


Digital Control of CE 151 Ball & Plate Model

Bc. Ľuboš Spaček

Master's thesis
2016

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Ľuboš Spaček**
Osobní číslo: **A14399**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Automatické řízení a informatika**
Forma studia: **prezenční**

Téma práce: **Číslicové řízení laboratorního modelu CE 151 Kulička na ploše**
Téma anglicky: **Digital Control of CE 151 Ball & Plate Model**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Seznamte se s laboratorním modelem Kulička na ploše CE 151.
3. Provedte experimentální identifikaci laboratorního modelu a na jejím základě navrhnete vhodné řídicí algoritmy.
4. Navržené algoritmy ověřte v simulačních podmínkách v programovém prostředí MATLAB/SIMULINK.
5. S přihlédnutím na simulační ověřování provedte řízení laboratorního modelu v reálném čase.
6. Provedte diskusi výsledků simulačního ověřování i řízení v reálném časem čase.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. HUMUSOFT. CE 151 Ball & Plate Apparatus User's Manual. Praha, 2006.
2. BOBÁL, V. Identifikace systémů. Zlín: Univerzita Tomáše Bati ve Zlíně. Akademické centrum, 2009, ISBN 978-80-7318-888-3.
3. BOBÁL, V. Adaptivní a prediktivní řízení. Zlín: Univerzita Tomáše Bati ve Zlíně. Akademické centrum, 2008, ISBN 978-80-7318-662-3.
4. BOBÁL, V., P. CHALUPA, P. DOSTÁL a M. KUBALČÍK. Digital Control of Unstable and Integrating Time-delay Processes. International Journal of Circuits, Systems and Signal Processing. 2014, (8), 9. ISSN 1998-4464.
5. LANDAU, Ioan D. Digital control systems design, identification and implementation. 1st ed. London: Springer, 2007. ISBN 978-184-6280-566.

Vedoucí diplomové práce:

prof. Ing. Vladimír Bobál, CSc.

Ústav řízení procesů

Datum zadání diplomové práce:

19. února 2016

Termín odevzdání diplomové práce:

25. května 2016

Ve Zlíně dne 19. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

I hereby declare that:

- I understand that by submitting my Diploma thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Diploma Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Diploma/Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlin, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Diploma Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlin has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Diploma Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlin with a view to the fact that Tomas Bata University in Zlin must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Diploma Thesis include the use of software provided by Tomas Bata University in Zlin or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Diploma Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Diploma Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlin; dated: 16.05.2016


.....
Student's Signature

ABSTRAKT

Číslicové riadenie je dôležitým prvkom dnešných technológií a priemyselného prostredia. Táto práca prezentuje číslicové riadenie nestabilného systému guľôčky na plošine, počínajúc matematickým modelom a končiac riadením reálneho modelu. Existuje mnoho prístupov návrhu regulátorov, pričom postup použitý v tejto práci je založený na minimalizácii lineárneho kvadratického LQ kritéria pre 2DOF štruktúru regulátora s využitím spektrálnej faktorizácie, ktorého výsledkom je takmer optimálne nastavený a relatívne robustný regulátor schopný poskytnúť dobré sledovanie žiadanej hodnoty a potlačenie poruchy. Aby bola voľba žiadanej hodnoty zaujímavejšia a automatická, tak bolo na plošine zostrojené jednoduché 2D bludisko. Počítač automaticky a algoritmicke vyrieši toto bludisko, čím sa získa žiadaná dráha guľôčky. Pre zjednotenie celého riadiaceho procesu bolo tiež navrhnuté grafické užívateľské prostredie. Výsledky prezentované v tejto práci sú sľubné a dokazujú, že LQ regulátor je veľmi vhodný pre tento typ aplikácie.

Kľúčové slová: guľôčka na ploche, číslicové riadenie, CE 151, LQ metóda, spektrálna faktorizácia, riadenie v reálnom čase, riešenie bludiska, watershed transformácia

ABSTRACT

The digital control is an important aspect of today's technologies and industrial environment. This thesis presents the digital control of an unstable Ball & Plate system from the mathematical model to real-model control. There are many different controller design approaches and the one used here is based on the minimization of linear quadratic LQ criterion for 2DOF controller structure using spectral factorization, which results in almost optimal and relatively robust controller able to provide good reference tracking and disturbance rejection. To make the reference signal selection interesting and automatic, a simple 2D maze was constructed on top of the plate. The computer automatically and algorithmically solves the maze, thus obtaining the desired trajectory of the ball. A graphical user interface was also designed to encapsulate the model control. Results shown in this thesis are promising and prove that the LQ controller is a very suitable for this model.

Keywords: Ball & Plate, digital control, CE 151, LQ method, real-time control, spectral factorization, maze solving, watershed transform

ACKNOWLEDGEMENTS

I would first like to thank my thesis supervisor prof. Ing. Vladimír Bobál, CSc. for his valuable notes and suggestions. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it.

I must also express my very profound gratitude to my family and friends for providing me with unfailing support and encouragement throughout my years of study.

I hereby declare that the print version of my Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

Motto

“The pleasure of finding things out.”

-- Richard Phillips Feynman

CONTENTS

INTRODUCTION	8
I THEORY	9
1 CE151 BALL & PLATE APPARATUS.....	10
1.1 TECHNICAL DETAILS	11
1.2 ALTERNATIVE SOLUTIONS AND COMPARISON.....	12
1.2.1 Pivot point	12
1.2.2 Actuator - plate connection	12
1.2.3 Sensory system.....	13
2 MATHEMATICAL MODEL	14
2.1 SETUP AND REQUIREMENTS.....	14
2.2 FORCE ANALYSIS AND SYSTEM EQUATIONS.....	14
2.2.1 System equations.....	15
2.2.2 Interpretation of terms in system equations	17
2.2.3 Matrix form of system equations	18
2.3 LINEARIZATION AND SIMPLIFICATION OF THE MODEL	18
3 LQ CONTROLLER DESIGN	20
3.1 DISCRETE MODEL STRUCTURE.....	20
3.2 CONTROLLER STRUCTURE	20
3.3 CONTROL LAW	21
3.4 CONTROLLER PARAMETERS DETERMINATION.....	22
3.4.1 Spectral factorization of a polynomial	23
3.4.2 Polynomial Toolbox for MATLAB	24
II ANALYSIS	25
4 IDENTIFICATION.....	26
5 SIMULATION.....	30
5.1 SIMULINK MODEL.....	30
5.2 CONTROLLER DESIGN	32
5.3 SIMULATION RESULTS	33
6 REAL MODEL CONTROL.....	41
6.1 SIMULINK MODEL.....	41
6.2 CONTROLLER DESIGN	44
6.3 REAL MODEL CONTROL RESULTS.....	45
6.4 NAVIGATING THE MAZE.....	49
6.4.1 Automatic path determination.....	49
6.4.2 Maze navigation results.....	51
6.4.3 Watershed transform	54
7 GRAPHICAL USER INTERFACE	55
CONCLUSION	57
BIBLIOGRAPHY	58
LIST OF FIGURES	60
APPENDICES	62

INTRODUCTION

Digital control is a widespread branch of control theory thanks to cheap and flexible microcontrollers. They can easily act as the system controller with just few lines of code, thus providing an easy way to implement various controllers. Even more so, controller parameters can be changed adaptively during run time, which extends the possibilities of its use. Also many sensors provide discrete time data, so just by implementing them in the system introduces sampling frequency which needs to be dealt with. Discrete time models are not the best way to describe real systems, but they are very simple and straightforward.

