

Model predictive control of magnetic levitation system

Prediktivní řízení magnetické levitace

Bc. Martin Malý

Master's thesis
2015



Tomas Bata University in Zlín
Faculty of Applied Informatics

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin Malý**
Osobní číslo: **A13434**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Automatické řízení a informatika**
Forma studia: **prezenční**

Téma práce: **Prediktivní řízení magnetické levitace**
Téma anglicky: **A Model for the Predictive Control of a Magnetic Levitation System**

Zásady pro vypracování:

1. Vypracujte literární rešerši zabývající se algoritmy prediktivního řízení. Zvláštní pozornost věnujte algoritmům použitelným pro rychlé systémy.
2. Na základě matematicko-fyzikální analýzy vytvořte matematický model laboratorní soustavy systému CE152 magnetická levitace.
3. V prostředí MATLAB / Simulink vytvořte model systému CE152 magnetická levitace. Model porovnejte s reálným systémem a případně jej upravte, aby byla dosažena co nejlepší shoda v chování.
4. Navrhněte vhodné algoritmy prediktivního řízení pro simulinkový model systému CE152.
5. Algoritmy prediktivního řízení, které prokázaly dobré simulační výsledky, otestujte na reálném systému CE152 a zhodnoťte dosažené výsledky.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. KWON, W a S HAN. Receding horizon control: model predictive control for state models. London: Springer, 2005, xiv, 380 s. ISBN 1-84628-024-9.
2. MACIEJOWSKI, Jan Marian. Predictive control: with constraints. 1st ed. Harlow: Prentice Hall, 2002, xviii, 331 s. ISBN 0201398230.
3. CAMACHO, E a Carlos BORDONS. Model predictive control. 2nd ed. London: Springer, c2007, xxii, 405 s. ISBN 1-85233-694-3.
4. KVASNICA, Michal. Real-time model predictive control via multi-parametric programming: theory and tools. Saarbrücken: VDM Verlag, 2009, xiv, 249 s. ISBN 978-3-639-20644-9.
5. ALLGÖWER, Frank a Alex ZHENG. Nonlinear model predictive control. Basel: Birkhäuser Verlag, 2000, vii, 472 s. ISBN 3764362979.
6. NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, c2001, xvii, 785 s. ISBN 3-540-67369-5.
7. QIN, Yemei, Hui PENG, Wenjie RUAN, Jun WU a Jiacheng GAO. A modeling and control approach to magnetic levitation system based on state-dependent ARX model. Journal of Process Control. 2014, vol. 24, issue 1, s. 93-112. DOI: 10.1016/j.jprocont.2013.10.016. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S0959152413002230>
8. BÄCHLE, Thomas, Sebastian HENTZELT a Knut GRAICHEN. Nonlinear model predictive control of a magnetic levitation system. Control

Vedoucí diplomové práce:

Ing. Petr Chalupa, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

27. února 2015

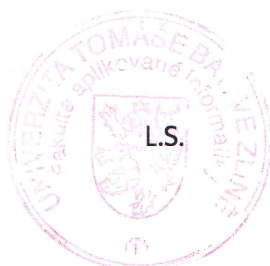
Termín odevzdání diplomové práce:

20. května 2015

Ve Zlíně dne 27. února 2015

doc. Mgr. Milan Adámek, Ph.D.

děkan



prof. Ing. Vladimír Vašek, CSc.

ředitel ústavu

THESIS AUTHOR STATEMENT

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

20.5.2015



.....

podpis autora

ABSTRAKT

Tato práce pojednává o problematice prediktivního řízení modelu reálného systému magnetické levitace CE152. Část práce byla věnována vytvoření modelu systému magnetické levitace, který by měl věrně vystihovat chování tohoto systému. Další částí bylo nalezení algoritmů použitelných pro tento řízený systém a jejich následná aplikace a testování na modelu a posléze i na reálném systému. Pro simulace a řízení bylo použito programové prostředí MATLAB/SIMULINK.

Klíčová slova: Magnetická levitace, Maglev, Prediktivní řízení, MPC, Řízení, Simulace, Modelování, CE152, Pozorovatel stavu

ABSTRACT

This thesis discusses issues of predictive control of the real-time system of magnetic levitation model CE152. One part of this thesis was dedicated to the creation of a model of the magnetic levitation system, that should sufficiently reflect behaviour of this system. Other part consisted in finding proper algorithms suitable for this controlled system, their application and testing on created model and later even on the real system. For simulations and control programming environment MATLAB/SIMULINK was used.

Keywords: Magnetic levitation, Maglev, Predictive control, MPC, Control, Simulation, Modelling, Regulation, CE152, State observer

ACKNOWLEDGEMENTS

At this point I would like to thank to my thesis supervisor Ing. Petr Chalupa, Ph.D. for his support, valuable advices and comprehensive guidance. I would also like to thank to my closest for their support during my whole study.

TABLE OF CONTENTS

PREFACE	9
I THEORY	9
1 MODELLING BASICS	11
1.1 MODEL BUILDING	11
1.2 MODEL VERIFICATION	11
1.3 MODEL VALIDATION.....	11
2 COMPUTER CONTROL	12
2.1 SAMPLING.....	13
2.1.1 Sampling a continuous-time linear state-space system	13
2.1.2 Sampling a continuous-time nonlinear state-space system	14
3 STATE ESTIMATION	17
3.1 KALMAN FILTERING.....	17
3.1.1 Kalman filter	17
3.1.2 Extended Kalman filter	18
3.2 MOVING HORIZON ESTIMATION.....	19
3.2.1 Finite impulse response filter	19
4 MODEL PREDICTIVE CONTROL AND ITS ALGORITHMS	22
4.1 PREVIEW	22
4.2 MODEL PREDICTIVE CONTROL BASICS	22
4.2.1 Model	23
4.2.2 Optimizer	24
4.3 REVIEW OF MODEL PREDICTIVE CONTROL ALGORITHMS.....	26
4.3.1 Linear model predictive control algorithms	26
4.3.2 Nonlinear model predictive control algorithms	27
4.4 MODEL PREDICTIVE CONTROL ALGORITHM FOR STATE-SPACE MODEL	27
4.4.1 Prediction with state-space models	27
4.4.2 Control law.....	29
II PRACTICAL PART	30
5 THE CE152 MAGNETIC LEVITATION SYSTEM SPECIFICATION..	32
5.1 THE CE152 MAGNETIC LEVITATION STRUCTURE	32
6 SYSTEM BEHAVIOUR	33
7 IDENTIFICATION OF THE SYSTEM	34
7.1 D/A CONVERTER	34

7.2	A/D CONVERTER	34
7.3	THE POSITION SENSOR.....	35
7.4	THE POWER AMPLIFIER	36
7.5	THE BALL AND COIL SUBSYSTEM.....	39
7.5.1	Two point calibration parameter estimation.....	41
7.5.2	The second order integrator	42
7.6	THE WHOLE SYSTEM MODEL.....	43
7.7	VERIFICATION AND VALIDATION OF THE NONLINEAR MODEL.....	43
8	LINEARISATION OF THE NONLINEAR MODEL.....	45
8.1	SELECTION OF AN OPERATING POINT	47
8.2	VERIFICATION AND VALIDATION OF THE LINEARIZED MODEL	47
9	DISCRETIZATION OF CREATED MODELS	48
9.1	DISCRETE LINEAR STATE-SPACE MODEL	48
9.2	DISCRETE NONLINEAR STATE-SPACE MODEL.....	48
9.2.1	Euler method solution.....	48
9.2.2	Runge-Kutta method solution.....	48
10	STATE OBSERVER	51
10.1	OBSERVABILITY AND CONTROLLABILITY	51
10.2	COMPARISON OF FILTERS.....	51
10.2.1	Simulations results.....	52
10.2.2	Performance results	54
11	MODEL PREDICTIVE CONTROL OF CE152 SYSTEM	55
11.1	CONTROL LAW	55
11.2	SIMULATION OF THE CONTROL WITH LINEAR MODEL.....	56
11.3	SIMULATION OF THE CONTROL WITH NONLINEAR MODEL.....	58
11.4	CONTROL OF THE REAL SYSTEM	60
11.5	COMPARISON WITH A SIMPLE CONTROLLERS.....	60
	CONCLUSION	63
	REFERENCES	64
	LIST OF ABBREVIATIONS.....	65
	LIST OF FIGURES.....	68
	LIST OF TABLES.....	69
	LIST OF APPENDICES	70

PREFACE

Levitation is the process by which a tangible object is held aloft in a stable position without mechanical support. Levitation is accomplished by overcoming the gravitational force. If acting forces are balanced, the object will levitate. If sum of acting forces wouldn't be zero, object would fall down or be pulled to the more powerful source. There are a few possibilities how to levitate an object for example acoustic levitation, optical levitation, electric or radio-frequency levitation and of course magnetic levitation. Magnetic levitation is the most widely used. The reasons are simple. First reason is general. When an object levitates, there is no mechanical contact between the object and the actuator, so there is no friction either. Other reason is that this principle is well described and "easy" to use for most industrial purposes. Nowadays, there are high speed trains, suspension bearings, flywheels and other applications based on magnetic levitation technology.

Magnetic levitation (maglev) is using theory of magnetic fields and their behaviour. There are more principles. Probably the best principle is based on using cooled superconductors embedded in the magnetic field (technology used in high speed trains). Other option is to use diamagnetic or conductive object and put it into the magnetic field. Variant with conductive material is used in the CE152 magnetic levitation system.

Examined magnetic levitation system is the CE152 magnetic levitation model offered by company Humusoft. This model is being used for teaching system dynamics and control engineering principles. System consist from these main parts: a coil, metal ball and an inductive sensor. As electric current passes through the coil, it creates magnetic field. This field acts on metal ball and create magnetic field of opposite orientation. Metal ball is then attracted to the coil. To levitate with the ball, acting forces must be balanced. So based on desired position of the ball the amount of electric current must be controlled. Model is controlled from PC and it is connected via PCI card with A/D and D/A converters.

For simulations and control was used standard personal computer with programming environment MATLAB/Simulink. MATLAB is a high performance software package for scientific and numeric computation, signal processing and graphics. Simulink is a block oriented environment for simulation of dynamic systems and it extends the power of MATLAB environment. The CE152 magnetic levitation system can be controlled by classic conventional regulation principles and methods, but thanks to its specifications more complex methods brings some benefits to the control. In this case a model predictive control was used.

Model predictive control (MPC) is a popular technique for the control, which replaces conventional control techniques. It became popular primarily in industry and later even in academic research community. Basic difference between conventional control and MPC is that MPC doesn't observe only the current values but also the future values and remembers the past values. This approach is more natural and reminds human behaviour. In this thesis was implemented basic predictive control algorithm based on state-space model. This algorithm was applied primarily to identified nonlinear model of CE152 magnetic levitation system and later to the real system. Final results of MPC control are compared with PID regulation in simple form and advanced form, which was designed by Humusoft as a demonstration of the magnetic levitation system control.

I. THEORY

1 MODELLING BASICS

Modelling is used for analysis, prediction and control of a system. There are two types of models physical (real) and mathematical(abstract). Physical models can be small size models with same structure and behaviour as original system or be analogical to original system and work on other principle which follow similar laws. Mathematical models are mostly expressed by equations (algebraic, differential etc.), while unimportant properties are neglected and only significant properties are regarded.

Models can be categorized into different categories based on their characteristics. In terms of linearity models can be divided into linear or nonlinear, in terms of the number of inputs and outputs into one-dimensional or multidimensional. According to a further categorization models can be also continuous or discrete, deterministic or stochastic, time-variant or time-invariant and so on [1].

1.1 Model building

Two base approaches are used, analytical and empirical. Models based on empirical approach (experimental model) express only relations between input and output but they don't have any information about system's internal bonds or states. This models are obtained through measurements and observed data. Second approach is creating a model based on analytical approach (state space model) which refers to system's inner structure. This method allows to split up the system into subsystems, whose properties are well understood from previous experience. This means that we rely on "laws of nature" and well-established relationship. Created subsystems are joined mathematically and a model of the whole system is obtained. In practice a combination of both approaches is used. In conclusion, we can say that in the construction of a model, three base entities are involved, data itself, set of candidate (verified) models and a criterion to determine the best model (model validation) [1].

1.2 Model verification

Verification is like debugging, it is intended to ensure that the model does what it should. There are no strict principles and methods and it is up to personal discretion, how deep should verification go. A good practice is, that person who creates a model, also creates documentation at the same time. This helps to avoiding some fundamental errors. A deeper verification can be given through step by step checking. It is also advisable, pass the work to an expert on the process and let him to analyse it [1].

1.3 Model validation

When model is verified it remains to test whether this model is valid for its purpose. Such tests are known as model validation and they involve various procedures to assess how the model relates to observed data and to its intended use. Good results are the reason for acceptance of the model. Needless to say, created model doesn't give a final and true description of system, rather that he gives good enough description of certain aspects that are of particular interest to use [1]. Base validation can be obtained by comparison of the real process response and the model response to the same input signal. Used comparative method can be visual (from simulation cycles) or statistical.

2 COMPUTER CONTROL

There is a lot of techniques for controller design. According to what needs to be controlled, measured and control variables are chosen. Also inner states are taken into account. As was said before these are mostly unmeasurable so they are reconstructed by so called state observer. Creation of the controller can be done as a program, written and run on the PC. Then it is talked about so-called computer control. Good source devoted to the computer control problematic is provided in [2]. This kind of control is used for the purpose of control of the studied CE152 magnetic levitation system.

Computer control is conceived as it is presented in Fig. 2.1. Measured controlled signals from process, that we need to control, entering the PC through A/D converter. PC performs control algorithm and sending control signal to D/A converter and then the signal is going into the process actuators. Computer control and also other control approaches has to deal with some difficulties. Since there are used A/D and D/A converters the problem of sampling must be solved. Sampling represents difficulties with a conversion of continuous signal to discrete signal. Due to the conversion, an aliasing phenomenon appears. Aliasing causes overlap of frequency spectra of the sampled signal and with this associated loss of information. With pre-filtering and a proper sampling frequency(Shannon-Nyquist-Kotelnikov theorem) aliasing problem disappears. Controlled signals are measured by sensors. Sensors are usually imperfect. Different sensors have different characteristics, they are varies in their static characteristics (linearity, effective measurable range, sensitivity, accuracy or resolution) and in dynamic characteristics (response, noise level etc.). Some sensor errors (e.g. noise) can be compensated by filtering or special connections of the sensor. Sampling and imperfection of sensor aren't the only problem, there is other and probably much bigger problem. It is computational delay. Computational delay depends critically on the details of the algorithm and can be shorten by reorganization and better implementation of the source code. Great role in computer control plays also the operator interface. The interface should display some of the reasonable informations and control elements like visualization of the process, desired and actual value of a controlled variable, graph of the regulation cycle, adjustable control parameters, start and stop buttons and a lot more other optional visual and control elements.

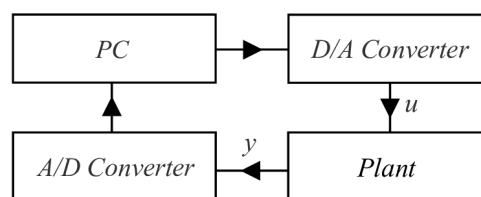


Fig. 2.1 Principal scheme of the computer control system

Nowadays, computer control have its undisputed place in the area of control. Even when computers don't handle the control algorithm, they are still used for simulations and graphic representation of the process with using the measurements. Regarding to the complexity of the predictive control, computer control represents the powerful tool for the implementation of predictive control algorithms.

2.1 Sampling

It is assumed that further in thesis models in the state-space form will be used, for that reason following lines are dedicated to the sampling (discretization) of continuous-time state-space system and the sampling of differential equations. This procedure is necessary, because PC's can work only with a discrete-time values of the process and used algorithms are also designed in a discrete form.

2.1.1 Sampling a continuous-time linear state-space system

It is assumed, that discrete-time state space system with a step size Δt looks as follows

$$\begin{aligned} x(k\Delta t + \Delta t) &= \Phi x(k\Delta t) + \Gamma u(k\Delta t) \\ y(k\Delta t) &= Cx(k\Delta t) + Du(k\Delta t) \end{aligned} \quad (2.1)$$

where

$$\begin{aligned} \Phi &= e^{A\Delta t} \\ \Gamma &= \int_0^{\Delta t} e^{sA} ds B \end{aligned} \quad (2.2)$$

There are more solutions, how get the matrices Φ and Γ . Φ can be given by inverse Laplace transform of expression $(sI - A)^{-1}$ and then Γ is computed by solving integral $\int_0^{\Delta t} e^{sA} ds B$. Other possibility is consider that the matrices satisfy the equation:

$$\frac{d}{dt} \left(\begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix} \right) = \begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \quad (2.3)$$

where I is a unit matrix of the same dimension as the number of inputs. The matrices $\Phi(\Delta t)$ and $\Gamma(\Delta t)$ for the sampling period Δt can be obtained from the block matrix

$$M = \begin{bmatrix} \Phi(\Delta t) & \Gamma(\Delta t) \\ 0 & I \end{bmatrix} = \exp \left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \Delta t \right) \quad (2.4)$$

If the second solution is used, then only the computation of the matrix exponential needs to be done. One way how to do it, was mentioned above and it is to do the inverse Laplace transform:

$$e^{M\Delta t} = \mathcal{L}^{-1}(\lambda I - M)^{-1} \quad (2.5)$$

Second is to find eigenvalues and eigenvectors of the matrix, find the fundamental matrix solution and eigenvectors matrix inverse and multiplying all together. Whole procedure is discussed below.

Eigenvalues are given through solving:

$$\det(\lambda I - M)^{-1} = 0 \quad (2.6)$$

Determined eigenvalues $\lambda_{1\dots n}$ are then used to compute eigenvectors, which are computed for each eigenvalue by solving the equation:

$$(M - \lambda_i I)X = 0 \quad (2.7)$$

where X represents vector of searched eigenvector values, which are determined by solving the three equations of three unknown variables. Let's consider following eigenvector matrix:

$$V = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix} \quad (2.8)$$

and matrix D , which represents diagonal matrix of exponential functions with eigenvalues as an exponents and looks like this:

$$D = \begin{bmatrix} e^{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e^{\lambda_n} \end{bmatrix} \quad (2.9)$$

then the final solution of matrix exponential is:

$$e^{M\Delta t} = V * D * V^{-1} \quad (2.10)$$

Output value y represents only gain of the state variable x , so once variable x is discrete, output value y is discrete too. Detailed procedure is given in [2].

2.1.2 Sampling a continuous-time nonlinear state-space system

Nonlinear state-space system can be represented by following equation.

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= g(x, u) \end{aligned} \quad (2.11)$$

The state variables of the system are described by ordinary differential equation (ODE). The way how to get the values of the state variables in a certain discrete-time is to use one of the methods to their solving. Probably the simplest method is Euler method and very popular is method Runge-Kutta. Both method will be discussed further for the first and second order ODEs.

