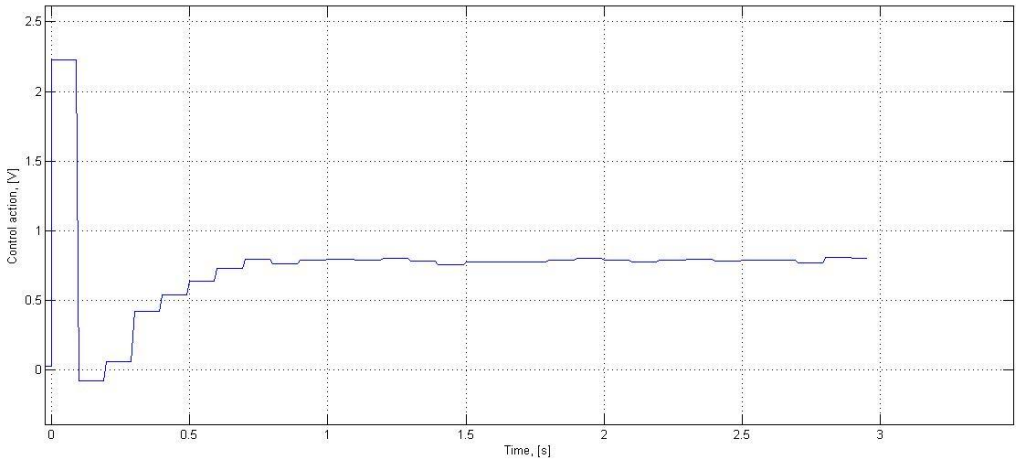

Fig. 5.6: Control action in time with MC

The result of regulation with MC is satisfactory. Comparison with PC version of control shows, that dynamic of system doesn't change, but some noise is added.

CONCLUSION

As a result of this work the controller for system “DC motor – generator” was developed and explored.

During the work, system identification on the basis of quadratic criterion minimization was made. As a result of identification, static and dynamic behaviors were derived. The mathematical model was obtained.

On the basis of identification data explicit MPC regulator was synthesized. *Hybrid Toolbox* for *Matlab* helped with it. Obtained control law was implemented for control of the system “DC motor – generator” in *Matlab Simulink*. After that, exploration of synthesis parameters influence on the dynamic behavior was made. On the basis of this exploration the optimal values for parameters were obtained. Also explicit MPC was compared with PID control and robust control. The comparison showed that for the system explicit MPC is preferred control method.

The synthesized controller was realized on microcontroller STM32F100RBT6B with the evaluation board STM32VLDISCOVERY. For connection with the plant interface schemes were created. The resulting device successfully realizes explicit MPC controller without using the PC.

The further explorations in this topic can be connected with implementation of control algorithms to multi-channel systems. The existing program can be improved by organizing the changing of reference value with button.

LITERATURE

[1] Model Predictive Control. [Online]. Available: http://en.wikipedia.org/wiki/Model_predictive_control. [Accessed 8.05.2015]

[2] MAGNI, Lalo. Nonlinear model predictive control: towards new challenging applications. Berlin: Springer, 2009, s. 345-369. ISBN 978-3-642-01094-1

[3] Pulp and paper industry. [Online]. Available: http://www.eposc.com/index.php?option=com_content&task=view&id=27&Itemid=38&lang=en [Accessed 8.04.2015]

[4] MACIEJOWSKI, Jan Marian. Predictive control: with constraints. New York: Prentice Hall, 2002, xviii, 331 p. ISBN 02-013-9823-0.

[5] VOSTRIKOV, A. S., FRANTSUZOVA, G. A., GAVRILOV, E. B. Fundamentals of continuous and discrete control systems theory. Novosibirsk: NSTU, 2008, 476 p. ISBN 978-5-7782-1129-9

[6] BEMPORAD A., Explicit model predictive control. [Online]. Available: http://www.seas.upenn.edu/~ese680/papers/explicit_linear_mpc.pdf. [Accessed 18.04.2015]

[7] BEMPORAD, A. Explicit Model Predictive Control via Multiparametric Programming. [Online]. Available: http://cse.lab.imtlucca.it/~bemporad/teaching/mpc/stuttgart_2012/3-explicit_linear_mpc.pdf. [Accessed 18.04.2015]

[8] BEMPORAD, A., MORARI, M., DUA, V., PISTIKOPOULOS, E. The explicit linear quadratic regulator for constrained systems. [Online]. Available: http://robot2.disp.uniroma2.it/~zack/ftp/users/Tingshu/papers/BemporadExplicitAuto.pdf. [Accessed 26.04.2015]

[9] CAGIENARD, R., GRIEDER, P., KERRIGAN, E.C., MORARI, M. Move Blocking Strategies in Receding Horizon Control. [Online]. Available: http://www2.ee.ic.ac.uk/publications/p5072.pdf. [Accessed 1.05.2015]

[10] TONDEL, P., JOHANSEN, T.A., BEMPORAD, A. Evaluation of piecewise affine control via binary search tree. Automatica 39 (2003).

[11] MODRLAK, O., HUBKA, L., GARTNER, S., WORLITZ, F. Rotational Speed Control DC Motor with load. Liberec: TUL, 2012.

[12] MODRLAK, O., HUBKA, L., GARTNER, S. Rotational Speed Control by elastic Coupling. Liberec: TUL, 2012.

[13] KOPAL, M. Explicit Model Predictive Control and its application. Liberec, 2014. Diploma thesis. TUL.

[14] BEMPORAD, A. Hybrid Toolbox User's Guide. [Online]. Available: http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/. [Accessed 18.04.2015]

[15] MODRLAK, O., HUBKA, L. Robust and Fuzzy Control. Lectures. Liberec: TUL, 2014.

[16] STMicroelectronics STM32F100x4 STM32F100x6 STM32F100x8 STM32F100xB. [Online]. Available: http://www.st.com/web/en/resource/technical/document/datasheet/CD00251732.pdf. [Accessed 5.05.2015]

[17] STMicroelectronics STM32VLDISCOVERY STM32 value line Discovery. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/CD00267113.pdf. [Accessed 5.05.2015]

[18] Texas Instruments Dual Operational Amplifiers LM158, LM158A, LM258, LM258A, LM358, LM358A, LM2904, LM2904V. [Online]. Available: http://www.farnell.com/datasheets/1874566.pdf. [Accessed 1.05.2015]

[19] STMicroelectronics RM0041 Reference manual STM32F100xx advanced ARM-based 32-bit MCUs. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/CD00246267.pdf. [Accessed 5.05.2015]

[20] STMicroelectronics PM0056 STM32F10xxx (20xxx) (21xxx) (L1xxxx) Cortex-M3 programming manual. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/programming_manual/CD00228163.pdf. [Accessed 5.05.2015]

CD-ROM

The CD-ROM contains the following components:

- PDF file of diploma thesis;
  - Kirill_Lastochkin_Master_Thesis_2015.pdf
- Header files and source files of libraries;
  - ADC_Class.h
  - ADC_Class.cpp
  - DAC_Class.h
  - DAC_Class.cpp
  - expcon.h
  - expcon.c
- Source file of microcontroller control program;
  - main.cpp
- ZIP file with all necessary setup files to install Eclipse on the PC
  - Documentations.zip
- M file with script of parameters setting
  - explicitMPC.m