The CE 151 model from Humusoft is educational model designed to explore various tasks in continuous or discrete time, and in state-space or I/O model. It provides supporting hardware and software so that user can fully concentrate on the control problem ahead. Despite the good background support, the user has to deal with typical problems which arise before the design of controller and also after its design. User has to implement his own solutions to make the model suitable for control, ranging from normalization and signals scaling to model initialization. One could say that the controller design is only a small part of a bigger picture.

The Ball & Plate model itself is unstable system, which is challenging in every aspect of the design process. Identification and more importantly testing and debugging are steps that are negatively influenced by system's instability. The controller design is not influenced by this, if chosen correctly. LQ controller design with pole placement and polynomial method for determining controller parameters proved to be suitable as it can be computed algorithmically and the result is relatively robust regarding the unstable systems.

I. THEORY

1 CE151 BALL & PLATE APPARATUS

The CE151 Ball & Plate Apparatus is two dimensional system designed to control ball position and trajectory on the plate with two degrees of freedom. The model is unstable with second order astaticism and is suited for studying system dynamics, identification and design of various control algorithms. The plate is pivoted at its centre and can rotate around two perpendicular axes using two stepper motors as shown in Fig. 1. [1] The rotational movement of stepper motors is transformed to the plate inclination via steel wires. The ball position in Cartesian coordinate system is obtained from camera located above the plate. In this arrangement, the model has two inputs (stepper motors voltages) and two outputs (2D coordinates of the ball).

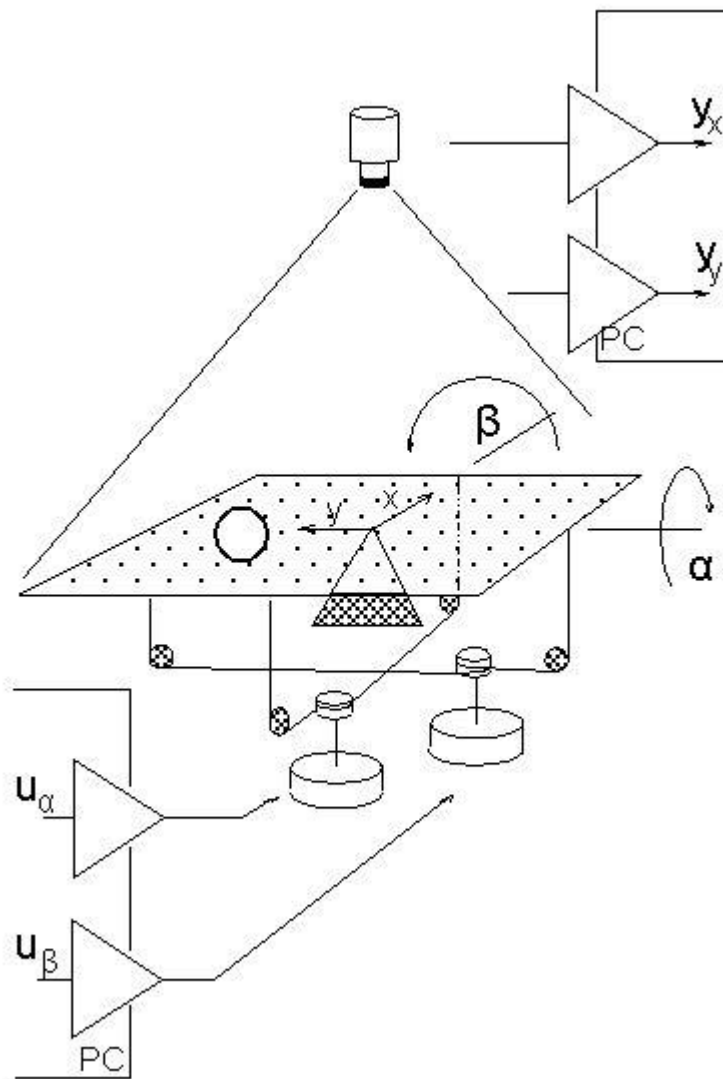


Fig. 1 CE151 Ball & Plate model diagram [1]

1.1 Technical details

Technical details of the CE151 model were obtained mainly from model manual [1].

- Main body:
 - Dimensions 430 x 430 x 200 mm
 - Plate Dimensions 400 x 400 mm
 - Length of the camera stand 1100 mm
 - Weight 9 kg
 - Plate actuation 2 stepper motors in open loop
 - Step/angle conversion 1 step $\approx 0.001^\circ$

- Power Supply:
 - DC Power Supply 32 V, 50 VA
 - Dimensions 175 x 175 x 100 mm
 - Weight 2 kg

- Multifunction I/O card
 - Type MF 624 – PCI
 - A/D converter 8 14-bit single-ended channels
 - D/A converter 4 14-bit channels
 - Digital I/O 8 TTL inputs and 8 TTL outputs
 - Additional 4 encoder inputs, 4 counters/timers

- CCD camera:
 - Type Logitech QuickCam Pro 5000
 - Video format RGB24
 - Resolution (set) 160 x 120 pixels
 - Resolution (max) 640 x 480 pixels
 - Frame rate (max) 30 fps
 - Height (from the plate) 600 mm

- Range of experiments:
 - Real time processing
 - Digital PID controller design
 - LQ/LQG controller design
 - Fuzzy controller design
 - Adaptive controller design
 - Path planning

- Software provided:
 - Interface drivers
 - Demo package using PID controller
 - Drivers for Real Time Toolbox for MATLAB/Simulink

1.2 Alternative solutions and comparison

Alternative solutions of the Ball & Plate model are proposed in this chapter, so they can be compared to show other ways to control ball position and trajectory on the plate. This chapter is not meant to explain every possible solution in detail, but merely show different solutions, compare them and make several conclusions.