Euler method

The Euler method with a respect to definition of nonlinear state-space equation (2.11) for first order ODE is represented as:

$$x_{n+1} = x_n + \Delta t f(x_n, u_n) + O(h^2) \quad (2.12)$$

It is a first-order method, which means that the local error (error per step) O is proportional to the square of the step size (Δt). The reason why, can be presented by Taylor theorem:

$$f(x_n + \Delta t) = f(x_n) + \sum_{k=1}^m \frac{\Delta t^k}{k!} f^{(k)}(x_n, u_n) + O(h^m + 1) \quad (2.13)$$

Because here is Euler order $m = 1$ then $O(h^2)$. Now let's presume that we have second order ODE. It is possible to use Euler method for higher order by using midpoint method. At first instead of taking full step is taken half step.

$$\tilde{x}_{n+1} = x_n + \frac{\Delta t}{2} f(x_n, u_n) \quad (2.14)$$

$$x_{n+1} = x_n + \Delta t f(\tilde{x}_{n+1}) = x_n + \Delta t f(x_n, u_n) + \frac{\Delta t^2}{2!} f'(x_n, u_n) + O(h^3) \quad (2.15)$$

Generalizations of this scheme, with more intermediate steps, leads to the family of Runge-Kutta methods.

Runge-Kutta method

This method have different versions, here will be presented classical RK4 version for first and second order ODE. Calculation of the first order ODE is done by solving the following equations:

$$x_{n+1} = x_n + \frac{dx_1 + 2dx_2 + 2dx_3 + dx_4}{6} \quad (2.16)$$

where:

$$\begin{aligned} dx_1 &= \Delta t f(x_n, u_n) \\ dx_2 &= \Delta t f\left(x_n + \frac{\Delta t}{2} dx_1, u_n\right) \\ dx_3 &= \Delta t f\left(x_n + \frac{\Delta t}{2} dx_2, u_n\right) \\ dx_4 &= \Delta t f(x_n + \Delta dx_3, u_n) \end{aligned} \quad (2.17)$$

If the second order system is given in a following form:

$$\ddot{x} = f(\dot{x}, x, u) \quad (2.18)$$

where we assume that $\dot{x} = v$ and $\dot{v} = F(v, x, u)$ then the computation can be separated into two linked computations in the same form as was defined for the first order RK4 method.

$$\begin{aligned} v_{n+1} &= v_n + \frac{dv_1 + 2dv_2 + 2dv_3 + dv_4}{6} \\ x_{n+1} &= x_n + \frac{dx_1 + 2dx_2 + 2dx_3 + dx_4}{6} \end{aligned} \quad (2.19)$$

where:

$$\begin{aligned} dx_1 &= \Delta t v_n \\ dx_2 &= \Delta t \left(v_n + \frac{dv_1}{2} \right) \\ dx_3 &= \Delta t \left(v_n + \frac{dv_2}{2} \right) \\ dx_4 &= \Delta t (v_n + dv_3) \end{aligned} \quad (2.20)$$

$$\begin{aligned} dv_1 &= \Delta t F(v_n, x_n, u_n) \\ dv_2 &= \Delta t F \left(v_n + \frac{dv_1}{2}, x_n + \frac{dx_1}{2}, u_n \right) \\ dv_3 &= \Delta t F \left(v_n + \frac{dv_2}{2}, x_n + \frac{dx_2}{2}, u_n \right) \\ dv_4 &= \Delta t F(v_n + dv_3, x_n + dx_3, u_n) \end{aligned} \quad (2.21)$$

Output value y in the case of discrete-time nonlinear state-space system represents also only gain of the state variable x , so like before once variable x is discrete, then output value y is discrete too.

3 STATE ESTIMATION

State variables are mostly unmeasurable and they need to be estimated if the state space form is used. For observation (prediction) of state variables are used so called filters. If anyone asks why filter, the answer is simple. Part of this estimator works in the same way as classic filter, which is the separation of the components of the mixture, here the measured signal and noise. Other part estimates state variables from measurable variables.

3.1 Kalman filtering

Kalman filters represent a form of predictor/corrector algorithm used extensively in control systems engineering for estimating unmeasured states of a process. The estimated states are often used as a part of a strategy for control law design. The purpose of the classical discrete-time Kalman filter is to provide the closed form recursive solution for estimation of linear discrete-time dynamic systems. In many cases examined dynamic systems are not linear so traditional Kalman filter is not suitable. However, there are some versions of Kalman filters, which can be applied for estimating nonlinear dynamical systems. One of these is Extended Kalman filter (EKF) or Unscented Kalman filter (UKF) [3].

3.1.1 Kalman filter

If it concedes, that model used in the KF is the same as the discrete-time model described by the equation (2.1) with addition of process and measurement noise, then equation looks like this

$$\begin{aligned}x_{k+1} &= \Phi x_k + \Gamma_u u_k + \Gamma_w w_k \\ y_k &= C_k x_k + D_v v_k\end{aligned}\tag{3.1}$$

where x_k represents the states, y_k represents the measurements, w_k is the process noise and v_k is the measurement noise. Because we have information only about measurement noise we assume that $w_k = v_k$. Whole process can be split into two steps: the prediction step, where the next state of the system is predicted and the update step, where the state is corrected with a participation of the measurement at actual time step. Detailed description of Kalman filtering principle could be found in [3].

- Prediction

$$\begin{aligned}x_k^- &= \Phi_{k-1} x_{k-1} + \Gamma_{u_{k-1}} u_k \\ P_k^- &= \Phi_{k-1} P_{k-1} \Phi_{k-1}^T + Q_{k-1}\end{aligned}\tag{3.2}$$

- Update

$$\begin{aligned}
v_k &= y_k - C_k(x_k^-) \\
S_k &= C_k P_k^- C_k^T + R_k \\
K_k &= P_k^- C_k^T S_k^{-1} \\
x_k &= x_k^- + K_k v_k \\
P_k &= P_k^- - K_k S_k K_k^T
\end{aligned} \tag{3.3}$$

where x_k^- and P_k^- are the predicted mean and covariance of the state. Q_k and R_k are covariances of w and v given as:

$$\begin{aligned}
Q_k &= E(w w^T) \\
R_k &= E(v v^T)
\end{aligned} \tag{3.4}$$

3.1.2 Extended Kalman filter

EKF extends the scope of Kalman filter to nonlinear processes by forming a Gaussian approximation to the joint distribution of the states x and the measurements y using a Taylor series based transformation. Taylor series can be only first order or higher order [3]. In this thesis first order EKF is used.

The filtering model used in the EKF is:

$$\begin{aligned}
x_{k+1} &= f(x_k, k) + w(k) \\
y_k &= g(x_k, k) + v_k
\end{aligned} \tag{3.5}$$

where x_k , y_k , w_k and v_k have the same meaning as it was presented for classical KF, f is then the dynamic model function and g is the measurement model function. Whole process is similar to classical KF with a difference in the computation of predictions. In EKF prediction is obtained from the dynamic model function $f(x_k, k)$. Whole process can be again separated into two parts:

- Prediction

$$\begin{aligned}
x_k^- &= f(x_{k-1}, k-1) \\
P_k^- &= F_x(x_{k-1}, k-1) P_{k-1} F_x^T(x_{k-1}, k-1) + Q_{k-1}
\end{aligned} \tag{3.6}$$

- Update

$$\begin{aligned}
v_k &= y_k - g(x_k^-, k) \\
S_k &= C_x(x_k^-, k) P_k^- C_x^T(x_k^-, k) + R_k \\
K_k &= P_k^- C_x^T(x_k^-, k) S_k^{-1} \\
x_k &= P_k^- + K_k v_k \\
P_k &= P_k^- - K_k S_k K_k^T
\end{aligned} \tag{3.7}$$

where Q_{k-1} and R_k represents process and measurement noise covariance and the matrices $F_x(x; k-1)$ and $G_x(x; k)$ are the Jacobians of $f(x_k, k)$ and $g(x_k, k)$ with the elements given by:

$$[F_x(x, k-1)]_{i,i'} = \left. \frac{\partial f_i(x, k-1)}{\partial x(i')} \right|_{x=x_k^-} \quad (3.8)$$

$$[G_x(x, k-1)]_{i,i'} = \left. \frac{\partial g_i(x, k)}{\partial x(i')} \right|_{x=x_k^-} \quad (3.9)$$

Big drawback of EKF is that in many cases the calculation of Jacobian and Hessian matrices can be a very difficult process, which is prone to errors.

3.2 Moving horizon estimation

Moving Horizon Estimation (MHE) is an optimization approach that uses observed measurements over prediction horizon, containing noise and disturbances, and produces estimates of unknown desired variables. Unlike Kalman filter which represents deterministic approach, MHE requires an iterative approach that relies on linear or nonlinear programming solvers to find a solution. There is a lot of solutions how to create filters working on moving horizon. Here was chosen L_2E filter (also known as LEF filter) which mostly give very satisfying results in terms of computation speed and precision. Whole procedure how to obtain LEF filter can be find in [4].

3.2.1 Finite impulse response filter

It is presumed, that we have system described by the equation (3.1), where output noise and input noise is considered as a same $v_k = w_k$. General FIR filter predictions on horizon $[k - N_f, k]$ are given as:

$$\hat{x}_k = \sum_{i=k-N}^{k-1} H_{k-i} y_i + \sum_{i=k-N}^{k-1} L_{k-i} u_i \quad (3.10)$$

where H_{k-i} and L_{k-i} are gain matrices with respect to y_i and u_i . Equation (3.10) can be represented in simpler form:

$$\hat{x}_k = HY_{k-1} + LU_{k-1} \quad (3.11)$$

Gain matrices have following form:

$$H = \begin{bmatrix} H_{N_f} & H_{N_f-1} & \cdots & H_1 \end{bmatrix} \\ L = \begin{bmatrix} L_{N_f} & L_{N_f-1} & \cdots & L_1 \end{bmatrix} \quad (3.12)$$

Output values Y_{k-1} and input values U_{k-1} on defined prediction horizon are given as:

$$Y_{k-1} = \begin{bmatrix} y_{k-N_f}^T & y_{k-N+1}^T & \cdots & y_{k-1}^T \end{bmatrix}^T \quad (3.13)$$

$$U_{k-1} = \begin{bmatrix} u_{k-N}^T & u_{k-N+1}^T & \cdots & u_{k-1}^T \end{bmatrix}^T \quad (3.14)$$

Output values Y_{k-1} are then obtained from:

$$Y_{k-1} = \overline{C}_N x_k + \overline{\Gamma}_{uN} U_{k-1} + \overline{\Gamma}_{wN} W_{k-1} + \overline{D}_{vN} V_{k-1} \quad (3.15)$$

where:

$$W_{k-1} = \begin{bmatrix} w_{k-N}^T & w_{k-N+1}^T & \cdots & w_{k-1}^T \end{bmatrix}^T \quad (3.16)$$

$$V_{k-1} = \begin{bmatrix} v_{k-N}^T & v_{k-N+1}^T & \cdots & v_{k-1}^T \end{bmatrix}^T \quad (3.17)$$

matrices $\overline{C}_N, \overline{\Gamma}_{uN}, \overline{\Gamma}_{wN}, \overline{D}_{vN}$ are given as:

$$\overline{C}_N = \begin{bmatrix} C\Phi^{-1} \\ C\Phi^{-2} \\ \vdots \\ C\Phi^{-N} \end{bmatrix}^T \quad (3.18)$$

$$\overline{\Gamma}_{uN} = \begin{bmatrix} C\Phi^{-1}\Gamma_u & 0 & \cdots & 0 \\ C\Phi^{-2}\Gamma_u & C\Phi^{-1}\Gamma_u & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\Phi^{-N}\Gamma_u & C\Phi^{-N+1}\Gamma_u & \cdots & C\Phi^{-1}\Gamma_u \end{bmatrix} \quad (3.19)$$

$$\overline{\Gamma}_{wN} = \begin{bmatrix} C\Phi^{-1}\Gamma_w & 0 & \cdots & 0 \\ C\Phi^{-2}\Gamma_w & C\Phi^{-1}\Gamma_w & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\Phi^{-N}\Gamma_w & C\Phi^{-N+1}\Gamma_w & \cdots & C\Phi^{-1}\Gamma_w \end{bmatrix} \quad (3.20)$$

$$\overline{D}_{vN} = \text{diag} \left[D_{v1} \quad D_{v2} \quad \cdots \quad D_{vN} \right] \quad (3.21)$$

If we now substitute equation (3.15) into (3.11) we get state estimation in a form:

$$\hat{x}_k = H \left(\overline{C}_N x_k + \overline{\Gamma}_{uN} U_{k-1} + \overline{\Gamma}_{wN} W_{k-1} + \overline{D}_{vN} V_{k-1} \right) + L U_{k-1} \quad (3.22)$$

We require a constraint that predicted state variables must be unbiased from the real state variables. Condition could be defined as:

$$E \left[\hat{x}_k \right] = E[x_k] \quad (3.23)$$

This condition is satisfied if there exist no noises on the prediction horizon. Then equation (3.12) have the form:

$$E \left[x_k \right] = H \left(\overline{C}_N E \left[x_k \right] + \overline{\Gamma}_{uN} U_{k-1} \right) + L U_{k-1} \quad (3.24)$$

where following constraints must be met:

$$\begin{aligned} H \overline{C}_N &= I \\ H \overline{\Gamma}_{uN} &= -L \end{aligned} \quad (3.25)$$

Equation for state variables prediction with consideration of mentioned conditions is then given as:

$$\hat{x}_k = H \left(Y_{k-1} - \overline{\Gamma}_{uN} U_{k-1} \right) \quad (3.26)$$

State estimation error have the form:

$$e_k = x_k - \hat{x}_k = H \left(\overline{\Gamma}_{wN} W_{k-1} + \overline{D}_{vN} V_{k-1} \right) \quad (3.27)$$

Matrices H and L are obtained from a solution to the optimal performance criterion:

$$\min_{H,L} \max_{w_i} \frac{\left[x_k - \hat{x}_k \right]^T \left[x_k - \hat{x}_k \right]}{\sum_{i=1}^{N_f} w_{k-1}^T w_{k-1}} \quad (3.28)$$

By using (3.26) and (3.27) can be (3.28) written as:

$$\min_{H,L} \max_{W_{k-1}} \frac{\left[W_{k-1}^T \left(\overline{\Gamma}_{wN} + \overline{D}_{vN} \right)^t H^T H \left(\overline{\Gamma}_{wN} + \overline{D}_{vN} \right) W_{k-1} \right]}{W_{k-1}^T W_{k-1}} \quad (3.29)$$

and solution to this criterion with the unbiased condition for LEF filter is then provided by:

$$\hat{x}_k = H \left(\overline{C}_N^T \Xi_N^{-1} \overline{C}_N \right)^{-1} \overline{C}_N^T \Xi_N^{-1} \left(Y_{k-1} - \overline{\Gamma}_{uN} U_{k-1} \right) \quad (3.30)$$

where Ξ_N :

$$\Xi_N = \overline{\Gamma}_{wN} \overline{\Gamma}_{wN}^T + \overline{D}_{vN} \overline{D}_{vN}^T \quad (3.31)$$

4 MODEL PREDICTIVE CONTROL AND ITS ALGORITHMS

4.1 Preview

The origins date back to the second half of the twentieth century when modern control was on the rise. In the beginning an LQG (*Linear Quadratic Gaussian*) algorithm was created. This algorithm uses so-called observer status also known as filter, whose foundations were created by Kalman. This kind of control wasn't too much suitable for industrial purposes, which led to the development of a better methodology an MPC (*Model Predictive Control*) methodology.

MPC is also known as MHC (*Moving Horizon Control*) or RHC (*Receding Horizon Control*) and it found its use primarily in industry for example in chemical process control, in the petrochemical, pulp and paper industries, gas pipeline control and active vibration control of railway vehicles [5]. Many sources describes this control as playing chess, where the player has to think ahead a few possible moves (solve optimization problem). In the best-case scenario the player thinks ahead all his possible moves and all his opponent moves so he always knows what to do to win. In comparison with classical PID control, which uses output feedback control law, model predictive control uses a discrete-time model of the system to obtain prediction of its future behaviour. Based on this prediction an optimal control problem can be solved. This optimization is done by cost function, which mostly minimizing difference between the predicted output and the desired reference. MPC also allows to use different types of constraints to the input, output and states, which is one of the biggest advantages of this control. But on the other hand more time and more memory is needed to compute solution of an optimization problem. Because of these computational demands, model predictive control was not used for fast processes, but nowadays with current computational systems the doors are wide opened [5, 6].

4.2 Model predictive control basics

Main strategy of the MPC is presented in Fig. 4.1 and can be described as follows:

STEP 1: Model is an explicit part of controller and he is used for prediction of future outputs $y(t)$. Predictions are calculated at each time step in the view of available information and the unknown future control signals $u(t)$ [5, 7].

STEP 2: The sequence of future control signals is computed as a result of optimized performance criterion, which often means minimize the error between a reference trajectory and the predicted process output. Performance criterion consists of cost function and constraints. Cost functions includes future output predictions, reference trajectory and future control signals. Constraints can be applied either on the output and on the input of the process [5, 7].

STEP 3: Only the current control signal $u(t)$ is transmitted to the process. At the next sampling new output value $y(t+1)$ is measured and whole sequences is repeated. This strategy is known as the receding horizon concept [5, 7].

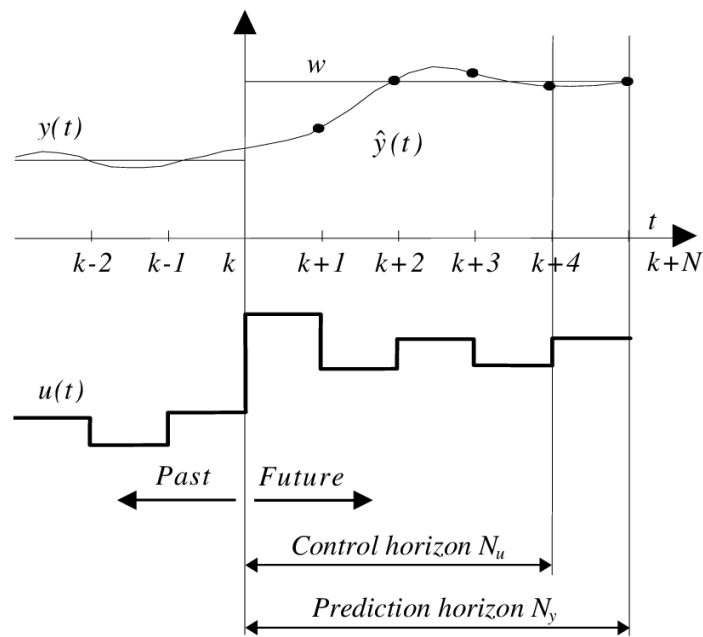


Fig. 4.1 Model predictive control strategy

Basic structure of an MPC is shown in Fig. 4.2 and it consists of a model and an optimizer, which includes cost function and constraints. These named parts will be discussed further. Note: For a state-space models are past outputs replaced by current states or, if they are unavailable, by current state estimates.

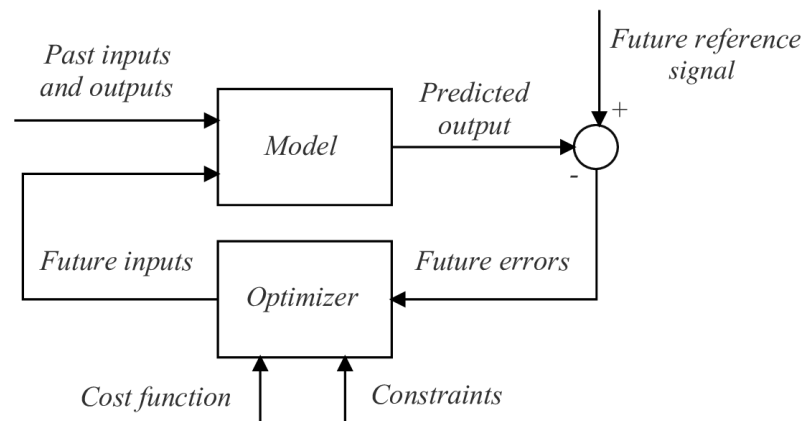


Fig. 4.2 Basic structure of model predictive control

4.2.1 Model

For MPC can be used any type of model. Primary categorization of applicable models is into linear and nonlinear. Cost function must be adjusted based on the chosen model. Computational requirements also depends on the choice of the model, because a simulation model must be capable of running faster than the real one. Used model

must accurately reflect the real system, because it is used for prediction of its output values. If disturbances are considered, they can be implemented into the model.