1.2.1 Pivot point

The CE151 model has one pivot point in the center and motors are indirectly connected with the centers of plate edges. This solution is actually very simple and effective. Only two motors are needed, each to control one axis. The other solution would be to have 3 pivot points in the corners of the plate, with 2 of them directly connected to 2 motors. This setup is slightly different from CE151, but it still retains its 2 degrees of freedom and could be described as the 2DOF version of the Stewart platform. Such device was used in [2] and is shown in Fig. 2.

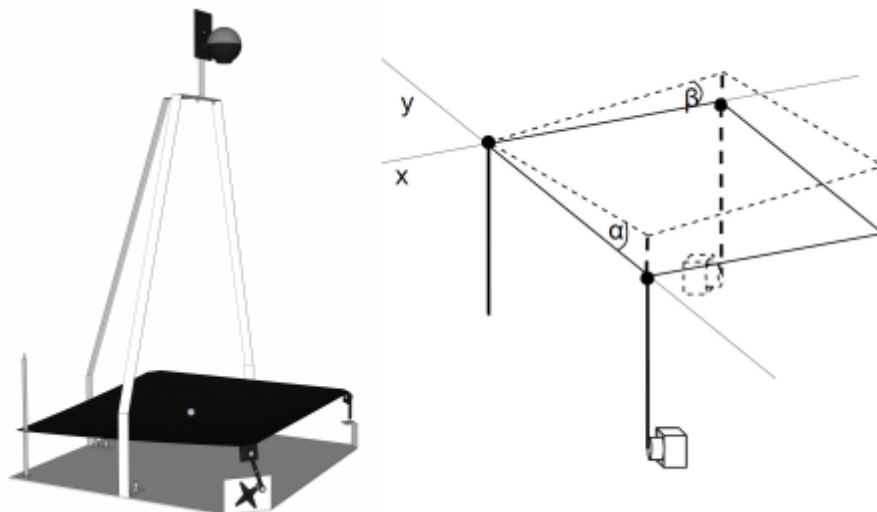


Fig. 2 Alternative Ball & Plate model [2]

1.2.2 Actuator - plate connection

The CE151 model has motors connected with plate via steel wires using pulleys as shown in Fig. 1. This solution may be simple, but wires are relatively prone to elastic deformations. This may cause strong nonlinear behavior, mostly notable in rapid inclination changes. The bigger problem is in wire flexibility, as it doesn't have any counteracting force that would compensate plate vibrations and sudden movements. Actuators should be connected to the plate using more rigid arms with joints. Such setup would be more rigid and reliable, thus

providing better control of plate's movement and inclination and reducing vibrations caused by rapid changes. Arms could be connected to the plate anywhere on the main axes, preferably closer to the center of the plate (pivot point) to reduce arm length, but not too close to make use of the leverage provided.

There is also an option to increase the number of degrees of freedom, e.g. to 6DOF (or 3DOF) in Stewart platform arrangement (Fig. 3), which would add extra control over the model in exchange for its simplicity.

Another solution (Fig. 3) is to use inner plate rotating inside outer frame with perpendicular rotation axes as shown in [3]. At least one actuator in this setup (the inner one) would have to be directly connected to the plate, so without any transmission or gearbox, it could be challenging to assume that the problem is symmetrical in the matter of actuation, although this solution seems to be quite reliable and even more aesthetic.

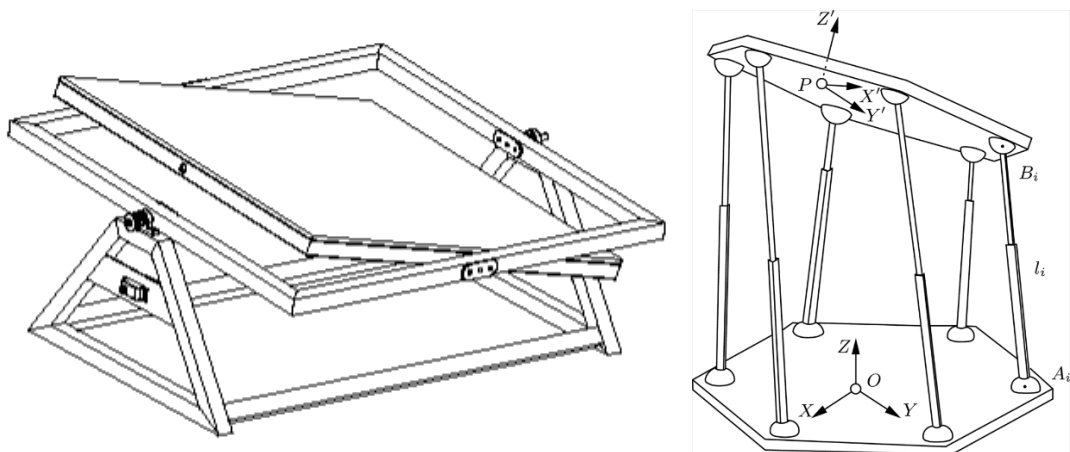


Fig. 3 Frame pivoting layout [3] and 6DOF Stewart platform [4]

1.2.3 Sensory system

The CE151 model uses RGB camera and the ball position is obtained from grayscale image. The camera is located above the plate in certain height so that it captures the whole scene. As an alternative to the camera, it is possible to use special motion detecting system that is able to detect and track pre-defined objects (e.g. Kinect[™]), but these solutions are rather expensive. The other possibility is to use resistive touch panel (or any other suitable technology) to obtain the ball's position. This method is probably better at dealing with external noise and disturbances in form of foreign object above the plate (in the camera vision field), possible reflections or light intensity changes. Touch panel would remove the need to use optical object tracking algorithm or greatly reduce its computational cost.

2 MATHEMATICAL MODEL

This chapter's aim is to derive an authentic mathematical model, which would be precise enough to cover dynamics of the real model, but simple enough to make it suitable for following implementation and controller design.

2.1 Setup and requirements

Before the modelling itself, it is essential to set requirements and presumptions taken into account. The model can be divided into a ball-plate model and a servo motor model. This tactic is possible because it is assumed that servo motors are not influenced by the motion of the plate or the ball. The ball-plate model describes the motion of the ball on the plate and how plate inclination is influenced by the ball and driving forces (Fig. 4). The following assumptions and simplifications are considered:

- There is no slip between the ball and the plate.
- The contact between the ball and the plate is not lost.
- There is no friction (e.g. from air or ball-plate contact).
- The ball is an ideal sphere or spherical shell and homogenous.
- The plate is an infinite plane and its inclination has no boundary.

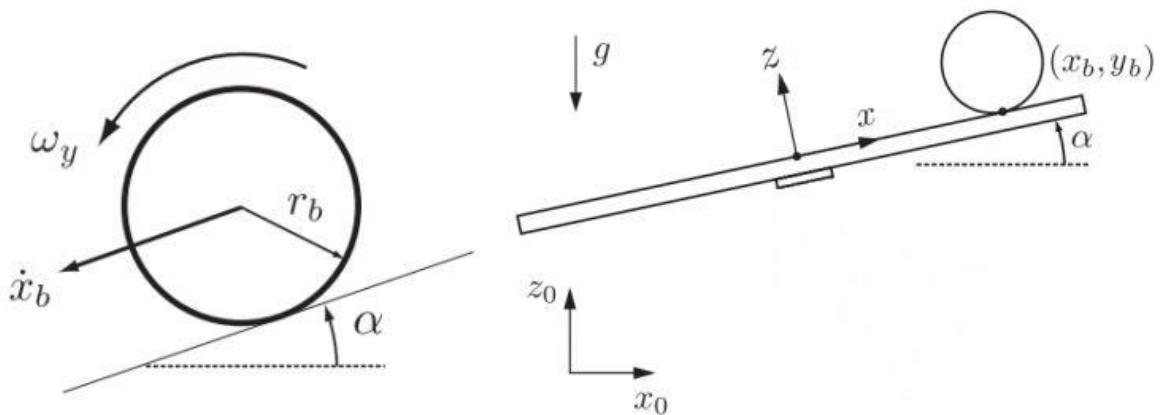


Fig. 4 Mathematical model setup [5]

2.2 Force analysis and system equations

It is necessary to clarify that there is a modelling part present in [1] and this chapter follows the same principles and methods, but there are significant mistakes which compromise the resulting model. Hence it is not advised to closely follow the modelling part described in [1]. Similar approach is taken in [5], but it also contains few negligible mistakes. Many projects

and theses regarding CE151 were found, where mistakes from its manual pages were repeated (more likely just copied) without further concern.

2.2.1 System equations

To obtain dynamical equations of the model, the general form of Euler-Lagrange equation is used:

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i \quad (2.1)$$

Where T is overall kinetic energy, V is overall potential energy, Q_i is i -th generalized force and q_i is i -th generalized coordinate. The generalized coordinates are two ball position coordinates $[x, y]$ and two plate inclinations $[\alpha = \theta_x, \beta = \theta_y]$, where $[x, y] = [0, 0]$ is center of the plate.

The kinetic energy of the system consists of the kinetic energy of the ball and the kinetic energy of the plate.

$$T = T_{ball} + T_{plate} \quad (2.2)$$

The kinetic energy of the ball consists of translational and rotational parts:

$$T_{ball} = T_{trans} + T_{rot} \quad (2.3)$$

The translational energy of the ball is:

$$T_{trans} = \frac{1}{2} m v^2 = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2) \quad (2.4)$$

The rotational energy of the ball:

$$T_{rot} = \frac{1}{2} I_b \omega^2 = \frac{1}{2} I_b \frac{v^2}{r^2} = \frac{1}{2} \frac{I_b}{r^2} (\dot{x}^2 + \dot{y}^2) \quad (2.5)$$

Where v is the velocity of the ball (not just its speed), I_b is the moment of inertia of the ball, r is the radius of the ball, ω is the angular velocity of the ball and m is the mass of the ball.

By substituting equations (2.4) and (2.5) into equation (2.3):

$$T_{ball} = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} \frac{I_b}{r^2} (\dot{x}^2 + \dot{y}^2) = \frac{1}{2} \left(m + \frac{I_b}{r^2} \right) (\dot{x}^2 + \dot{y}^2) \quad (2.6)$$

The kinetic energy of the plate (with its moment of inertia I_p) can be expressed:

$$T_{plate} = \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m(\dot{\alpha}x + \dot{\beta}y)^2 \quad (2.7)$$

After substitution of equations (2.6) and (2.7) into equation (2.2):

$$T = \frac{1}{2}\left(m + \frac{I_b}{r^2}\right)(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m(\dot{\alpha}x + \dot{\beta}y)^2 \quad (2.8)$$

The potential energy of the ball with h being ball height relative to the plate center:

$$V = mgh = mg(x \sin \alpha + y \sin \beta) \quad (2.9)$$

Individual parts of Euler-Lagrange equation:

$$\frac{\partial T}{\partial \dot{x}} = \left(m + \frac{I_b}{r^2}\right)\dot{x}, \quad \frac{\partial T}{\partial \dot{y}} = \left(m + \frac{I_b}{r^2}\right)\dot{y} \quad (2.10)$$

$$\frac{\partial T}{\partial \dot{\alpha}} = m(\dot{\alpha}x^2 + \dot{\beta}xy) + (I_p + I_b)\dot{\alpha}, \quad \frac{\partial T}{\partial \dot{\beta}} = m(\dot{\alpha}xy + \dot{\beta}y^2) + (I_p + I_b)\dot{\beta} \quad (2.11)$$

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{x}} = \left(m + \frac{I_b}{r^2}\right)\ddot{x}, \quad \frac{d}{dt} \frac{\partial T}{\partial \dot{y}} = \left(m + \frac{I_b}{r^2}\right)\ddot{y} \quad (2.12)$$

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{\alpha}} = (I_p + I_b + mx^2)\ddot{\alpha} + m(\ddot{\beta}xy + \dot{\beta}(\dot{x}y + x\dot{y}) + 2\dot{\alpha}\dot{x}) \quad (2.13)$$

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{\beta}} = (I_p + I_b + my^2)\ddot{\beta} + m(\ddot{\alpha}xy + \dot{\alpha}(\dot{x}y + x\dot{y}) + 2\dot{\beta}\dot{y}) \quad (2.14)$$

$$\frac{\partial T}{\partial x} = m(\dot{\alpha}\dot{\beta}y + \dot{\alpha}^2x), \quad \frac{\partial T}{\partial y} = m(\dot{\alpha}\dot{\beta}x + \dot{\beta}^2y), \quad \frac{\partial T}{\partial \alpha} = 0, \quad \frac{\partial T}{\partial \beta} = 0 \quad (2.15)$$

$$\frac{\partial V}{\partial x} = mg \sin \alpha, \quad \frac{\partial V}{\partial y} = mg \sin \beta, \quad \frac{\partial V}{\partial \alpha} = mgx \cos \alpha, \quad \frac{\partial V}{\partial \beta} = mgy \cos \beta \quad (2.16)$$

It is important to note there are no external forces (except gravity) acting on the ball itself ($Q_x = 0$ and $Q_y = 0$) and there are forces in the form of torque acting on the plate and changing its inclination ($Q_\alpha = \tau_\alpha$ and $Q_\beta = \tau_\beta$).