One of the most often used models is model based on impulse response, which can be obtained through measurement of the output, when the process is excited with an impulse input. This model can't be used for unstable processes and large number of parameters is needed, but it is used for his simplicity and easy identification without knowing anything about system structure. Other very similar type is the step response model, obtained when the input is a step. This model shares the same advantages and disadvantages with previous model. Very popular are models based on transfer function and state-space models. The models based on transfer function are presented as relation between the input and output of a linear time-invariant system with zero initial conditions. Transfer function models are used in GPC algorithm and they are widely spread in industry and academic research community. They can be used even for unstable systems, but the degree of the numerator and denominator must be known. State-space models use state variables to describe a system by a set of first-order differential or difference equations. State variables $x(t)$ can be reconstructed from the measured input-output data, but mostly they are not themselves measured during an experiment (they can be unmeasurable). Big advantage is that state-space models have the same representation for one-dimensional and multidimensional system, but disadvantage is requirement of the state observer. State-space models can be presented as nonlinear and as such, they can be reduced to polynomial models of specific order, then we speak about so called Volterra series models. Discrete type of linear and nonlinear state-space equation could have been seen in (2.1) and (2.11). Other representations of models are the models described by differential equations or the models based on neural networks and fuzzy logic [8].

4.2.2 Optimizer

Cost functions

The cost function (objective function) is necessary part of MPC. Cost function of linear models is mostly quadratic. Basic task is to minimize this function which gives sequence of input control signals. For better results penalization constant can be added. The cost function is often expressed in matrices and is computed by simulation in the prediction horizon for all sequences of the manipulated variable and the manipulated variable sequence is calculated by a numerical algorithm such as gradient method or even by some evolutionary algorithm. Duration of the calculation is extended after addition of constraints [8, 9].

Typical cost function look as follows:

$$J(N_d, N_y, N_u) = \sum_{j=N_d}^{N_y} \|\hat{y}(k+j|k) - w(k+j)\|_2^2 + \sum_{j=N_d}^{N_u-1} \lambda(j) \|\Delta u(k+j-1)\|_2^2 \quad (4.1)$$

where:

$\hat{y}(k+j|k)$ - sequence of future output values

$w(k+j)$ - sequence of desired values

$\Delta u(k+j-1)$ - sequence of future control efforts

Cost function contain some optional parameters. Parameters N_d and N_y represents minimal, maximal prediction horizon and they determine the future interval, when a reference signal trajectory should have been followed. Most often it is assumed, that reference trajectory is known. That helps to make a timely reaction, before any changes has been effectively made. Parameter N_u represents control horizon, which does not necessarily have to coincide with the maximum horizon. Lower value of N_u brings less computations. The coefficients $\delta(j)$ and $\lambda(j)$ are sequences that consider the future behaviour of controlled process. They are usually expressed in the form of constant values or exponential sequences. All these parameters are assumed as tuning parameters [8].

Constraints

In practice, we often encounter with constraints. It can be constraints of sensor, actuator or some technological limitations. Usually input variables are constrained because they operate only in a certain range of values. In addition, there are also some constraints for the process output variables (with respect to the environment or the safety of workplaces). Ability to work with constraints is one of the main advantages of predictive control, which had an impact on MPC expansion in the industry, where large number of industrial processes is controlled to values close to restrictive conditions [5].

The simplest and most widely used method in practice is analytical solution to optimization problem without constraints and subsequent application of constraints (saturation of the output). This solution is simple, but does not guarantee optimal control by selected criteria. A more appropriate solution is to solve the optimization problem already with the given constraints. This approach will not only reduce the output variable algorithm, but also the actual output of the system, and in the event of a state-space model even individual internal states of the system [7].

Constraints can be distinguished to:

Hard constraints

For hard constraints applies, that they can never be exceeded (they must be satisfied). Typical examples are:

- constraint of control input variable: $u_{min} \leq u(k) \leq u_{max}$
- constraint of output variable: $y_{min} \leq y(k) \leq y_{max}$

Soft constraints

For constraint variable it is allowed to exceed the established limits on certain limit tolerance ε . Typical examples are:

- constraint of control input variable: $u_{min} + \varepsilon_{min} \leq u(k) \leq u_{max} + \varepsilon_{max}$
- constraint of output variable: $y_{min} + \varepsilon_{min} \leq y(k) \leq y_{max} + \varepsilon_{max}$

4.3 Review of model predictive control algorithms

Reviews of MPC algorithms can be found in a lot of publications [8, 10, 11, 12]. Algorithms differs mainly in the way the system is modeled, also there are differences in formulations of the cost function and of the optimization task and utilization of constraints. Algorithms can be divided into two main groups based on used model, these groups are linear and nonlinear MPC algorithms. First created algorithms were based on linear computations, which is obvious, because they were sufficient for many applications and methodology of their solving was quite quick and "easy" and former computational hardware was powerful enough to solve them.

4.3.1 Linear model predictive control algorithms

First generation of MPC algorithms were algorithms MPHC (*Model Predictive Heuristic Control*) and DMC (*Dynamic Matrix Control*). MPHC is commonly known as MAC (*Model Algorithmic Control*) or under its software solution name IDCOM (*IDentification COMmand*). MAC uses a linear process model in the form of a finite impulse response (FIR) and the cost function in the form of a sum of squared deviations of the predicted trajectory from a reference trajectory. Time constant of the reference trajectory changes speed of the controller. More increased value leads to a slower but more robust controller. The DMC algorithm is quite similar with some differences. It uses a linear process model in the form of a step response instead of impulse response, it calculates prediction only for first N terms and it uses control input difference $\Delta u(t)$ instead of $u(t)$. Some kind of superstructure of the DMC algorithm is the QDMC algorithm (*Quadratic Dynamic Matrix Control*). By this superstructure is meant the way of handling with constraints. In QDMC constraints are treated as linear inequalities, so the final solution is obtained through solving of a quadratic programming problem. This algorithm belongs to the second-generation of MPC algorithms. Although this algorithm solved some disadvantages of his predecessors some drawbacks still remained. So the new generation of algorithms came up. These algorithms improved current solutions and provides some mechanism to recover from an infeasible solution, provides a richer set of options for feedback and they are suitable for wider range of dynamic processes (stable, unstable and integrating). First mentioned algorithm is named after his predecessor IDCOM-S/M (S refers to SISO version and M refers to MISO version). An distinction of the IDCOM-S/M algorithm is that it uses two separate objective functions, one for the outputs and one for the inputs if there are extra degrees of freedom. To define the reference trajectory IDCOM-S/M uses two basic tuning parameters a coincidence point and a closed-loop response time. IDCOM-S/M algorithm was later modified into an SMCA (*Setpoint Multivariable Control Architectur*) algorithm. An improved numerical solution engine of SMCA algorithm allow to solve a sequence of separate steady-state target optimizations . SMOC (*Shell Multivariable Optimizing Controller*) algorithm algorithm is nearly equivalent to solving the LQR problem with input and output constraints, except the fact that it is still formulated on a finite horizon. SMOC using the state-space model as well as algorithm PFC (*Predictive Functional Control*). Also there were created algorithms, which uses a model based on transfer function. Probably the best known algorithm of this group is GPC (*Generalized Predictive Control*) algorithm, other algorithms are EPSAC (*Extended Prediction Self Adaptive Control*) and EHAC (*Extended Horizon Adaptive Control*). The differences between these algorithms is in the definition of the used model structure. Some companies made their own versions of MPC algorithms, from all lets name RMPC (*Robust Multivariable Predictive Control*) algorithm developed by Honeywell [8, 10].

4.3.2 Nonlinear model predictive control algorithms

First idea how to obtain nonlinear MPC is based on leaving the algorithm design unchanged and use nonlinear instead of linear process model for predicting the outputs. Algorithm, which uses this approach was named as MPC-NO (*MPC with Nonlinear Optimization*). Seemingly small difference when a nonlinear process model is used instead of a linear one, causes that the optimization problem becomes non-quadratic and non-convex. To these days there are no universal optimization procedures for solving such problems which would ensure finding a solution reliably and quickly. Gradient optimization procedures usually find only local minima. For this drawback MPC algorithms with a nonlinear model used in the controller optimization problem, which is solved at every sampling instant, can be applied mainly for slow and strongly nonlinear processes. Another suitable approach is an attempt to use a branch-and-bound method. To apply this method it's necessary to discretize each of the decision control input variables. Through this method a suboptimal algorithm is obtained, with the level of suboptimality and the computational burden depending on the accuracy of this discretization. The algorithms with suboptimal structure, at each sampling instant, perform a linearization of the nonlinear model, at a current process state. The linearization may not be necessary at each sampling instant, if the nonlinearity of the system is weak or if it's working close to the equilibrium point for longer time periods. This fact reduces computational demands, however, this makes the algorithm inappropriate for situations, when a quick transfer to a distant set-point is needed or for dynamic transients after strong and rapid changes of disturbances. This algorithmic solution was named as MPC-NSL (*MPC with Nonlinear with Successive Linearization*) algorithm. A more precise algorithmic solution than the MPC-NSL algorithm offers an MPC-NPL (*MPC with Nonlinear Prediction and Linearization*) algorithm. MPC-NPL uses a linearized model only in the optimization problem, i.e., with only a forced trajectory of controlled outputs linearly depending on decision variables, but evaluating the free trajectory of predicted outputs from a nonlinear model at each sampling instant. This approach does not significantly affect the total amount of performed calculations but usage of the MPC-NPL algorithm may not be satisfactory, if fast and large changes in operating points are required. The way of improving the MPC-NPL algorithm can be proposed as providing iterative corrections of the nonlinear free output response and of the resulting quadratic programming problem. This algorithm is then called an MPC-NPL+ algorithm. One of the promising research directions in the area of nonlinear MPC is the application of neural models. MPC algorithms which are using ANNs (Artificial Neural Networks) to modelling and control are widely used these days [11].

4.4 Model predictive control algorithm for state-space model

In this part will be presented GPC algorithm in modification for the state-space model. Creation of the MPC controller can be separated into two actions: prediction based on created model and minimisation of the cost function (obtaining the control law).

4.4.1 Prediction with state-space models

Prediction with a state-space model is straightforward and it could be found in [12]. Consider the state-space model which gives the one-step ahead predictions in the form:

$$\begin{aligned}x_{k+1} &= \Phi x_k + \Gamma u_k \\y_{k+1} &= C x_{k+1} + L d_{k+1}\end{aligned}\tag{4.2}$$

where parameter d_k represents compensation parameter, which gives value of prediction error and can be computed as:

$$d_k = y_k - Cx_k \quad (4.3)$$

Relationship given in (4.2) can be used recursively to find predictions:

$$\begin{aligned} x_{k+2} &= \Phi x_{k+1} + \Gamma u_{k+1} \\ y_{k+2} &= Cx_{k+2} + d_{k+2} \end{aligned} \quad (4.4)$$

If (4.2) is substituted into (4.4) then x_{k+1} is eliminated:

$$\begin{aligned} x_{k+2} &= \Phi^2 x_k + \Phi \Gamma u_k + \Gamma u_{k+1} \\ y_{k+2} &= Cx_{k+2} + d_{k+2} \end{aligned} \quad (4.5)$$

In another step is get:

$$\begin{aligned} x_{k+3} &= \Phi^2 x_{k+1} + \Phi \Gamma u_{k+1} + \Gamma u_{k+2} \\ y_{k+3} &= Cx_{k+3} + d_{k+3} \end{aligned} \quad (4.6)$$

Again substitute from (4.2):

$$\begin{aligned} x_{k+3} &= \Phi^3 x_k + \Phi^2 \Gamma u_k + \Phi \Gamma u_{k+1} + \Gamma u_{k+2} \\ y_{k+3} &= Cx_{k+3} + d_{k+3} \end{aligned} \quad (4.7)$$

For the n-step ahead predictions can be this recursion expressed more generally as:

$$\begin{aligned} x_{k+n} &= \Phi^n x_k + \Phi^{n-1} \Gamma u_k + \Phi^{n-2} \Gamma u_{k+1} + \dots + \Gamma u_{k+n-1} \\ y_{k+n} &= C(\Phi^n x_k + \Phi^{n-1} \Gamma u_k + \Phi^{n-2} \Gamma u_{k+1} + \dots + \Gamma u_{k+n-1}) + d_{k+n} \end{aligned} \quad (4.8)$$

Hence can be formed the whole vector of future predictions up to a horizon N_y as follows:

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N_y} \end{bmatrix} = \begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^{N_y} \end{bmatrix} x_k + \begin{bmatrix} \Gamma & 0 & \dots & 0 \\ \Phi \Gamma & \Gamma_w & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Phi^{N_y-1} \Gamma & \Phi^{N_y-2} \Gamma & \dots & \Gamma \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_y} \end{bmatrix} \quad (4.9)$$

In a simple form:

$$X_k = P_x x_k + H_x U_k; \quad (4.10)$$

then for y_k we can write:

$$Y_k = C(P_x x_k + H_x U_k) + L d_k; \quad (4.11)$$

Matrices can be multiplied by matrix C which gives:

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+N_y} \end{bmatrix} = \begin{bmatrix} C\Phi \\ C\Phi^2 \\ \vdots \\ C\Phi^{N_y} \end{bmatrix} x_k + \begin{bmatrix} C\Gamma & 0 & \cdots & 0 \\ C\Phi\Gamma & C\Gamma_w & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N_y-1}\Gamma & C\Phi^{N_y-2}\Gamma & \cdots & C\Gamma \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_y-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} d_k \quad (4.12)$$

and in a simple form:

$$Y_k = P x_k + H U_k + L d_k \quad (4.13)$$

Because typical control law is given as a minimisation of the cost functions, where penalties of control input values are used, matrix U_k must be converted into ΔU_k as follows:

$$\Delta U_k = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_k \\ \vdots \\ u_{k+N_y-2} \end{bmatrix} \quad (4.14)$$

$$\Delta U_k = R U_k \quad (4.15)$$

Because first control input is previous input which is constant that can't be changed we can adjust previous equation into following form:

$$\Delta U_k = R_d u_{k-1} + R_r U_k \quad (4.16)$$

Reason why penalties are used is, because it gives integral action and eliminates optional deviation of the predicted output from the set-point.

4.4.2 Control law

The control law is determined from the minimisation of a cost function. A prime component of this function is the cost of deviations of the predicted outputs Y_k from the set-points W_k , i.e., the cost of predicted control errors $E_k = W_k - Y_k$. Second component is the cost of penalties of control input values. Most commonly used cost function are second norm, then considering the two mentioned components, the form of the predictive control law is given by minimisation of the equation (4.1).

$$\min_{\Delta U_k} J(N_d, N_y, N_u) = \|E_k\|_2^2 + \lambda \|\Delta U_k\|_2^2 \quad (4.17)$$

II. PRACTICAL PART

5 THE CE152 MAGNETIC LEVITATION SYSTEM SPECIFICATION

The CE152 Magnetic levitation system is a nonlinear unstable dynamic system with one input and one output. Input signal is the control signal and output signal gives information about position of a steel ball. Both signal values are converted and scaled to the specific range of the machine unit [MU].

5.1 The CE152 Magnetic levitation structure

System consist of a model of the magnetic levitation system, power supply and an universal data acquisition card MF624. MF624 is a standard PCI card with A/D, D/A converters, analog/digital inputs and outputs, counters, timers and appropriate drivers. The model is connected to the PC via this card. More detailed description can be find in [13].

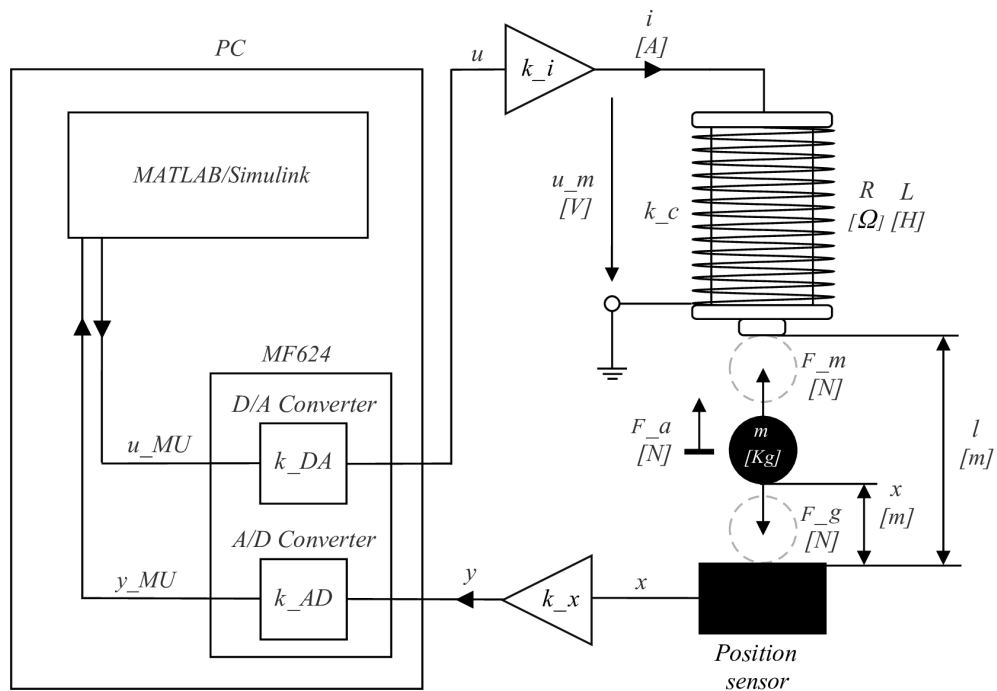


Fig. 5.1 Principal scheme of the magnetic levitation model

For a model of the system are considered these parts: D/A converter, A/D converter, the position sensor, the power amplifier, the ball and coil subsystem

Simplified inner structure is shown in Fig. 5.1 . A steel ball levitates in magnetic field of the coil driven by power amplifier connected to D/A converter. Position of the steel ball is measured by inductive linear position sensor connected to A/D converter. Both control and measured parameters are sent and received by Simulink.

6 SYSTEM BEHAVIOUR

In this part system behaviour is discussed. Like was said before the CE152 Magnetic levitation system is a nonlinear unstable dynamic system with one input and one output. When an input control signal of certain value is sent to the system, the ball is lifted upwards to the magnetic core and he stops when he hit the core. This behaviour is caused because electromagnetic force of the magnetic core overcame the force of gravity. As the ball getting closer, accelerating force grows. Because of an obstacle in the form of magnetic core, both the ball and accelerating force stops. Higher input signal means higher electromagnetic force and much more rapidly increasing acceleration. If input control value is decreased under the certain value, the ball falls down. That is happening, because the electromagnetic force is to low to overcome gravitational force. Ball fell down to the head of the sensor, which stops the ball. When the ball hits the sensor he bounces a few times. Presented behaviour is the system basic behaviour on a constant input signal. Following figure shows the step response on different input values and the reaction, when the input value has changed to zero. If input value is not exactly zero, but still to low to hold up the ball, then the ball still falls, but his bouncing behaviour is slightly different thanks to the non zero force of attraction. Tested input signal was step signal because other input signals like ramp or sinus signal don't have adequate information value, thanks to the described system behaviour, when once the electromagnetic force is strong enough, the ball is attracted to magnetic core, despite increasing input signal. In opposite case, when input signal decreasing, it is the same situation but in opposite meaning of the ball movement.