After substitution of equations (2.12) - (2.16) into (2.1), the nonlinear system equations are:

$$x: \left(m + \frac{I_b}{r^2}\right)\ddot{x} - m(\dot{\alpha}\dot{\beta}y + \dot{\alpha}^2x) + mg \sin \alpha = 0 \quad (2.17)$$

$$y: \left(m + \frac{I_b}{r^2}\right)\ddot{y} - m(\dot{\alpha}\dot{\beta}x + \dot{\beta}^2y) + mg \sin \beta = 0 \quad (2.18)$$

$$\alpha: (I_p + I_b + mx^2)\ddot{\alpha} + m(\ddot{\beta}xy + \dot{\beta}(\dot{x}y + x\dot{y}) + 2\dot{\alpha}\dot{x}x) + mgx \cos \alpha = \tau_\alpha \quad (2.19)$$

$$\beta: (I_p + I_b + my^2)\ddot{\beta} + m(\ddot{\alpha}xy + \dot{\alpha}(\dot{x}y + x\dot{y}) + 2\dot{\beta}\dot{y}y) + mgy \cos \beta = \tau_\beta \quad (2.20)$$

2.2.2 Interpretation of terms in system equations

It can be seen that equations (2.17) and (2.18) describe the ball motion and how the acceleration of the ball depends on its position on the plate and on angles and angular velocities of the plate. Equations (2.19) and (2.20) show plate dynamics and how it depends on external torques, ball position, velocity, angular velocity and acceleration of the plate. It is necessary to note that dynamics of the stepper motors are neglected, thus the system equations describe only the ball and plate problem. These dynamics will be added later.

- m Mass of the ball - [kg].
- r Radius of the ball - [m].
- I_b Moment of inertia of the ball - [kgm²].
- I_p Moment of inertia of the plate - [kgm²].
- x, y Coordinates of the ball from center of the plate - [m].
- \dot{x}, \dot{y} First time derivatives of coordinates - [ms⁻¹].
- \ddot{x}, \ddot{y} Second time derivatives of coordinates - [ms⁻²].
- α, β Plate angles [θ_x, θ_y] respective to coordinates - [rad].
- $\dot{\alpha}, \dot{\beta}$ First time derivatives of plate angles - [rads⁻¹].
- $\ddot{\alpha}, \ddot{\beta}$ Second time derivatives of plate angles - [rads⁻²].
- τ_α, τ_β Torques acting on the plate - [Nm].
- $m(\dot{\alpha}\dot{\beta}y + \dot{\alpha}^2x)$ Centrifugal force - [N].
- $(I_p + I_b + mx^2)\ddot{\alpha}$ Torque as a result of combined inertia - [Nm].
- $m(\ddot{\beta}xy + \dot{\beta}(\dot{x}y + \dot{\beta}x\dot{y}))$ Gyroscopic influence - [Nm].
- $2m\dot{\alpha}\dot{x}x$ Coriolis influence - [Nm].
- $mgx \cos \alpha$ Gravitational influence - [Nm].

2.2.3 Matrix form of system equations

System equations can be generally written in the following tensor form:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Q \quad (2.21)$$

Where $M(q)$ stands for inertia matrix, $C(q, \dot{q})$ is Coriolis matrix (matrix of Coriolis and centrifugal forces) and $G(q)$ is gravity matrix. These terms are in equations (2.22) to (2.25).

$$q = \begin{bmatrix} x \\ y \\ \alpha \\ \beta \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix}, \quad \ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}, \quad Q = \begin{bmatrix} 0 \\ 0 \\ \tau_\alpha \\ \tau_\beta \end{bmatrix} \quad (2.22)$$

$$M(q) = \begin{bmatrix} \left(m + \frac{I_b}{r^2}\right) & 0 & 0 & 0 \\ 0 & \left(m + \frac{I_b}{r^2}\right) & 0 & 0 \\ 0 & 0 & (I_p + I_b + mx^2) & mxy \\ 0 & 0 & mxy & (I_p + I_b + my^2) \end{bmatrix} \quad (2.23)$$

$$C(q, \dot{q}) = m \begin{bmatrix} 0 & 0 & -\dot{\alpha}x & -\dot{\alpha}y \\ 0 & 0 & -\dot{\beta}x & -\dot{\beta}y \\ 2\dot{\alpha}x & 0 & 0 & (x\dot{y} + y\dot{x}) \\ 0 & 2\dot{\beta}y & (x\dot{y} + y\dot{x}) & 0 \end{bmatrix} \quad (2.24)$$

$$G(q) = \begin{bmatrix} mg \sin \alpha \\ mg \sin \beta \\ mgx \cos \alpha \\ mgy \cos \beta \end{bmatrix} \quad (2.25)$$

2.3 Linearization and simplification of the model

To simplify the model, it is assumed that stepper motors don't lose any step and load doesn't affect their performance, thus angles α and β can be direct system inputs. That's why equations (2.19) and (2.20) can be omitted in behalf of simplification process.

Because the ball is assumed to be homogenous sphere or hollow sphere, its moment of inertia can be approximated by the moment of inertia of sphere or spherical shell as in eq. (2.26).

$$I_{sphere} = \frac{2}{5}mr^2; \quad I_{shell} = \frac{2}{3}mr^2 \quad (2.26)$$

To linearize given equations around steady state where angles α and β are zero, it is assumed they change in range $(-5^\circ; 5^\circ)$ or in other words, $|\alpha| \ll 1$ and $|\beta| \ll 1$ in radians. Because of that, sine functions can be replaced by their arguments. In the same manner, it is assumed that the rate of change of angles is close to zero. As the equations contain squares of angular velocities or products of their combination, these terms can be assumed to be zero.

To sum up linearization assumptions:

- When $|\alpha| \ll 1$; $|\beta| \ll 1 \Rightarrow \sin \alpha \approx \alpha$; $\sin \beta \approx \beta$.
- When $|\dot{\alpha}| \ll 1$; $|\dot{\beta}| \ll 1 \Rightarrow \dot{\alpha}\dot{\beta} \cong 0$; $\dot{\alpha}^2 \cong 0$; $\dot{\beta}^2 \cong 0$.

System equations after simplification and linearization have the following structure:

$$x: \quad \ddot{x} = -Kg\alpha \quad (2.27)$$

$$y: \quad \ddot{y} = -Kg\beta \quad (2.28)$$

Where K is constant dependent only on the type of ball, whether it is hollow sphere or not:

$$K_{sphere} = \frac{m}{\left(m + \frac{I_{sphere}}{r_b^2}\right)} = \frac{m}{\left(m + \frac{2}{5}mr_b^2\right)} = \frac{5}{7}; \quad K_{shell} = \frac{3}{5} \quad (2.29)$$

It is now easy to obtain transfer functions from linearized model:

$$x: \quad G_{x/\alpha}(s) = -\frac{Kg}{s^2} \quad (2.30)$$

$$y: \quad G_{y/\beta}(s) = -\frac{Kg}{s^2} \quad (2.31)$$

It is obvious either from system equations, their matrix form or linearization that this problem is symmetric. Thus it is possible to analyze system and design controller for only one ball coordinate and plate angle.