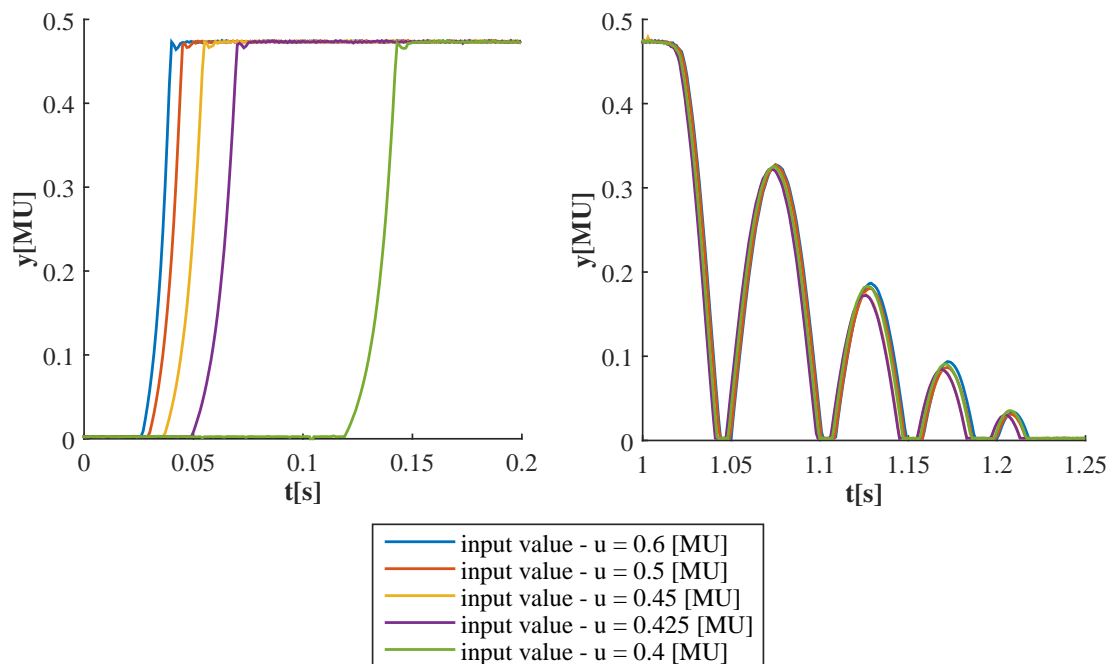


Fig. 6.1 Step response on different input value and reaction on input value change to zero

As it can be seen, there is some kind of delay, before the system starts to attract the ball. This is happening because coil first need to attract the ball straight to the middle under its core and then the ball is lifted. This can be overcome by setting an optimal starting point of the ball. Even though, the small but non zero delay still remains and it is probably caused by internal processes in the system. Because of later comparison of the created model with the real processes, counting with this delay should be in mind.

7 IDENTIFICATION OF THE SYSTEM

Based on mentioned approaches and methodologies, an adequate model of CE152 magnetic levitation system is deduced in this part. For creation of the model MATLAB/SIMULINK environment was used. On the next several pages whole procedure of modelling and identification is described. According to above preview at the system specification and composition, model can be decomposed to following parts: D/A and A/D converter, position sensor, power amplifier and ball and coil subsystem. All parts can be identified separately. All parts are eventually interconnected and they form the final model of CE152 magnetic levitation system.

7.1 D/A converter

The D/A converter converts digital signal u_{MU} from PC into an analog voltage signal u . The D/A converter can be described by a linear function (7.1) and represented by a Simulink block (Fig. 7.1).

$$u = k_{DA}u_{MU} \quad (7.1)$$

where:

u - D/A converter output signal/coil input voltage [V]

u_{MU} - D/A converter input signal [MU]

k_{DA} - D/A converter gain [V/MU]

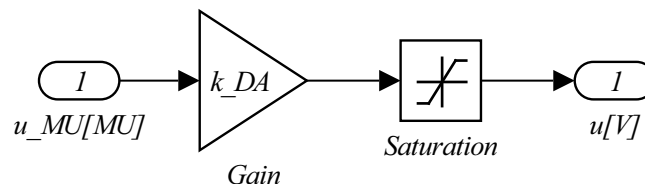


Fig. 7.1 Simulation diagram of the D/A converter

D/A converter maps the input signal range $u_{MU} = -1[MU]$ to $u_{MU} = 1[MU]$ to the range $u = 0[V]$ to $u = 5[V]$. This leads to converter gain $k_{DA} = 10[MU]$. Real range of D/A input signal is -10 to $10[V]$ but input to the CE152 magnetic levitation system is constructed for the range 0 to $5[V]$, so signal input must be constrained.

7.2 A/D converter

The A/D converter converts analog voltage signal y into a digital signal y_{MU} . The A/D converter can be described by a linear function (7.2) and represented by a Simulink block (Fig. 7.2).

$$y_{MU} = k_{AD}y \quad (7.2)$$

where:

y - A/D converter output signal/position sensor voltage [V]

y_{MU} - A/D converter output signal [MU]

k_{AD} - A/D converter gain [MU/V]

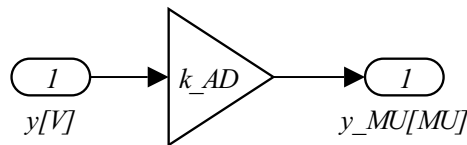


Fig. 7.2 Simulation diagram of the A/D converter

A/D converter maps the input signal range $y = -10[V]$ to $y = 10[V]$ to the range $y_{MU} = -1[MU]$ to $y_{MU} = 1[MU]$. This leads to converter gain $k_{AD} = 0.1[MU]$.

7.3 The position sensor

An inductive position sensor is used to measure the ball position x . Maximum declared height l is calculated as a difference between physical distance of the coil and the sensor l_0 and the ball diameter d_k . The position of the ball is obtained by reading the voltage from A/D converter's output. Sensor voltage varies with ball position. The relation between ball position and voltage is approximately linear.

$$y = k_x x + y_0 \quad (7.3)$$

where:

x - ball position [m]

y_0 - position sensor offset [V]

y - position sensor voltage [V]

k_x - position sensor gain [V/m]

Calibration experiment must be done. First of all a travelling distance of the ball l has to be calculated. It is a difference between physical distance of the coil and the sensor $l_0 = 18.4 \cdot 10^{-3}[m]$ and the ball diameter $d_k = 12.7 \cdot 10^{-3}[m]$.

$$l = l_0 - d_k = 5.7 \cdot 10^{-3}[m]$$

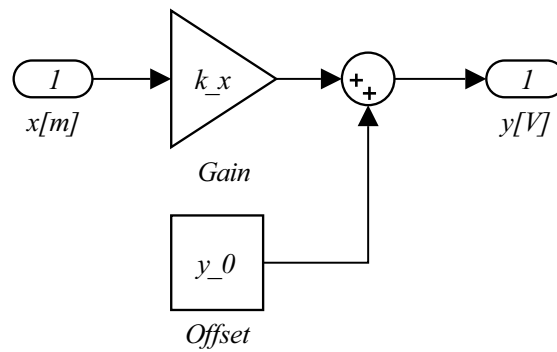


Fig. 7.3 Simulation diagram of the position sensor

Tab. 7.1 Calibration data of the position sensor

i	$x_i[m]$	$y_{MU_i}[MU]$	$y_i[V]$
1	0	0.00254	0.02537
2	0.0057	0.47384	4.73840

Final results of the boundary values were noted, and based on calculation with these values, position sensor gain is obtained. Position sensor offset y_0 is taken as an initial value from the inductive sensor, when input action signal is zero.

$$y_0 = y_1 = 0.02537[V]$$

$$k_x = \frac{y_2 - y_1}{x_2 - x_1} = 826.8525[V/m]$$

7.4 The power amplifier

The power amplifier works as transconductance amplifier whose differential voltage between input voltage u from the D/A converter and u_i produces an output current supplied to the coil. The power amplifier essentially represents a source of constant current with the current stabilisation. Internal structure of the amplifier is in Fig. 7.4.

where:

k_{am} - amplifier gain [-]

k_s - current sensor gain [-]

L - coil inductance [H]

R_c - coil resistance [Ω]

R_s - resistance of feedback resistor [Ω]

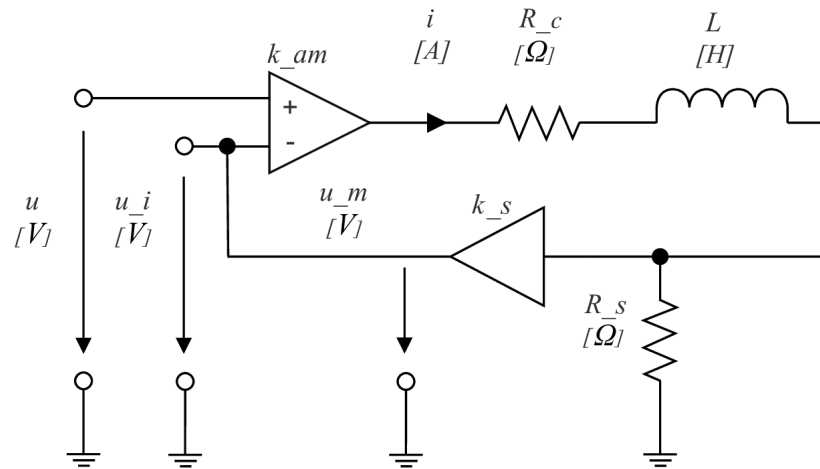


Fig. 7.4 Internal structure of the power amplifier

$$u_m = \frac{di}{dt}L + i(R_c + R_s) \quad (7.4)$$

$$u_m = k_{am}(u - R_s k_s i) \quad (7.5)$$

$$\frac{di}{dt}L + i(R_c + R_s) = k_{am}(u - R_s k_s i) \quad (7.6)$$

$$\frac{di}{dt}L + i(R_c + R_s + k_{am}R_s k_s) = k_{am}u \quad (7.7)$$

Now if we do direct Laplace transform with zero initial conditions we get:

$$LsI(s) + (R_c + R_s + k_{am}R_s k_s)I(s) = k_{am}U(s) \quad (7.8)$$

$$\frac{I(s)}{U(s)} = \frac{k_{am}}{Ls + (R_c + R_s + k_{am}R_s k_s)} \quad (7.9)$$

$$\frac{I(s)}{U(s)} = \frac{\frac{k_{am}}{R_c + R_s + k_{am}R_s k_s}}{\frac{L}{R_c + R_s + k_{am}R_s k_s}s + 1} \quad (7.10)$$

We can also do simplification of this equation and define it by transfer function of 1st order as it is shown on the next page.

$$G_{PA}(s) = \frac{I(s)}{U(s)} = \frac{k_i}{T_a s + 1} \quad (7.11)$$

$$k_i = \frac{k_{am}}{R_c + R_s + k_{am} R_s k_s} \quad (7.12)$$

$$T_a = \frac{L}{R_c + R_s + k_{am} R_s k_s} \quad (7.13)$$

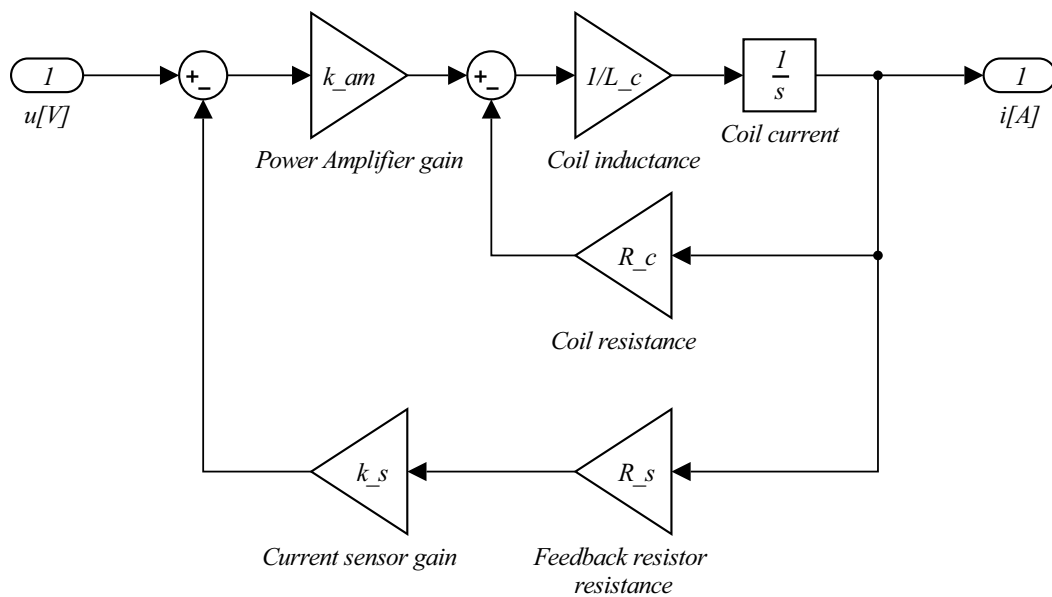


Fig. 7.5 The power amplifier detailed simulation diagram

Constant k_i is an amplifier gain and T_a is an amplifier time constant. Humusoft in their manual [13] states, that typical parameters of each component of the coil and power amplifier are:

$$k_{am} = 100[-]; k_s = 13.33[-]; L = 0.03[H]; R_c = 3.5[\Omega]; R_s = 0.25[\Omega]$$

When these values are substituted into equations (7.12) and (7.13) then power amplifier gain and time constant have following values.

$$T_a = 8.902 \cdot 10^{-5}[s]$$

$$k_i = 0.297[A/V]$$

Time constant is very small and can be neglected, then only power amplifier gain can remain. Difference between response of the detailed model and the simplified model with only power amplifier gain was assessed as minimal. Because of this minimal difference and faster computation of a simulation, simplified model was used. Simplified simulation diagram is then shown in Fig. 7.6.

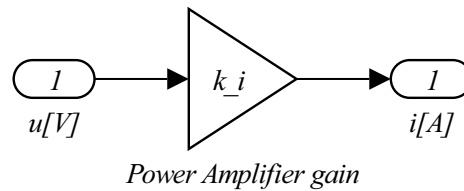


Fig. 7.6 The power amplifier simplified simulation diagram

7.5 The ball and coil subsystem

For modelling ball and coil subsystem we can use Lagrange's method. The motion equation is based on the balance of all acting forces. Final equation (7.16) is a nonlinear second order differential equation where input variable is electric current and output variable is ball position. Gravitational force F_g depends on ball mass and it is constant. This force acts against electromagnetic force F_m created by coil, when electric current pass through it. In view of the above it is clear that to lift the ball up, electromagnetic force must be greater than gravitational force (accelerating force grater than zero). From the equation (7.16) we can figure out that electromagnetic force relies on two parameters, the amount of electric current i and actual ball position x . In this model also damping force is considered.

$$F_a = F_m - Fg \quad (7.14)$$

$$F_a = m_k \ddot{x} + k_{fv} \dot{x} \quad (7.15)$$

$$F_m = \frac{i^2 k_c}{(x - x_0)^2} \quad (7.16)$$

$$F_g = m_k g \quad (7.17)$$

$$m_k \ddot{x} + k_{fv} \dot{x} = \frac{i^2 k_c}{(x - x_0)^2} - m_k g \quad (7.18)$$

where:

F_m - electromagnetic force [N]

F_a - accelerating force [N]

F_g - gravitational force [N]

g - gravitational acceleration [m.s⁻²]

l_0 - physical distance between the coil and the sensor[m]

x - ball position [m]

m_k - ball mass [kg]

x_0 - coil offset [m]

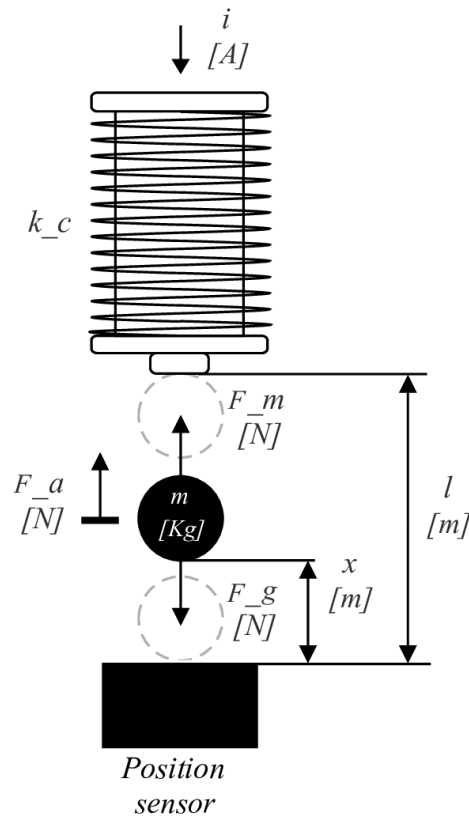


Fig. 7.7 The ball and coil simulation diagram

kc - coil constant [-]

i - coil current [A]

k_{fv} - dumping constant [N.s.m⁻¹]

Parameter of the ball mass m_k can be calculated from its parameters $d_k = 12.7 \cdot 10^{-3}$ [m] and $\rho = 7800$ [kg.m⁻³].

$$m_k = \rho V_k = \rho \frac{4}{3} \pi \left(\frac{d_k}{2} \right)^3 = 8.37 \cdot 10^{-3} [\text{kg}]$$

Parameters of the coil constant are obtained through the measurement of the steel ball stable positions, which means that the system have to be controlled somehow. Required data was get from closed loop control. Design of the control was taken over from Humusoft real time toolbox control simulation model for CE152 magnetic levitation system. Calculation of desired parameters has been done by two point calibration method with data given by (Tab. 7.2). If more than two stable positions can be calculated, also some interpolation technique can be used.