Instead of modelling the motor separately, a simple first order transfer function G_m is assumed, which is very simple and reliable approximation of its inner workings and drivers.

$$G_m(s) = \frac{K_m}{\tau_m s + 1} \quad (2.32)$$

3 LQ CONTROLLER DESIGN

The main purpose of this chapter is not to introduce various designs of controllers but to successfully design a digital controller, which would be suitable for the model. The chosen controller algorithm and design is thus briefly introduced in this chapter without going into much theory behind linear quadratic (LQ) control.

3.1 Discrete model structure

To discretize the model, it is needed to choose a sampling period, but as this is only the theoretical part of the design, a general discrete transfer function will be assumed.

The transfer function combined from the ball and plate model (2.30) and motor model (2.32) has the following general structure:

$$G(s) = \frac{K}{Ts^3 + s^2} = \frac{K}{s^2(Ts + 1)} \quad (3.1)$$

Thus it is assumed that resulting discrete transfer function has this structure:

$$G(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_1z^{-1} + b_2z^{-2} + b_3z^{-3}}{1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3}} \quad (3.2)$$

3.2 Controller structure

It is possible to use many controller structures ranging from simple 1DOF structure, various cascade structures, ICM (Internal Model Control) to controllers with fuzzy supervision. The controller structure proposed here is two degree of freedom (2DOF) closed-loop controller structure shown in Fig. 5, which provides separation of feedback part (responsible for stabilization and disturbance rejection) and feedforward part (responsible for reference tracking) [6]. This should provide better control over the model and its behavior.

Two degree of freedom closed-loop control system is shown in Fig. 5, where $G(z^{-1})$ is the controlled plant, $C_b(z^{-1})$ is the feed-back part of the controller, $C_f(z^{-1})$ is the feed-forward part of the controller, $\frac{1}{K(z^{-1})} = \frac{1}{1-z^{-1}}$ is the summation part, $w(k)$ is reference signal, $n(k)$ is load disturbance and $v(k)$ is disturbance signal. For the sake of simplification in the following chapters, there will be assumed no disturbances acting on the system.

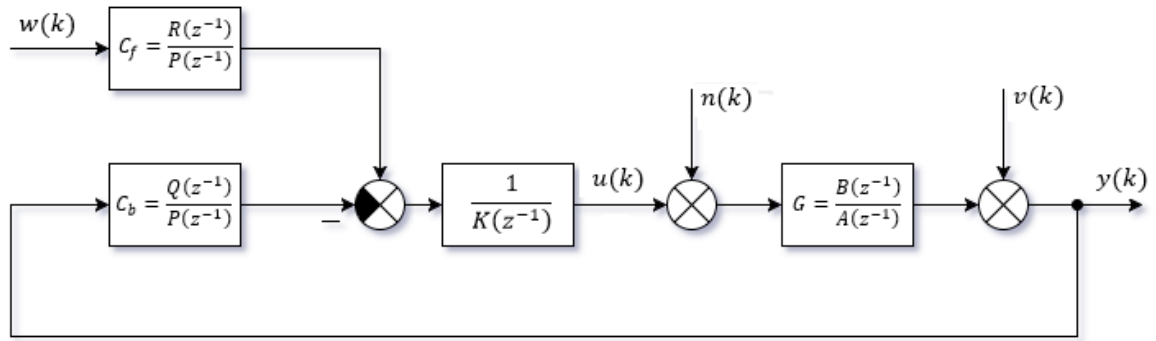


Fig. 5 Structure of 2DOF controller

3.3 Control law

By taking signals in Fig. 5 in their discrete polynomial forms and omitting disturbances $n(k)$ and $v(k)$, it is possible to write equations describing the plant and the controller [7]:

$$Y(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} U(z^{-1}) \quad (3.3)$$

$$U(z^{-1}) = \frac{R(z^{-1})}{P(z^{-1})K(z^{-1})} W(z^{-1}) - \frac{Q(z^{-1})}{P(z^{-1})K(z^{-1})} Y(z^{-1}) \quad (3.4)$$

By substituting equation (3.4) into (3.3) and vice versa:

$$Y(z^{-1}) = \frac{B(z^{-1})R(z^{-1})}{A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1})} W(z^{-1}) \quad (3.5)$$

$$U(z^{-1}) = \frac{A(z^{-1})R(z^{-1})}{A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1})} W(z^{-1}) \quad (3.6)$$

The characteristic polynomial $D(z^{-1})$ can be selected from equation (3.5):

$$A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}) = D(z^{-1}) \quad (3.7)$$

The chosen degree of characteristic polynomial is $\partial D(z^{-1}) = 6$ and plant's degrees are $\partial A(z^{-1}) = 3$ and $\partial B(z^{-1}) = 3$ as shown in (3.2). Hence according to [7] for a step-changing reference signal it is possible to obtain polynomial degrees considered in z^{-1} :

$$\partial Q = \partial A + \partial K - 1 = 3; \quad \partial P = \partial D - \partial A - \partial K = 2; \quad \partial R = 0 \quad (3.8)$$

Note that it is possible to write feed-back and feed-forward parts as $\frac{Q}{PK}$ and $\frac{R}{PK}$ respectively. It is used separately due to the implementation effectiveness, where it is useless for the feed-forward part to sum the reference value or the feed-back part to sum the output value.

In addition to the plant's structure in (3.2) and by making use of (3.8), the digital controllers can be expressed in following discrete transfer forms:

$$C_b(z^{-1}) = \frac{Q(z^{-1})}{P(z^{-1})} = \frac{q_0 + q_1z^{-1} + q_2z^{-2} + q_3z^{-3}}{1 + p_1z^{-1} + p_2z^{-2}} \quad (3.9)$$

$$C_f(z^{-1}) = \frac{R(z^{-1})}{P(z^{-1})} = \frac{r_0}{1 + p_1z^{-1} + p_2z^{-2}} \quad (3.10)$$

The resulting 2DOF controller output $u(k)$ is then given by:

$$u(k) = r_0w(k) - q_0y(k) - q_1y(k-1) - q_2y(k-2) - q_3y(k-3) + \\ +(1 - p_1)u(k-1) + (p_1 - p_2)u(k-2) + p_2u(k-3) \quad (3.11)$$