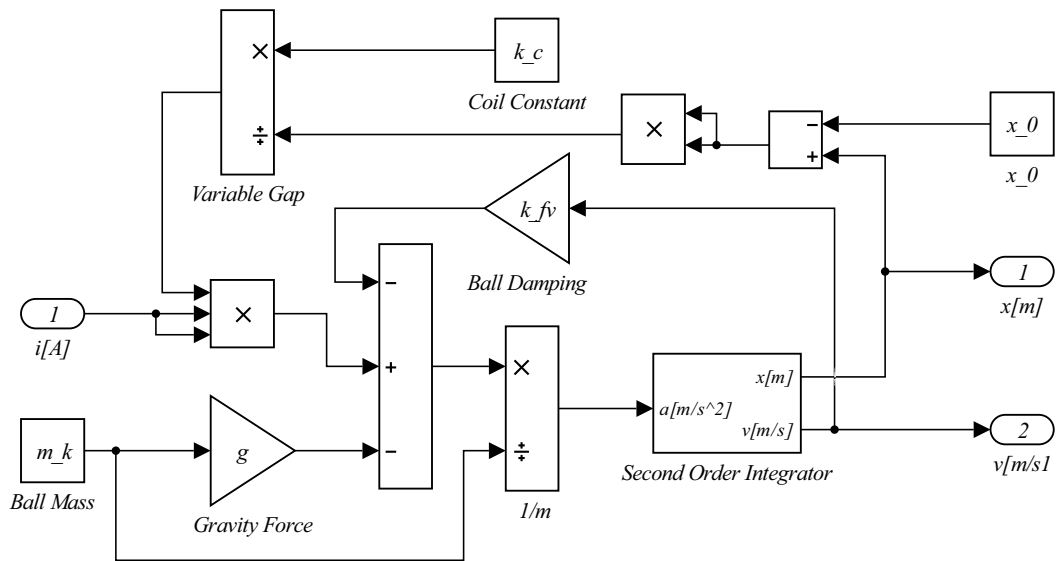


Fig. 7.8 The ball and coil simulation diagram

7.5.1 Two point calibration parameter estimation

Two stable points were measured and based on the following equation desired parameters were calculated.

$$x_0 = \frac{\frac{x_2^s}{u_2^s} - \frac{x_1^s}{u_1^s}}{\frac{1}{u_2^s} - \frac{1}{u_1^s}} \tag{7.19}$$

$$k_c = m_k g \frac{(x_2^s - x_0)^2}{u_2^{s2} k_i^2} \tag{7.20}$$

Tab. 7.2 Two point calibration data

i	$y_{MU_i}^s [MU]$	$x_i^s [m]$	$i_i^s [A]$	$u_i^s [V]$
1	0.1500	0.018	0.6579	2.2152
2	0.3500	0.0042	0.4144	1.3951

Final values are then:

$$x_0 = 0.0083[m]; \quad k_c = 8.0915 \cdot 10^{-6} [N.m^2.A^{-2}]$$

Parameter of the ball damping constant k_{fv} can not be measured directly or by a dedicated experiment and the only possibility is the trial-and-error method using real experimental data for comparison. The resulting value of k_{fv} was set to:

$$k_{fv} = 0.0195 [N.s.m^{-1}]$$

7.5.2 The second order integrator

In the structure of a Ball and Coil model can be seen a Second Order Integrator block. This block is available in newer versions of Matlab and Simulink, but in older versions he is not. Since this block is not available in university current Matlab version, he had to be created (Fig. 7.10). For creation two integration blocks was used. Second block expresses position of the ball which is saturated by minimal ball position and maximal ball position. This saturation does not affect the velocity in any certain way, and so the velocity still grow. Proper behaviour has to be handled manually. Block that handling this situation is Hard Stop and Bouncing block.

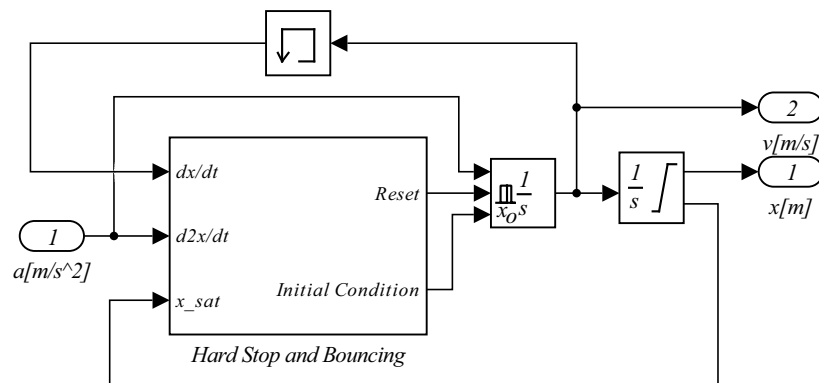


Fig. 7.9 Second order integrator diagram

Hard Stop and Bouncing block also includes other feature, which is bouncing ball physics. For bouncing ball behaviour a coefficient of restitution is used. This coefficient accounts for the loss of kinetic energy during each bounce. Bouncing behaviour of the model isn't too much close to the behaviour of the real system, because it doesn't take into account friction and other impacts of physics. Comparing can be found in section dedicated to validation and verification of the model.

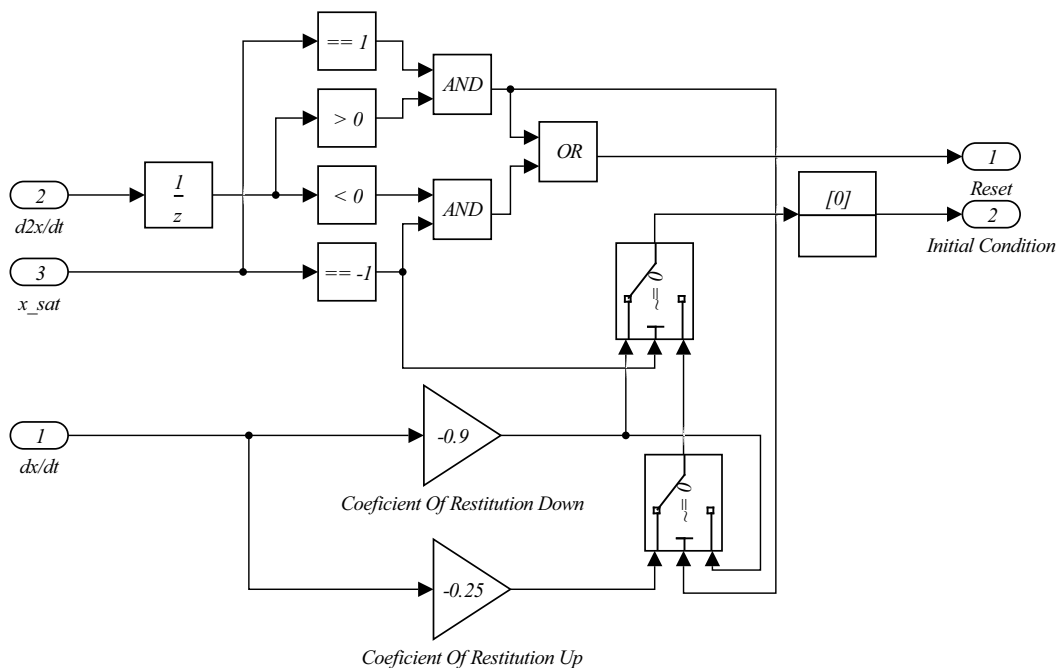


Fig. 7.10 Hard stop and bouncing block diagram

7.6 The whole system model

The whole system consists of created subsystem, which were included into subsystem blocks and interconnected. Interconnection of these blocks is shown on Fig. 7.11. In conclusion subsystem blocks were turn into a parent subsystem block (Fig. 7.12.).

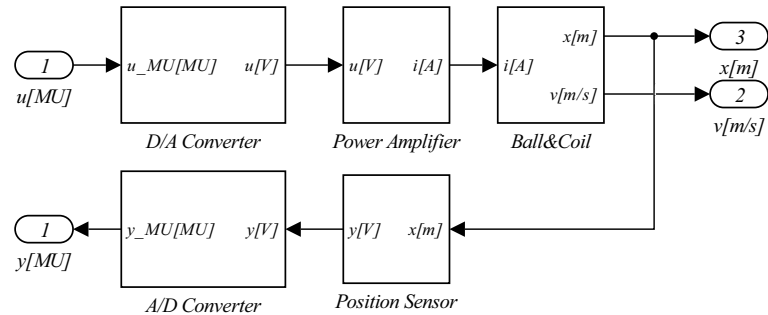


Fig. 7.11 Interconnection of subsystem blocks

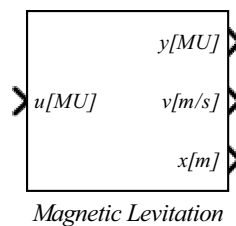


Fig. 7.12
Subsystem block
of the CE152
magnetic
levitation system

7.7 Verification and validation of the nonlinear model

From the perspective of a verification, created model is satisfactory. Behaviour of the model is just like the behaviour of the real system template. When we take a look on results of a validation, they are surprisingly good. The control input values have the same course and same is it with output values. The resulting characteristics depends strictly on estimated parameters. In Fig. 7.13 can be seen response of the created model of the system to the input value $u = 0.45[MU]$ and its confrontation with response of the real system. Response of the real system is then moved a little backwards, because of the delay discussed in a chapter dedicated to the system behaviour. From the same reason also response on the change of input value to zero needs to be moved a little backwards. This corresponds to an immediate response of the system. Now if responses are compared it can be said, that created model have the same behaviour. Waveforms have the same direction and tendency, only in a bouncing behaviour are big differences but this is obvious, because this behaviour was modelled in the simplest form.

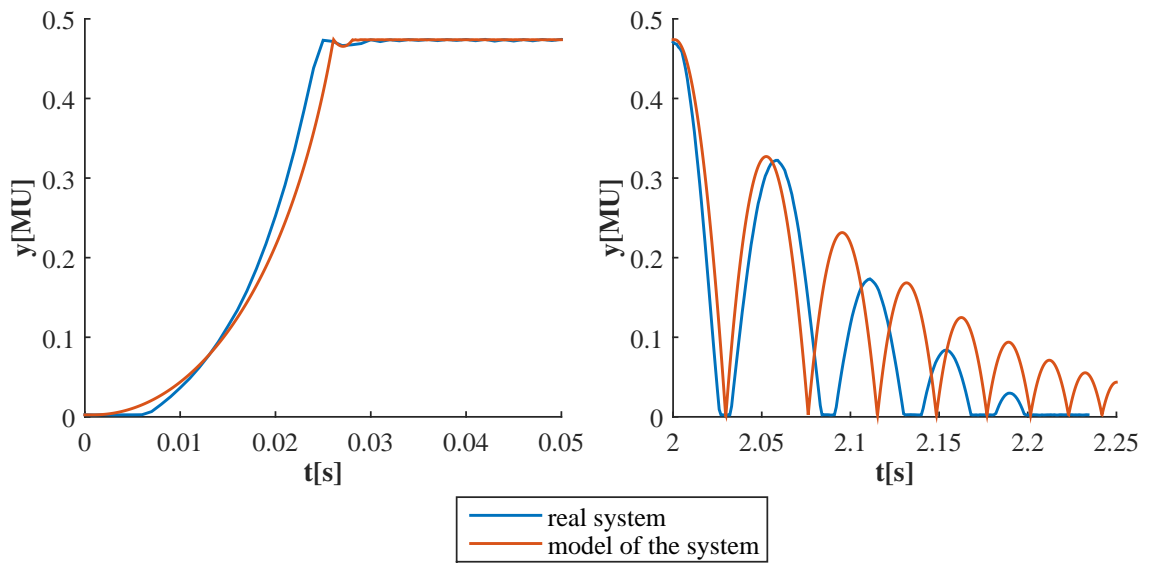


Fig. 7.13 Comparison of response of the model and the system

Validation is checked, by doing some control experiments on the system and then the same experiments are repeated on the model. Compared values are output values and input action values. From Fig. 7.14 can be said that even from the perspective of validation is created model sufficiently adequate, because both the output and even the control input signal have the same behaviour as the output and control input signal given by the real system.

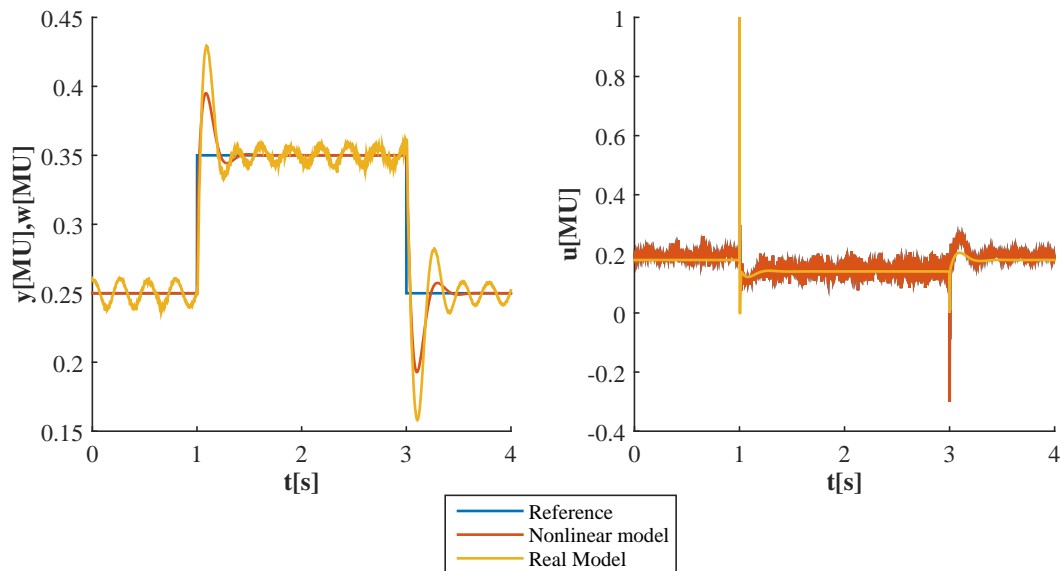


Fig. 7.14 Comparison of the output values of the real system and created model

8 LINEARISATION OF THE NONLINEAR MODEL

Linearised state-space model will be created in this part. We assume that model is a SISO model, whose state vector consist of coil current - i , ball velocity - v and ball position - x . Input voltage is represented by u and y is then A/D converter output. System can be described as follows.

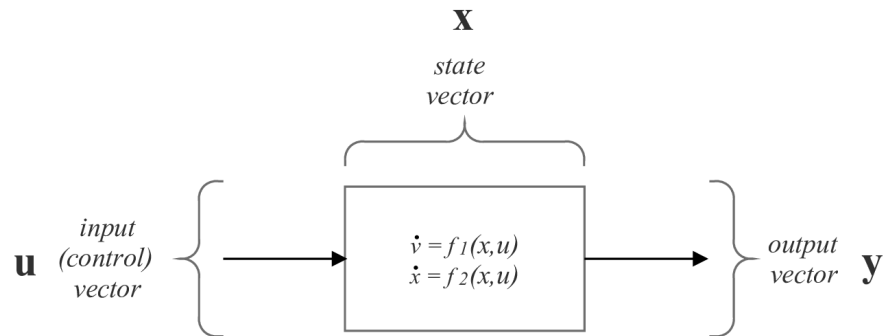


Fig. 8.1 State space system

Mathematical state-space representation of this system is as it was presented in equation (??). This is the general representation of a nonlinear system. In this part is required a linear time-invariant system, which can be described by equation (??). In this representation figure matrices **A**, **B**, **C** and **D**. Significance of matrices is:

- A** state (system) matrix with dimensions $n \times n$
- B** input matrix with dimensions $n \times p$
- C** output matrix with dimensions $q \times n$
- D** feedthrough (feedforward) matrix with dimensions $q \times p$

Based on previous assumption of the system parameters the dimensions values are $n = 2, p = 1, q = 1$. Then the resulting shape of the matrices is:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}; B = \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix}; C = \begin{bmatrix} c_{1,1} & c_{1,2} \end{bmatrix}; D = \begin{bmatrix} d_{1,1} \end{bmatrix}$$

Parameters of the above-mentioned matrices are obtained through linearisation of following differential equations at selected operating (nominal) point.

$$\frac{dv}{dt} = \frac{(k_{DA} * k_i * u_{MU})^2 k_c}{m_k (x - x_0)^2} - \frac{k_{fv}}{m_k} v - g \tag{8.1}$$

$$\frac{dx}{dt} = v \tag{8.2}$$

$$y_{MU} = k_{AD} k_x x + k_{AD} y_0 \tag{8.3}$$

Used linearisation method expands the nonlinear state equation into a Taylor series about an operating point usually the equilibrium point. This point is represented by $[v_{ss}, x_{ss}, u_{MU_{ss}}]$. All the terms of the Taylor series of order higher than the first are discarded. Before proper selection of the operating point will be done, the parameters of the matrices will be determined. Determination of matrices parameters is done based on partial derivatives of differential equations with respect to state variables and input variables at the selected operating point.

$$A = \left. \frac{\partial f_i}{\partial x_i}(x, u_{MU}) \right|_{v_{ss}, x_{ss}, u_{MU_{ss}}} \quad (8.4)$$

$$B = \left. \frac{\partial f_i}{\partial u_i}(x, u_{MU}) \right|_{v_{ss}, x_{ss}, u_{MU_{ss}}} \quad (8.5)$$

$$C = \left. \frac{\partial g_i}{\partial x_i}(x, u_{MU}) \right|_{v_{ss}, x_{ss}, u_{MU_{ss}}} \quad (8.6)$$

$$D = \left. \frac{\partial g_i}{\partial u_i}(x, u_{MU}) \right|_{v_{ss}, x_{ss}, u_{MU_{ss}}} \quad (8.7)$$

Steady state is assumed, then equations (8.1) and (8.2) are set to be zero, and it leads to

$$v_{ss} = 0 \quad (8.8)$$

$$u_{MU_{ss}} = \pm \sqrt{\frac{m_k g(x - x_0)^2}{k_c k_{DA}^2 k_i^2}} \quad (8.9)$$

Now final shape of the matrices is:

$$A = \begin{bmatrix} -\frac{k_{fv}}{m_k} & -\frac{2g}{x_{ss} - x_0} \\ 1 & 0 \end{bmatrix}; B = \begin{bmatrix} -\frac{2k_{DA}\sqrt{k_c k_i^2 g}}{\sqrt{m_k}(x_{ss} - x_0)} \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & k_{AD}k_x \end{bmatrix}; D = \begin{bmatrix} 0 \end{bmatrix}$$

State-space representation can be converted to transfer function form. This is done through the solving of the following equation:

$$G(s) = C(sI - A)^{-1}B + D = \frac{1}{\det(sI - A)} C \text{adj}(sI - A)B + D \quad (8.10)$$

For this case of a SISO model, the resulting transfer function is only one in the form:

$$G(s) = \frac{b_0}{a_2 s^2 + a_1 s + a_0} \quad (8.11)$$

where partial coefficients are:

$$b_0 = -\frac{2k_{DA}k_{AD}k_x\sqrt{k_c k_i^2 g}}{\sqrt{m_k}(x_{ss} - x_0)} \quad (8.12)$$

$$a_0 = -\frac{2g}{x_{ss} - x_0} \quad (8.13)$$

$$a_1 = \frac{k_{fv}}{m_k} \quad (8.14)$$

$$a_2 = 1 \quad (8.15)$$

Transfer function gain is a static variable and it doesn't depend on the chosen operating point. Its value is $k = -2.4357[-]$ and it is computed from equation (8.16).

$$k = \frac{b_0}{a_0} = k_{DA}k_{AD}k_x k_i \sqrt{\frac{k_c}{m_k g}} \quad (8.16)$$

8.1 Selection of an operating point

Chosen operating point don't have to be necessarily suitable for linearization. It is a question, how adequate is the linearized approximation in comparison with the real system dynamics. First step is to specify value of the operating point parameters, in this case only one parameter is needed and this is to specify a position of the ball x_{ss} . Probably the first thought, that comes to everybody's mind, is to choose the operating point when the position of a ball is approximately in the middle of the space between the magnetic core and the head of the inductive sensor. Then the result value is: $x_{ss} = l/2 = 2.85 \cdot 10^{-3}[m]$.

8.2 Verification and validation of the linearized model

It is assumed that nonlinear model is sufficiently accurate and so given linearized model should be too, but it must be considered, that linear model will be accurate only, when he work near the operating point.

9 DISCRETIZATION OF CREATED MODELS

Because computer control is based on discrete-time computations, first thing what needs to be done is a discretization of used models.