3.4 Controller parameters determination

To determine unknown parameters of the controller it is essential to get coefficients of the characteristic polynomial $D(z^{-1})$. As mentioned in previous chapter, the chosen degree of characteristic polynomial in negative powers of z is 6:

$$D_6(z^{-1}) = 1 + d_1z^{-1} + d_2z^{-2} + d_3z^{-3} + d_4z^{-4} + d_5z^{-5} + d_6z^{-6} \quad (3.12)$$

To determine its coefficients it is possible to place 6 poles on the z -plane which would compose this polynomial. Placing 6 roots on the z -plane is quite challenging and hardly leads to optimal solution. To get an optimal solution, it is possible to minimize quadratic criterion, which with use of spectral factorization ultimately provides half of the roots as an optimal solution. The quadratic criterion with controller output penalization is presented in [7]:

$$J = \sum_{k=0}^{\infty} \{[e(k)]^2 + q_u[u(k)]^2\} \quad (3.13)$$

The constant q_u is a penalization constant, $e(k) = w(k) - y(k)$ is the error and $u(k)$ is the controller output. Standard minimization of this criterion is done in state-space description and leads to the solution of algebraic Riccati equation. There is yet another solution which minimizes the quadratic criterion using spectral factorization and plant model expressed in transfer function as an I/O model. The criterion (3.13) is minimal for the (3.7), where $D(z^{-1})$ is the result of spectral factorization of the following equation. [7]

$$A(z^{-1})q_uA(z) + B(z^{-1})B(z) = D(z^{-1})\delta D(z) \quad (3.14)$$

Where δ is chosen so that coefficient $d_0 = 1$ and $A(z), B(z), D(z)$ are conjugate polynomials.

Because it is assumed that degrees of polynomials in (3.2) are maximum 3, the equation (3.14) has also polynomial of 3rd degree on both its sides. Thus only three coefficients of $D_6(z^{-1})$ can be obtained from this spectral factorization. Other three roots have to be placed accordingly to the preference.

When coefficients of the polynomial $D_6(z^{-1})$ are determined and degree of controller polynomials is known, the Diophantine equation (3.7) can be solved. The equation (3.7) with all polynomials expressed in their full form is long, hence only its matrix form will be provided. This form is obtained from comparison of coefficients of z^{-i} from either side of the equation, which represents linear system of six equations with six variables:

$$\begin{bmatrix} b_3 & 0 & 0 & 0 & -a_3 & 0 \\ b_2 & b_3 & 0 & 0 & a_3 - a_2 & -a_3 \\ b_1 & b_2 & b_3 & 0 & a_2 - a_1 & a_3 - a_2 \\ 0 & b_1 & b_2 & b_3 & a_1 - 1 & a_2 - a_1 \\ 0 & 0 & b_1 & b_2 & 1 & a_1 - 1 \\ 0 & 0 & 0 & b_1 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_3 \\ q_2 \\ q_1 \\ q_0 \\ p_2 \\ p_1 \end{bmatrix} = \begin{bmatrix} d_6 \\ d_5 \\ d_4 + a_3 \\ d_3 - a_3 + a_2 \\ d_2 - a_2 + a_1 \\ d_1 - a_1 + 1 \end{bmatrix} \quad (3.15)$$

If the reference value is assumed to be step-changing, it is possible to write [7]:

$$r_0 = \frac{1 + d_1 + d_2 + d_3 + d_4 + d_5 + d_6}{b_1 + b_2 + b_3} = q_0 + q_1 + q_2 + q_3 \quad (3.16)$$

This equation of the form $Ax = b$ can be easily solved in MATLAB or any other computing software capable of doing matrix operations. The result is vector of unknown controller parameters that can be inserted for example into equations (3.9), (3.10) and (3.11).

3.4.1 Spectral factorization of a polynomial

Spectral factorization changes the unstable part of the polynomial to reciprocal (stable). Polynomials of the 1st or 2nd order can be used in analytical solution of the spectral factorization. The spectral factorization of higher order polynomials is done iteratively, e.g. by using special function of Polynomial Toolbox in MATLAB. The spectral factorization example will be shown for 1st order polynomial, similar as in [7].

Having the 1st order polynomial $M(z^{-1})$ and its conjugate part $M(z)$ it is possible to solve the following equation to find its factorized polynomial $D(z^{-1})$. Conjugate polynomial has negative powers of variables replaced by their positive power counterparts as seen in (3.18).

$$M(z^{-1})M(z) = D(z^{-1})\delta D(z) \quad (3.17)$$

$$(m_0 + m_1 z^{-1})(m_0 + m_1 z) = (1 + d_1 z^{-1})\delta(1 + d_1 z) \quad (3.18)$$

The equation (3.15) can be rewritten into the following form:

$$(m_0^2 + m_1^2) + m_0 m_1 (z + z^{-1}) = \delta(1 + d_1^2) + \delta d_1 (z + z^{-1}) \quad (3.19)$$

The solution $\{\delta, d_1\}$ can be obtained by comparing the coefficients of $(z^i + z^{-i})$ terms.

3.4.2 Polynomial Toolbox for MATLAB

The Polynomial Toolbox is a package for systems, signals and control analysis and design based on advanced polynomial methods. It consists of as many as 222 M-files in MATLAB code and is easy to use. [8]

The Polynomial Toolbox consists of various tools and functions [8]:

- Polynomial Matrix Operations:
 - Easy manipulation with polynomials and polynomial matrices.
 - Pre-defined variables such as s or z .
 - Polynomial matrix Editor for larger matrices.
- Advanced, fast and reliable algorithms:
 - Linear matrix polynomial equation solvers based on Sylvester matrices.
 - Spectral factorization algorithms.
 - Diophantine equations solver, Riccati equation solver.
 - Many more.
- Polynomial Matrix Fractions support.
- Polynomial Equations support.
- Classic and graphical analysis tools:
 - Robustness, stability margins.
 - Parametric and polytopic uncertainties.
 - Interval polynomials.
- Built-in Design routines:
 - LQG design, deadbeat control, pole placement, H-infinity, ...
- Links to other packages:
 - Support and conversion of LTI objects, symbolic formats, descriptors.
 - Conversion of toolbox's formats to MATLAB formats and vice versa.
 - Formatting of polynomial matrix for use in a LaTeX document.

The Polynomial Toolbox provides vast range of tools and functions, but the most interesting for this thesis are functions for solving Diophantine equations $axbyc()$ and spectral factorization $spf()$ [9].

ANALYSIS

4 IDENTIFICATION

The model was identified simply by positioning the ball to the center of the plate manually and step-changing the plate's inclination. The step response of the system was thus obtained and properly identified. In early experiments, only the side walls were used to guide the ball on the straight path. However these walls were removed later and the ball was identified without any guide. Small variations in the perpendicular direction were compensated by taking multiple measurements and nevertheless they were taken as the part of the model to make the identification more precise for the given model. In other theses found dealing with identification of CE151 model a rail was used to guide the ball. This is rather ineffective because the ball loses a direct contact with the plate which could lead to wrong identification, although this effect might be negligible.