9.1 Discrete linear state-space model

Discrete-time linear state-space model is given by equation (2.1). And its matrices Φ and Γ are computed based on principal discussed in a chapter dedicated to sampling a continuous-time linear state-space system. Final values of matrices and discrete-time state space model have the form:

$$\Phi = \begin{bmatrix} 0.9995 & 3.5980 \\ 0.0010 & 1.0018 \end{bmatrix}; \Gamma = \begin{bmatrix} 0.1060 \\ 0.0001 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 82.6852 \end{bmatrix}; D = \begin{bmatrix} 0 \end{bmatrix}$$

9.2 Discrete nonlinear state-space model

9.2.1 Euler method solution

System states are defined by equation (8.1), which refers to basic equation (7.18). Velocity is get as a solution of first order ODE and position as a second order ODE solution.

$$v_{n+1} = v_n + \Delta t \frac{(k_{DA} * k_i * u_{MU})^2 k_c}{m_k (x_n - x_0)^2} - \frac{k_{fv}}{m_k} v_n - g \quad (9.1)$$

$$x_{n+1} = x_n + \Delta t v_n + \frac{\Delta t^2}{2} \frac{(k_{DA} * k_i * u_{MU})^2 k_c}{m_k (x_n - x_0)^2} - \frac{k_{fv}}{m_k} v_n - g \quad (9.2)$$

9.2.2 Runge-Kutta method solution

Runge-Kutta RK4 method gives a solution when both states can be calculated from second order ODE solution, but results are better when only position is computed from second order ODE solution and velocity is computed as a first order ODE solution. Calculations for both orders are shown below.

First order ODE solution

For simpler orientation will be defined function F :

$$F(v, x, u_{MU}) = \frac{dv}{\Delta t} = \frac{(k_{DA} * k_i * u_{MU})^2 k_c}{m_k (x_n - x_0)^2} - \frac{k_{fv}}{m_k} v_n - g$$

$$v_{n+1} = v_n + dv; \quad (9.3)$$

$$\begin{aligned} dv_1 &= \Delta t(F(v_n, x_n, u_{MU})) \\ dv_2 &= \Delta t \left(F \left(v_n + \frac{dv_1}{2}, x_n, u_{MU} \right) \right) \\ dv_3 &= \Delta t \left(F \left(v_n + \frac{dv_2}{2}, x_n, u_{MU} \right) \right) \\ dv_4 &= \Delta t(F(v_n) + dv_3, x_n, u_{MU}) \\ dv &= (dv_1 + 2 * dv_2 + 2 * dv_3 + dv_4)/6 \end{aligned} \quad (9.4)$$

Second order ODE solution

We assume same function F and also the fact that:

$$\frac{dx}{\Delta t} = v$$

Computation is done for both state variables:

$$\begin{aligned} v_{n+1} &= v_n + dv \\ x_{n+1} &= x_n + dx \end{aligned} \quad (9.5)$$

where:

$$\begin{aligned} dv_1 &= \Delta t(F(v_n, x_n, u_{MU})) \\ dv_2 &= \Delta t \left(F \left(v_n + \frac{dv_1}{2}, x_n + \frac{dx_1}{2}, u_{MU} \right) \right) \\ dv_3 &= \Delta t \left(F \left(v_n + \frac{dv_2}{2}, x_n + \frac{dx_2}{2}, u_{MU} \right) \right) \\ dv_4 &= \Delta t(F(v_n + dv_3, x_n + dx_3, u_{MU})) \\ dv &= \frac{(dv_1 + 2 * dv_2 + 2 * dv_3 + dv_4)}{6} \end{aligned} \quad (9.6)$$

$$dx_1 = \Delta t v_n$$

$$\begin{aligned} dx_2 &= \Delta t \left(v_n + \frac{dv_1}{2} \right) \\ dx_3 &= \Delta t \left(v_n + \frac{dv_2}{2} \right) \end{aligned} \quad (9.7)$$

$$\begin{aligned} dx_4 &= \Delta t \left(v_n + \frac{dv_3}{2} \right) \\ dv &= \frac{(dv_1 + 2 * dv_2 + 2 * dv_3 + dv_4)}{6} \end{aligned}$$

For the creation of EKF a discrete-time state space model given by Euler method will be used. The reason is, because in EKF partial derivatives must be made, and if Runge-Kutta method will be used, then these derivatives will be more difficult and time need for computation will be longer. In practice Runge-Kutta method gives better results, because its a higher order method, but for the price of higher computation time.

10 STATE OBSERVER

In order to know what is going on inside the system and to be able to control the input, the system must be observable and controllable. So before creation of state observer it is good to check whether the system is observable or not. Also the check of controllability is done.

10.1 Observability and controllability

Created linear system is observable if observability matrix has a full rank equal to the dimension of state vector n . Observability matrix is given by the result of the equation (10.1) and its dimensions are $nq \times n$. Rank of the observability matrix can be determined from the matrix modified into the row echelon form. Number of nonzero rows gives the rank of the matrix. The rank of the matrix is n so the system is observable.

$$Obsv = \begin{bmatrix} C \\ C\Phi \\ C\Phi^2 \end{bmatrix} \quad (10.1)$$

The system is controllable if controllability matrix has a full rank equal to the dimension of state vector n . Controllability matrix is given by the result of the equation (10.2) and its dimensions are $n \times np$. Rank of the controllability matrix is determiner in the same way as previous. The rank is also n and the system is controllable.

$$Ctrb = \begin{bmatrix} \Gamma & \Phi B & \Phi^2 \Gamma \end{bmatrix} \quad (10.2)$$

It was determined, that system is observable and controllable so state observer can be created. As a state observer was used Kalman filter in classical and extended version and L_2E FIR filter.

10.2 Comparison of filters

All filter were created as a function blocks in Simulink environment. This blocks are Embedded MATLAB functions with implemented computations of the above mentioned equations related to given filter. Simulation diagram are placed in appendices. For EKF it is not necessary to compute with respect to chosen operating point, so simulation diagram has a different structure, than the other two filters. For filters it is assumed, that measurement noise v was determine from measurement when input signal was zero. Output signal is loaded with noise and his variance was settled as a standard deviation from measured data. Since there is one output value, measurement noise v is also a single value $v = 1.5766 \cdot 10^{-4} [MU/sec]$. Then covariance R for Kalman filters is

$$R = v^2 = (1.5766 \cdot 10^{-4})^2$$

Like was said before it is assumed that process noise w have the same value as v then w have the following form $w = \begin{bmatrix} v & v \end{bmatrix}$ and matrix Q used in Kalman filters is defined as:

$$Q = E(ww^T) = \begin{bmatrix} (1.5766 \cdot 10^{-4})^2 & (1.5766 \cdot 10^{-4})^2 \\ (1.5766 \cdot 10^{-4})^2 & (1.5766 \cdot 10^{-4})^2 \end{bmatrix}$$

Test of filters estimations was done on nonlinear model, which was controlled by simple PID controller with parameters given from simulation demo of Humusoft. All simulations have simulation time $T = 4[s]$, sampling time $T_s = 0.001[s]$ and examined parameters were elapsed time and the value of performance criterion. Criterion have the following form:

$$S_x = \sum_{k=1}^N [x_k - \hat{x}_k]^2 \tag{10.3}$$

where x_k represents the current state of the nonlinear model and \hat{x}_k are the current state estimates. Prediction horizon of LEF filter was chosen as $N_f = 5$.

10.2.1 Simulations results

LEF filter

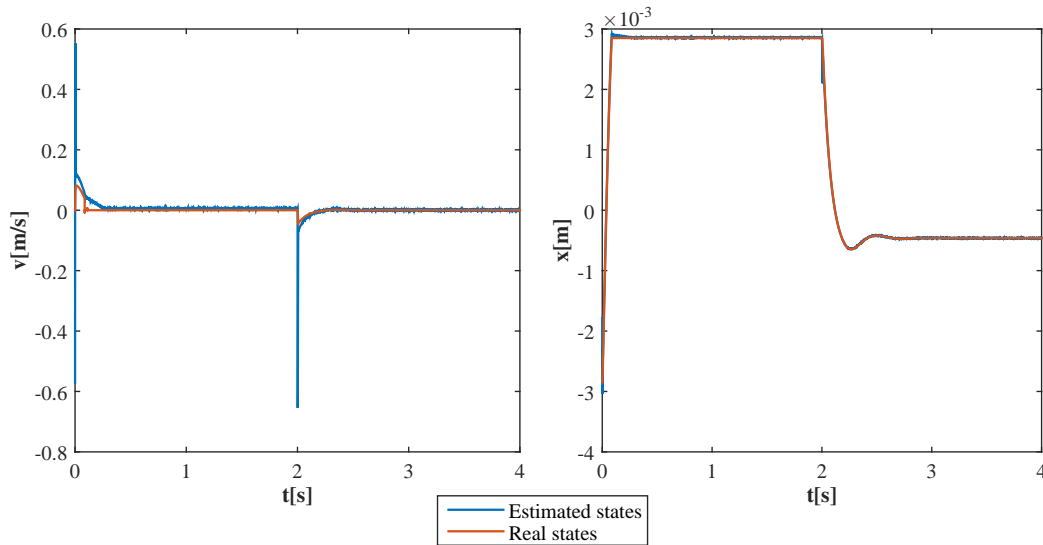


Fig. 10.1 Comparison of estimated state given by LEF with real state

Kalman filter

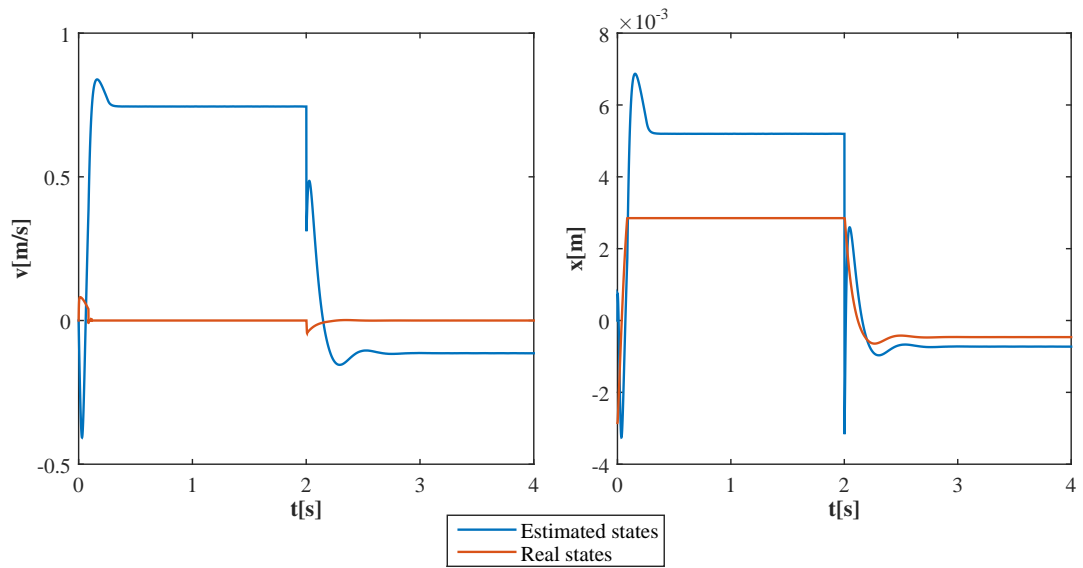


Fig. 10.2 Comparison of estimated state given by KF with real state

Extended Kalman filter

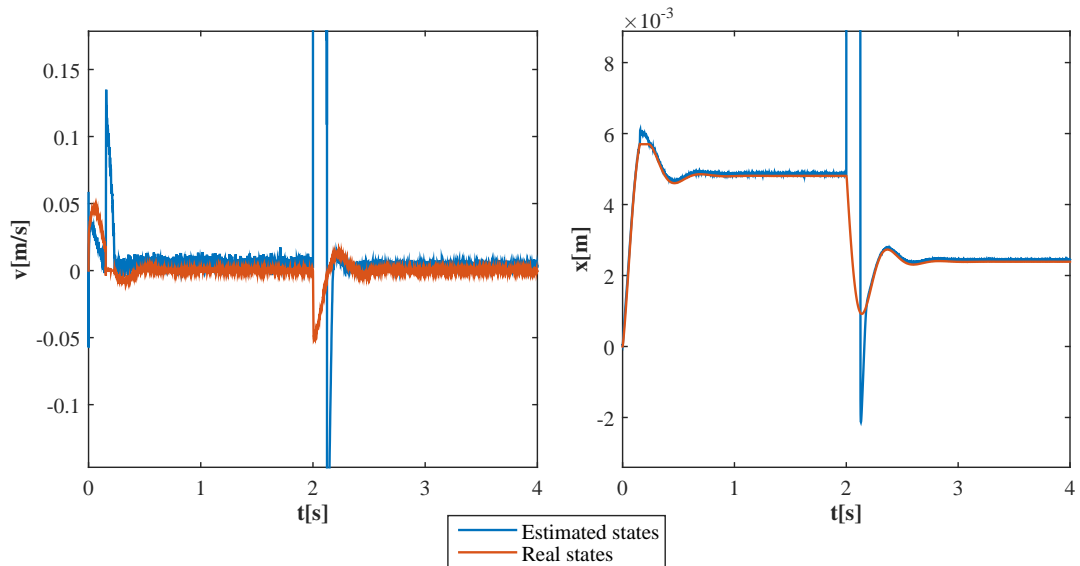


Fig. 10.3 Comparison of estimated state given by EKF with real state

10.2.2 Performance results

Tab. 10.1 Performance of filters

	<i>LEF filter</i>	<i>Kalman filter</i>	<i>Extended Kalman filter</i>
<i>Time</i> [s]	1.3072	1.9996	1.9112
S_v [-]	2.3040	1121	$1.3600 \cdot 10^4$
S_x [-]	$4.1690 \cdot 10^{-6}$	0.01259	1.005

It was found out, that LEF filter works very well and fast even on relatively small prediction horizon, whereas KF proved, that he is not suitable for nonlinear version of the magnetic levitation model. Also EKF did not work properly with tested period. If sampling period is increased then results are better but computation demands are higher and performance still don't have qualities of LEF filter. Reason why EKF didn't work properly, could have been wrong implementation or inaccuracy of Euler method. One thing is clear and that the all filters give much worse result for estimation of velocity, which leads to conclusion, that there is some inaccuracy in created models. Based on the results, LEF filter was chosen for model predictive control algorithm.

11 MODEL PREDICTIVE CONTROL OF CE152 SYSTEM

For the control of the CE152 magnetic levitation system an GPC algorithm described in theoretical part of this thesis was used. Prediction of output values is given by equation (4.12). And control law is given by minimisation of (4.1).

11.1 Control law

Cost function in a second norm can be converted as follows:

$$J = E_k^T E_k + \Delta U_k^T \lambda \Delta U_k \quad (11.1)$$

If we substitute for Y_k from (4.17) and for ΔU_k from (4.16) we get:

$$J = (W_k - Px_k + HU_k + Ld_k)^T (W_k - Px_k + HU_k + Ld_k) + (R_d u_{k-1} + R_r U_k)^T (R_d u_{k-1} + R_r U_k) \quad (11.2)$$

Let's consider that:

$$W_{dk} = W_k - Px_k + Ld_k \quad (11.3)$$

than equation (11.2) look like this:

$$J = (W_{dk} + HU_k)^T (W_{dk} + HU_k) + (R_d u_{k-1} + R_r U_k)^T (R_d u_{k-1} + R_r U_k) \quad (11.4)$$

If equation is expanded we get:

$$J = W_{dk}^T W_{dk} - 2W_{dk}^T HU_k + U_k^T H^T HU_k + \lambda (u_{k-1} R_d^T R_d u_{k-1} + 2u_{k-1} R_d^T R_p U_k + U_k^T R_p^T R_p U_k) \quad (11.5)$$

If we now differentiates J with respect to U_k we get:

$$\frac{\partial J}{\partial U_k} = 2 (-W_{dk}^T H + \lambda u_{k-1} R_d^T R_p)^{-1} 2U^T (H^T H + \lambda R_p^T R_p) \quad (11.6)$$

When we put above equation equal to zero we get control law:

$$U_k^T = (W_{dk}^T H - \lambda u_{k-1} R_d^T R_p)^{-1} (H^T H + \lambda R_p^T R_p) \quad (11.7)$$

For implementation of the MPC controller was used again MATLAB Embedded function block in Simulink environment. Control law is then computed in this function. Input values are current state estimate x_k , past control value u_{k-1} , compensation constant d_k and reference trajectory represented by vector of values of desired future position W_k . Output value is new action value u_k . Simulation diagram are placed in appendices.

11.2 Simulation of the control with linear model

On simulation of the control with linear model will be discussed influence of optional parameters, which are prediction horizon N_y and control horizon N_u of the controller, prediction horizon for the estimator N_f and value of weighting coefficient λ .

Three simulation sets with different parameters will follows, but first will be discussed influence of parameter change. Longer prediction horizon N_y and control horizon N_u gives better performance but for the cost of longer computation. If prediction horizon and control horizon are decreased, we can see more oscillations during the control cycle. That means that parameters of the horizons have a big influence on a control law. We can say, that low horizon means low information value and this means more uncertainty. Proper weighting coefficient λ can slow down the control cycle and minimise number of overshoots, or eliminate them completely.

- Parameters:

$$N_y = 15; N_u = 10; N_f = 15; \lambda = 0.001$$

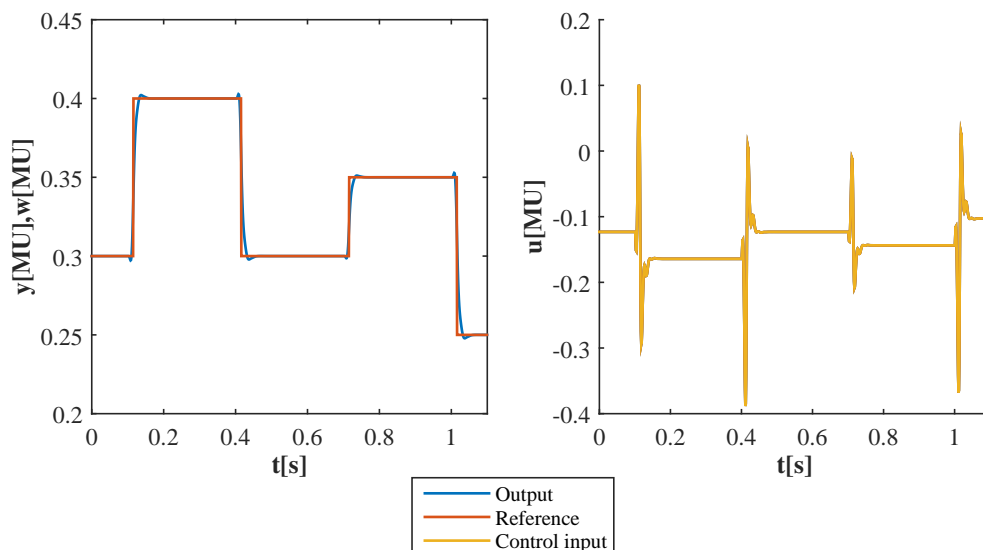


Fig. 11.1 MPC control of linear model with parameters

- Parameters:

$$N_y = 15; N_u = 10; N_f = 15; \lambda = 1$$

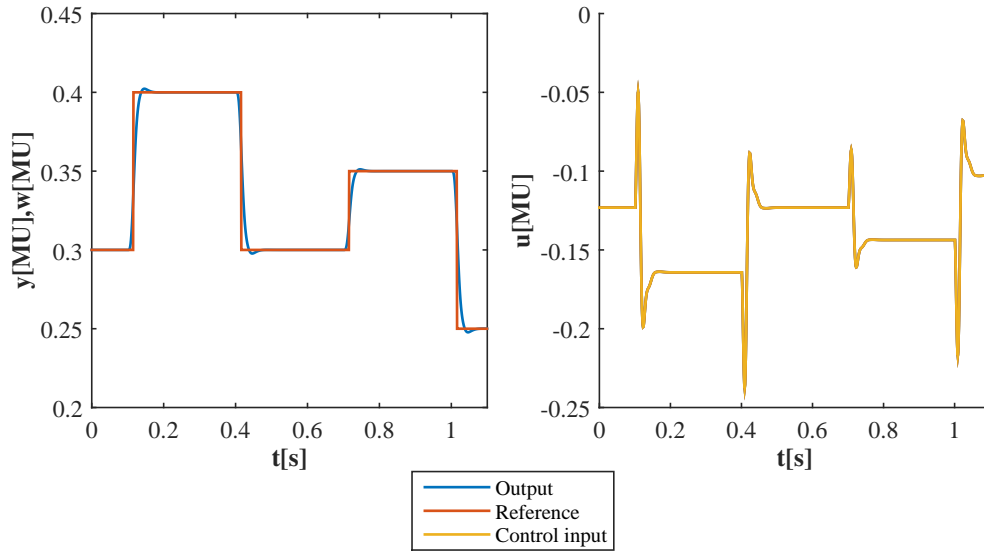


Fig. 11.2 MPC control of linear model with parameters:

- Parameters:

$$N_y = 10; N_u = 2; N_f = 15; \lambda = 1$$

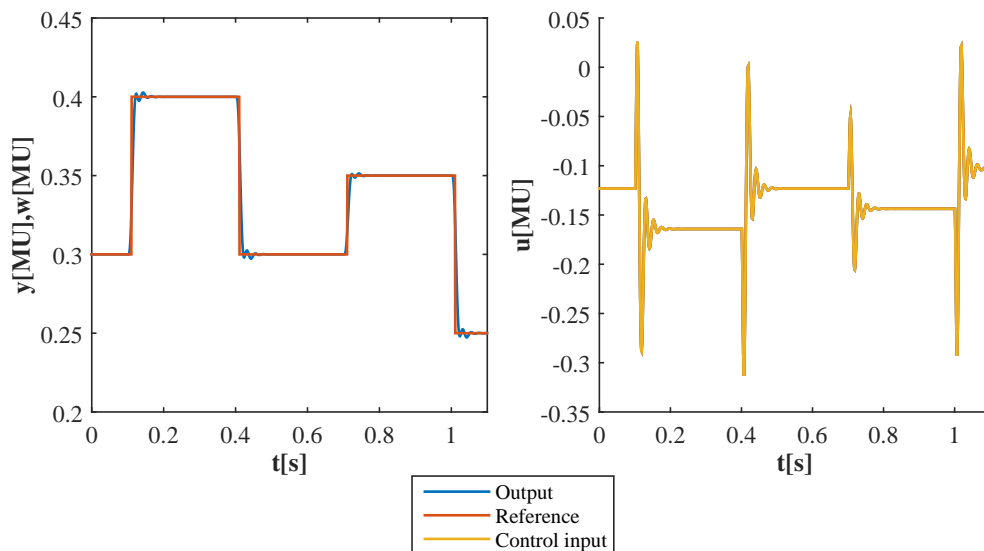


Fig. 11.3 MPC control of linear model with parameters

11.3 Simulation of the control with nonlinear model

- Parameters:

$$N_y = 15; N_u = 15; N_f = 15; \lambda = 5$$

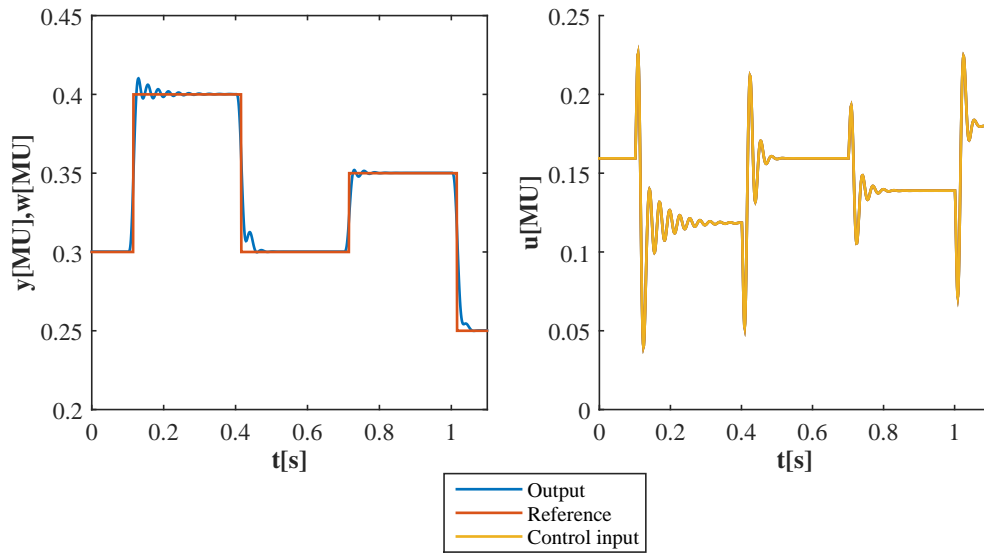


Fig. 11.4 MPC control of nonlinear model with parameters

- Parameters:

$$N_y = 15; N_u = 15; N_f = 15; \lambda = 1$$

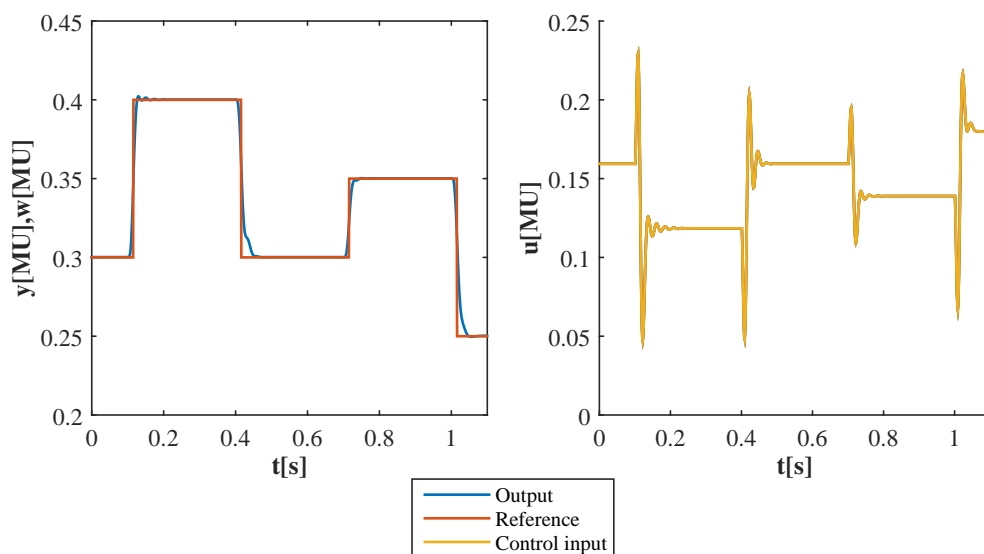


Fig. 11.5 MPC control of nonlinear model with parameters

- Parameters:

$$N_y = 15; N_u = 15; N_f = 15; \lambda = 1$$

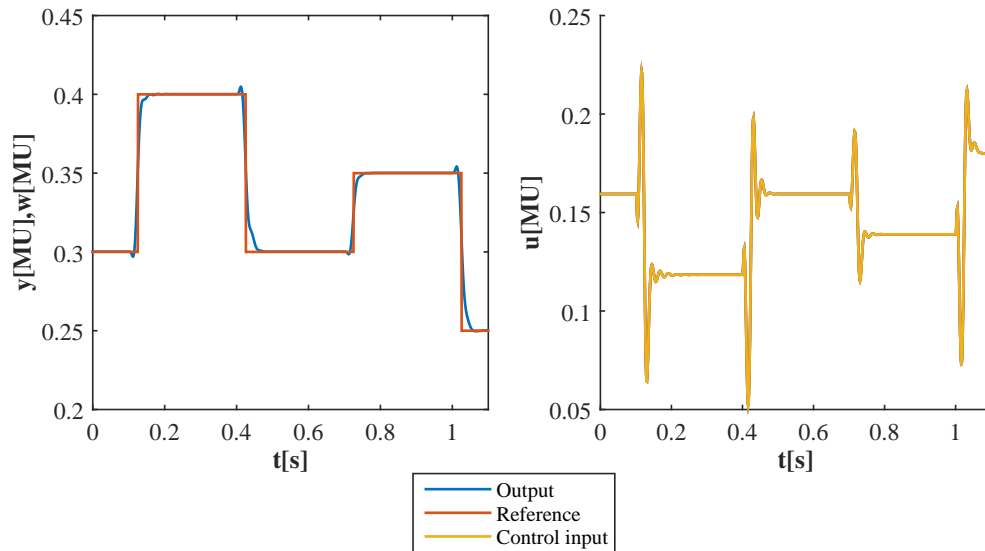


Fig. 11.6 MPC control of nonlinear model with parameters

Good performance of MPC controller was proved even for nonlinear model, where with bigger horizons are final results very good. Reaction on change of weighting coefficient λ is good visible from Fig. 11.6 .

11.4 Control of the real system

Control on the real system was also good but compensation coefficient d_k must have been increased manually, because some deviation still remained. It was found out that this deviation corresponds to prediction error, but it wasn't found out why. It must be said that control horizons should be chosen carefully, because when they are too high PC restarts without any warning.

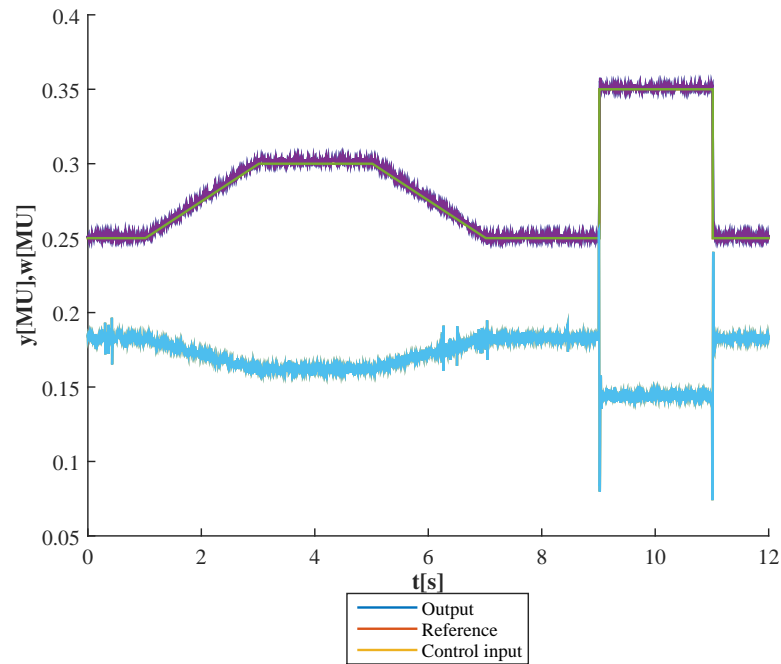


Fig. 11.7 MPC control of real model with parameters

11.5 Comparison with a simple controllers

Simulations was done with MPC controller, simple PID controller and PID controller in a given by Humusoft. All simulations was done under the same conditions (same starting point). From the comparison of controllers, designed MPC controller came out as the winner.

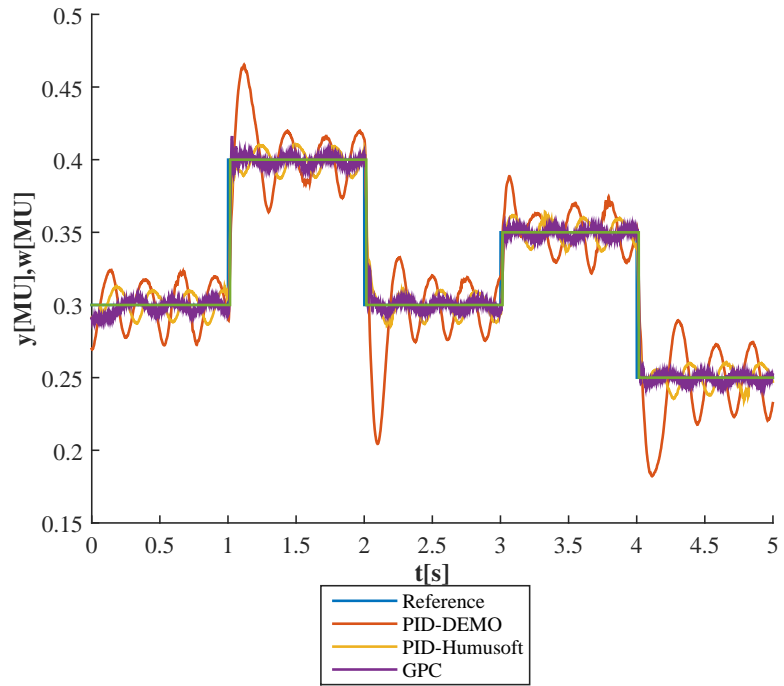


Fig. 11.8 Comparison of output values when desired trajectory is in the step form

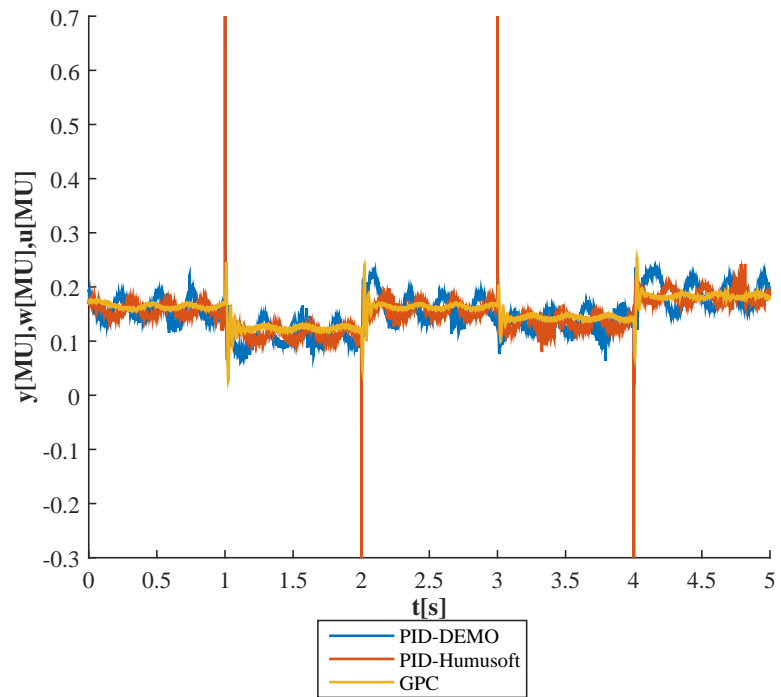


Fig. 11.9 Comparison of control values when desired trajectory is in the step form

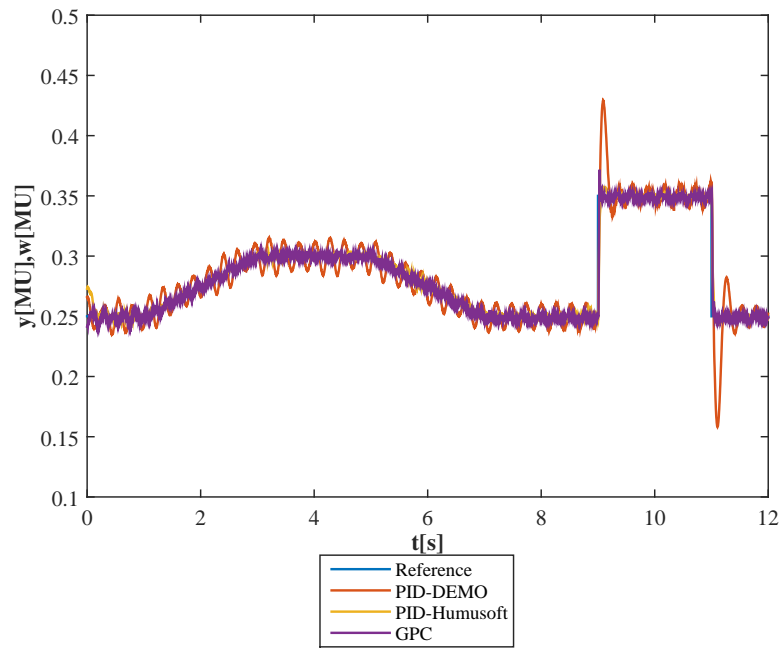


Fig. 11.10 Comparison of output values when desired trajectory is in the form of the step and the ramp

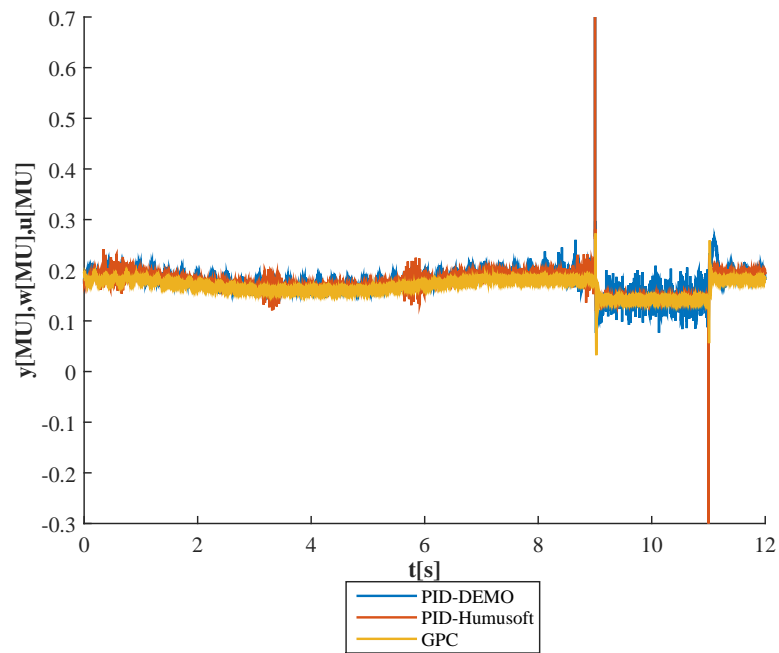


Fig. 11.11 Comparison of control values when desired trajectory is in the form of the step and the ramp

CONCLUSION

Model predictive control represents a modern and ever-growing methodology of control. Initial development of MPC was not uniform, but over time it was unified. Main disadvantage of MPC was computational requirements, and because of low performance hardware it was initially deployed only for slow processes. To these days there is a lot of algorithms and also a lot of approaches, which minimize computational demands and even performance of hardware has highly increased. These two factors caused, that MPC could have been applied even for fast real-time processes such as CE152 magnetic levitation system. It was proved that with some drawbacks even basic algorithm is suitable for control of this kind of system. Also it was proved how important is adequate model of the system.

Created linear model is sufficiently accurate only nearby an chosen operating point. In this area also created MPC controller works the best. Created MPC controller is based on state-space model and as such he has needed an estimator of his current states. It was find out, that also proper choice of the estimator has got a big influence on control. Classical Kalman filter showed up to be absolutely inappropriate and his Extended version for nonlinear systems too. Only LEF filter based on MHE concept proved good behaviour and he was used for the MPC control algorithm. Tested control algorithm was GPC algorithm for state-space models. From the perspective of control performance could be said that this algorithm works just fine, but regulation is far from optimal and there is a space to improve it. If there would have been more time then into the MPC strategy could have been implicated solution that will handling the constraints of the input and the state values. This can be the subject for the future work.

A chapter in itself were the used PC and the MATLAB/SIMULINK environment with his Real Time Windows Target mode. Low level of chosen horizons didn't cause any problem, but higher level of horizons leads to hard restart of the system without any warning. It was found out, that even behaviour of the system is different between two launches of the MATLAB/SIMULINK environment. The reason why this is happening, could have been another subject to deeper scrutiny.

Future work could have been dedicated to application of some kind of quadratic programming or LMI. These methods are implemented in MATLAB, but they can't be used with RTWT toolbox, so these algorithms must be programmed.

REFERENCES

- [1] LJUNG, Lennart. *System identification: theory for the user*. 2nd. ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999, 609 p. [cit. 2015-02-04]. ISBN 0-13-656695-2.
- [2] ASTROM, Karl J a Bjorn WITTENMARK. *Computer-controlled systems: theory and design*. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 1997, xiv, 557 p. [cit. 2015-02-04]. ISBN 0133148998.
- [3] Hartikainen, Jouni, Arno Solin, and Simo Särkkä. *Optimal filtering with Kalman filters and smoothers*. Department of Biomedica Engineering and Computational Sciences, Aalto University School of Science: Greater Helsinki, Finland 16 (2011). [cit. 2015-02-25].
- [4] KWON, W a S HAN. *Receding horizon control: model predictive control for state models*. London: Springer, 2005, xiv, 380 p. [cit. 2015-04-05] ISBN 1846280176-.
- [5] ORUKPE, P. E. Model Predictive Control Fundamentals. *Nigerian Journal of Technology (NIJOTECH)*. 2012, vol. 41, no. 2, p. 139-148 [cit. 2015-04-15]. ISSN 1115-8443
- [6] SCHLEGEL, Miloš a Jaroslav SOBOTA. Prediktivní regulátor pro průmyslovou praxi. *AUTOMA - časopis pro automatizační techniku*. 2007, no. 2, p. 12-16. [cit. 2015-04-15]. Available from: <http://automa.cz/download/au020712.pdf>
- [7] BOBÁL, Vladimír. *Adaptivní a prediktivní řízení*. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2008, 134 p. [cit. 2015-04-06]. ISBN 978-80-7318-662-3.
- [8] CAMACHO, E a C BORDONS. *Model predictive control*. 2nd. ed. New York: Springer, 2004, 405 p. [cit. 2015-04-06]. ISBN 978-0-85729-398-5.
- [9] HABER, Robert, R BARS a Ulrich SCHMITZ. *Predictive control in process engineering: From the basics to the applications* [online]. Weinheim: Wiley-VCH, c2011, 600 p. [cit. 2015-04-06]. ISBN 978-3-527-63624-2
- [10] S.Joe Qin, Thomas A. Badgwell, *A survey of industrial model predictive control technology*, Control Engineering Practice, 2003, vol. 11, is. 7, p. 733-764. [cit. 2015-04-06]. ISSN 0967-0661. <http://www.sciencedirect.com/science/article/pii/S0967066102001867>
- [11] TATJEWSKI, Piotr. *Advanced control of industrial processes: structures and algorithms*. London: Springer, 2007, xix, 332 p. [cit. 2015-04-06]. ISBN 978-1-84628-634-6.
- [12] ROSSITER, J. *Model-based predictive control: a practical approach*. Boca Raton: CRC Press, 2005, 344 p. [cit. 2015-04-10]. ISBN 0-8493-1291-4.
- [13] HUMUSOFT s.r.o. *CE 152 Magnetic levitation model – educational manual*. Prague: HUMUSOFT, 2002. [cit. 2015-04-15].

LIST OF ABBREVIATIONS

Symbols:

A	continuous state/system matrix
B	continuous input matrix
C	output matrix
D	feedthrough/feedforward matrix
exp	matrix exponential
f	mathematical function
F_a	accelerating force
F_a	accelerating force
F_g	electromagnetic force
F_m	gravitational force
Φ	discrete state/system matrix
g	gravitational acceleration force/mathematical function
Γ	discrete input matrix
H	gain matrix
i	electric current
J	cost function
k_{AD}	A/D converter constant
k_{am}	amplifier gain
k_c	coil constant
k_{DA}	D/A converter constant
k_{fv}	dumping constant
k_s	current sensor gain
k_x	position sensor gain
L	coil inductance/gain matrix
\mathcal{L}	Laplace transform
λ	eigenvalues/weighting coefficient
m	ball mass
N_f	filter prediction horizon
N_u	control horizon
N_y	controller prediction horizon
O	error per step
P_k	state covariance

Q_k	covariance of process noise
R	coil resistance
R_s	resistance of feedback resistor
R_k	covariance of measurement noise
S_x	criterion for state space estimates
u_0	D/A converter offset
u	D/A converter output signal/coil input voltage/input value
u_k	current input value
u_{k-1}	previous input value
u_m	coil voltage
u_{MU}	D/A converter input signal
U_k	vector of input values
Δu_k	penalization of u
Δt	sampling period
V_k	vector of measurement disturbances
W_k	vector of process disturbances/vector of desired values
x	ball position
x_0	coil offset
x_k	current state
X_k	vector of states
\hat{x}	current state estimate
y	A/D converter input signal/position sensor voltage/output value
y_0	position sensor offset
y_k	current output
y_{MU}	A/D converter output signal
y_{MU0}	A/D converter offset
Y_k	vector of current outputs

Acronyms:

<i>DMC</i>	Dynamic Matrix control
<i>EHAC</i>	Extended Horizon Adaptive Control
<i>EPSAC</i>	Extended Prediction Self Adaptive Control
<i>FIR</i>	Finite Impulse response
<i>GPC</i>	Generalized Predictive Control
<i>IDCOM</i>	Identification command
<i>LQG</i>	Linear Quadratic Gaussian
<i>LMI</i>	Linear Matrix Inequality
<i>MAC</i>	Model Algorithmic control
<i>MAC</i>	Multi Input Multi Output
<i>MHC</i>	Moving Horizon Control
<i>MPHC</i>	Model Predictive Heuristic Control
<i>MPC</i>	Model Predictive Control
<i>MPC – NSL</i>	MPC with Nonlinear with Successive Linearization
<i>MPC – NPL</i>	MPC with Nonlinear Prediction and Linearization
<i>MU</i>	Machine Unit
<i>ODE</i>	Ordinary Differential Equation
<i>PC</i>	Personal Computer
<i>PFC</i>	Predictive Functional Control
<i>QDMC</i>	Quadratic Dynamic Matrix Control
<i>RHC</i>	Receding Horizon Control
<i>RMPC</i>	Robust Multivariable Predictive Control
<i>SISO</i>	Single Input Single Output
<i>SMCA</i>	Setpoint Multivariable Control
<i>SMOC</i>	Shell Multivariable Optimizing Controller

LIST OF FIGURES

Fig. 2.1	Principal scheme of the computer control system	12
Fig. 4.1	Model predictive control strategy	23
Fig. 4.2	Basic structure of model predictive control	23
Fig. 5.1	Principal scheme of the magnetic levitation model	32
Fig. 6.1	Step response on different input value and reaction on input value change to zero	33
Fig. 7.1	Simulation diagram of the D/A converter	34
Fig. 7.2	Simulation diagram of the A/D converter	35
Fig. 7.3	Simulation diagram of the position sensor	36
Fig. 7.4	Internal structure of the power amplifier	37
Fig. 7.5	The power amplifier detailed simulation diagram	38
Fig. 7.6	The power amplifier simplified simulation diagram	39
Fig. 7.7	The ball and coil simulation diagram	40
Fig. 7.8	The ball and coil simulation diagram	41
Fig. 7.9	Second order integrator diagram	42
Fig. 7.10	Hard stop and bouncing block diagram	42
Fig. 7.11	Interconnection of subsystem blocks	43
Fig. 7.12	Subsystem block of the CE152 magnetic levitation system	43
Fig. 7.13	Comparison of response of the model and the system	44
Fig. 7.14	Comparison of the output values of the real system and created model	44
Fig. 8.1	State space system	45
Fig. 10.1	Comparison of estimated state given by LEF with real state	52
Fig. 10.2	Comparison of estimated state given by KF with real state	53
Fig. 10.3	Comparison of estimated state given by EKF with real state	53
Fig. 11.1	MPC control of linear model with parameters	56
Fig. 11.2	MPC control of linear model with parameters:	57
Fig. 11.3	MPC control of linear model with parameters	57
Fig. 11.4	MPC control of nonlinear model with parameters	58
Fig. 11.5	MPC control of nonlinear model with parameters	58
Fig. 11.6	MPC control of nonlinear model with parameters	59
Fig. 11.7	MPC control of real model with parameters	60
Fig. 11.8	Comparison of output values when desired trajectory is in the step form	61
Fig. 11.9	Comparison of control values when desired trajectory is in the step form	61
Fig. 11.10	Comparison of output values when desired trajectory is in the form of the step and the ramp	62
Fig. 11.11	Comparison of control values when desired trajectory is in the form of the step and the ramp	62

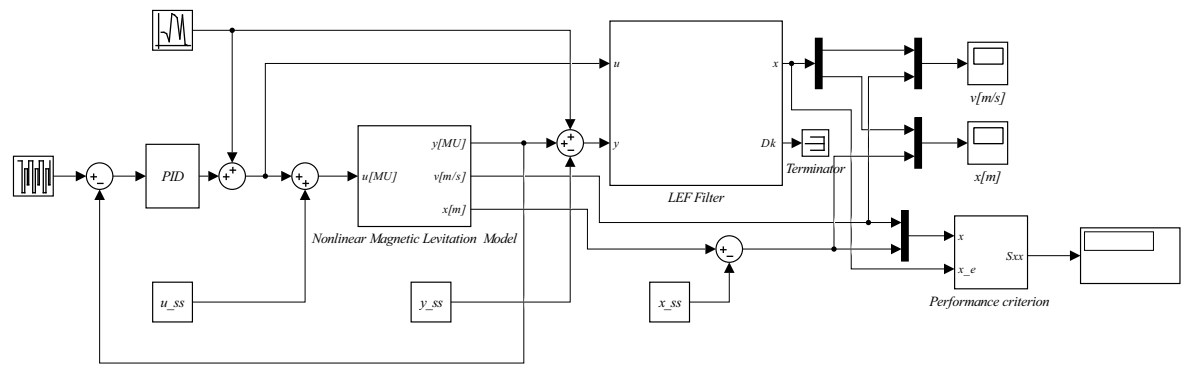
LIST OF TABLES

Tab. 7.1	Calibration data of the position sensor	36
Tab. 7.2	Two point calibration data	41
Tab. 10.1	Performance of filters	54

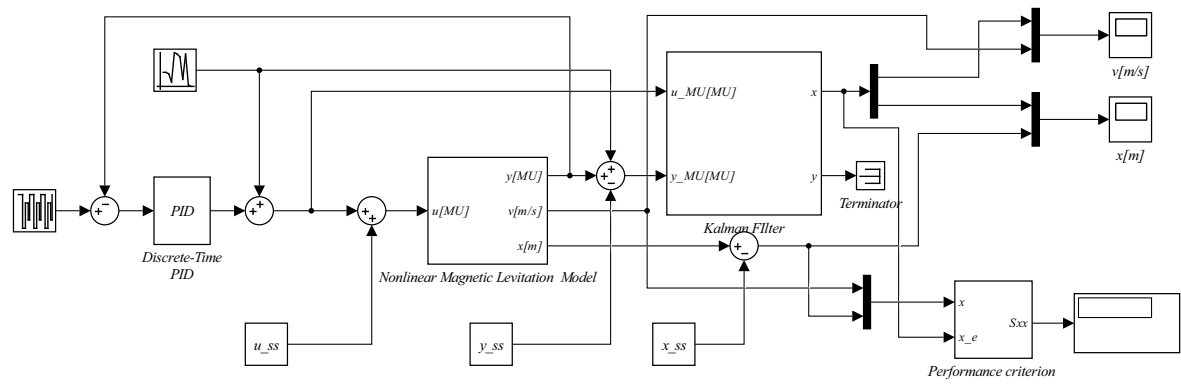
LIST OF APPENDICES

- P I. LEF filter
- P II. Kalman filter
- P III. Extended Kalman filter
- P IV. GPC diagram for linear model
- P V. GPC diagram for nonlinear model
- P VI. GPC diagram for the real system

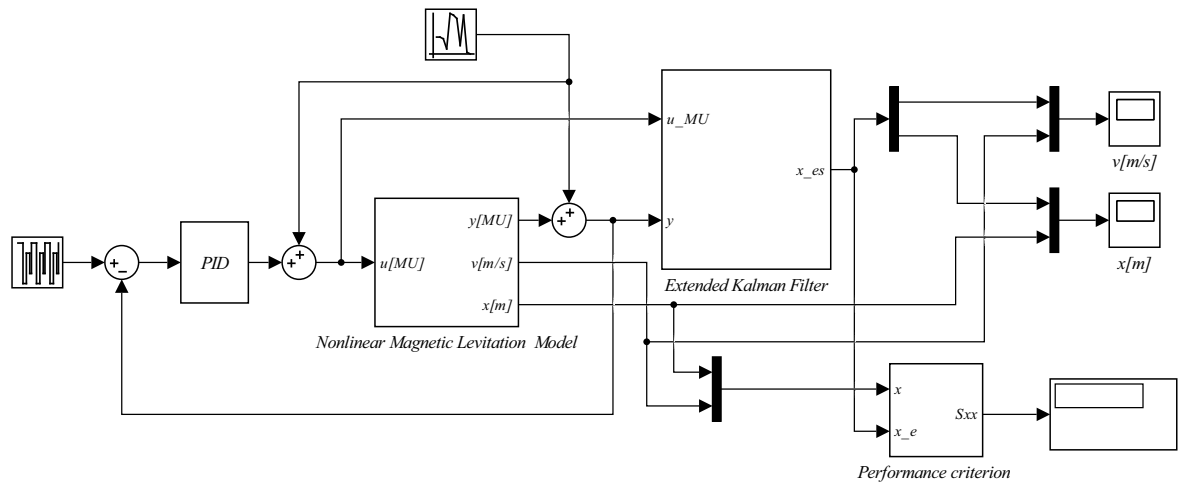
APPENDIX P I. LEF FILTER



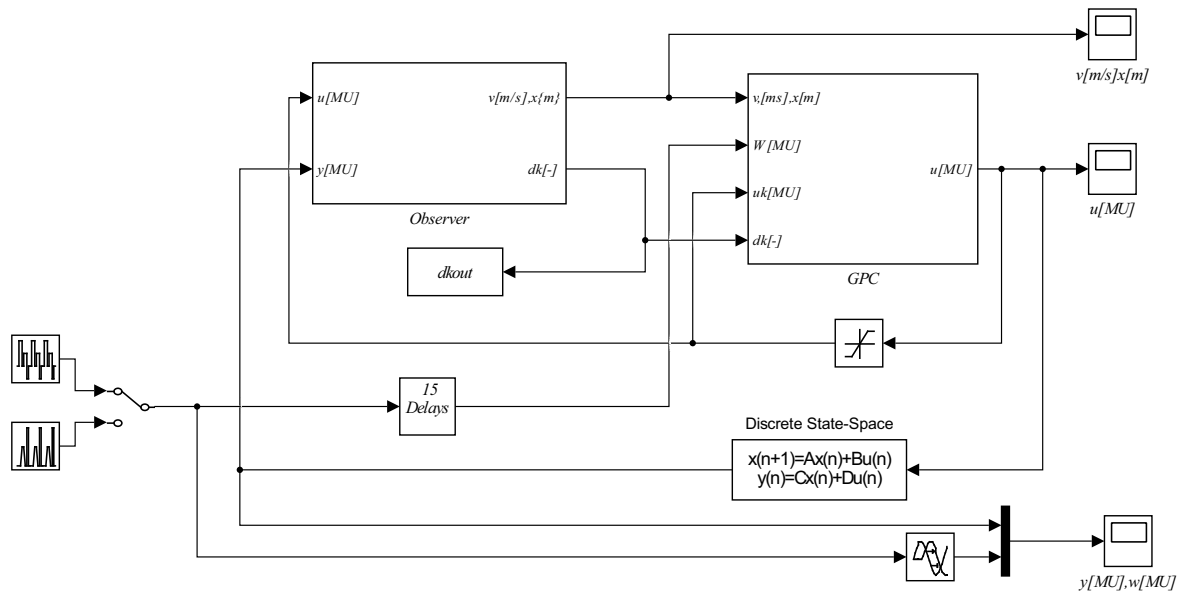
APPENDIX P II. KALMAN FILTER



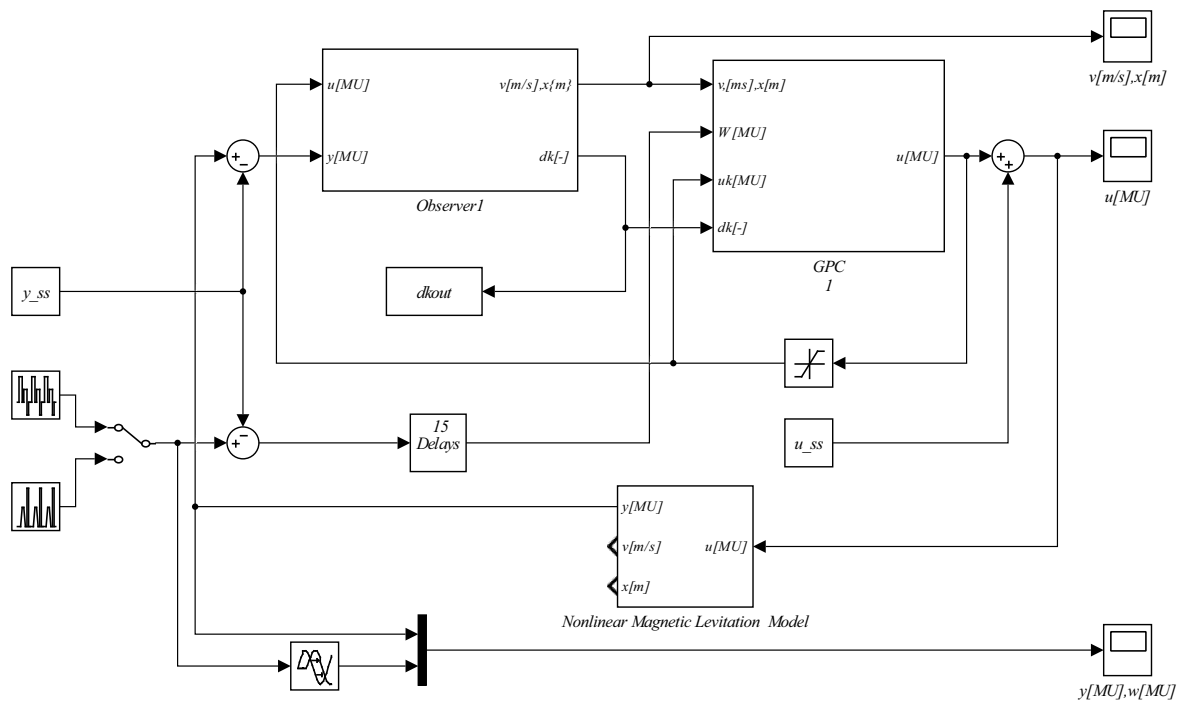
APPENDIX P III. EXTENDED KALMAN FILTER



APPENDIX P IV. GPC DIAGRAM FOR LINEAR MODEL



APPENDIX P V. GPC DIAGRAM FOR NONLINEAR MODEL



APPENDIX P VI. GPC DIAGRAM FOR THE REAL SYSTEM