The identification was made for plate inclination inputs 20%, 40%, 60% and 80%. For every input approximately 20 measurements were done. And for each input these measurements were repeated twice and in both directions. Measurements were done in one session for each input. In one session, 20 measurements for one input could take up to 400 seconds to measure. These data were processed accordingly to the input and only step-changing parts were chosen and the averaged result was identified. Step responses for plate inclination of 40% measured in one session are shown in Fig. 6 and their average in Fig. 7.

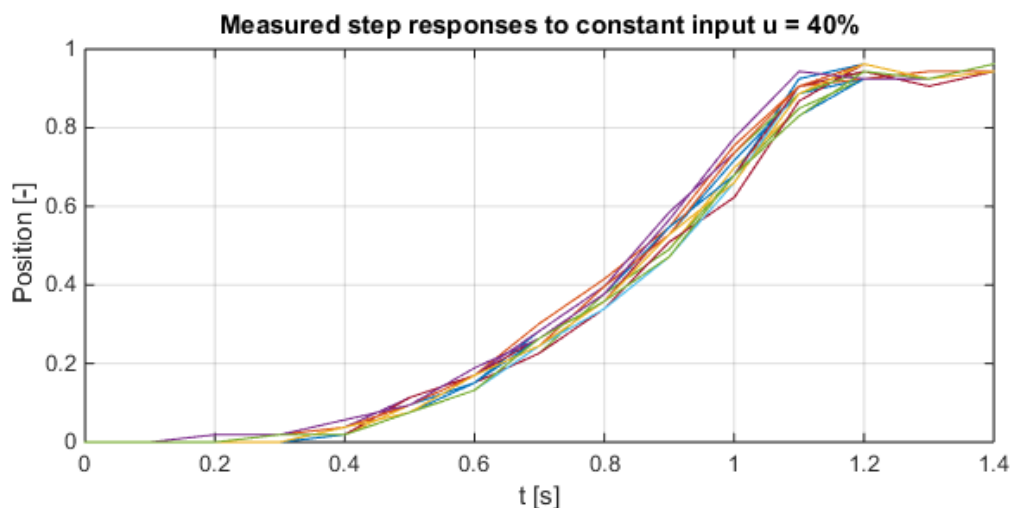


Fig. 6 Measured step responses

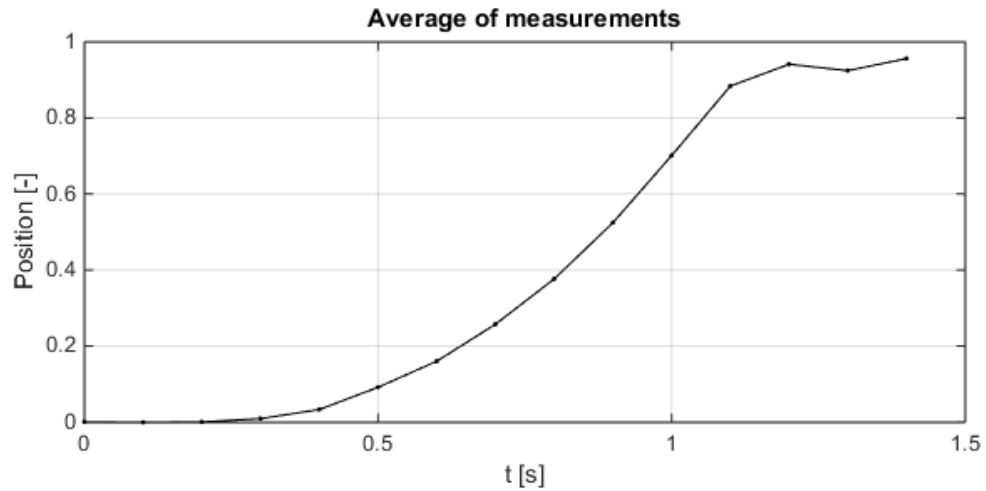


Fig. 7 Average of measured step responses

As it can be seen, the ball hit the edge of the plate and bounced back, so the response had to be identified only up until that point. The identification was done using *fminsearch* function in MATLAB. The reason to choose this was simply because it is simple and very flexible to use. It is not extraordinary fast, but as it is not used in adaptive scheme, that should not be a problem. Other identification methods were used, like Least Squares and Recursive Least Squares methods, as described in [10] and [11]. These methods were unfortunately inefficient because the general discrete transfer function structure is less descriptive than its continuous counterpart. The result of identification using the Recursive Least Squares method is shown in Fig. 8, from which can be seen that this method would require some tuning and modification to satisfy the demands. As the using different identification methods is not the main part of this thesis, the simplest solution in the form of MATLAB's *fminsearch* was used.

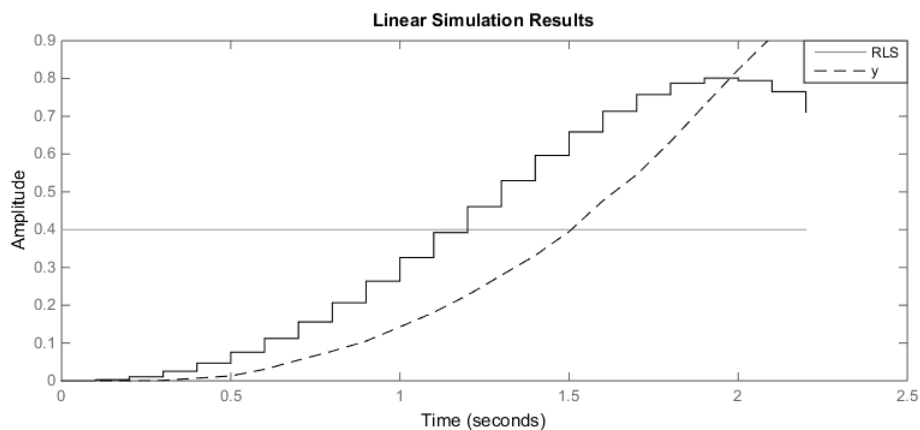


Fig. 8 Identification using Recursive Least Squares method

The overall process of identification is implemented in MATLAB in function *ident_bp.m* created for this purpose. The user is prompted to choose one of the *ident*.mat* files that contain measured sessions. Then the user is prompted to choose the bounce point (Fig. 9), as it is hard to algorithmically determine its position. The output of the function is the vector containing identified parameters and the transfer function structure in the form of anonymous function. Resulting identification process is shown in Fig. 10.

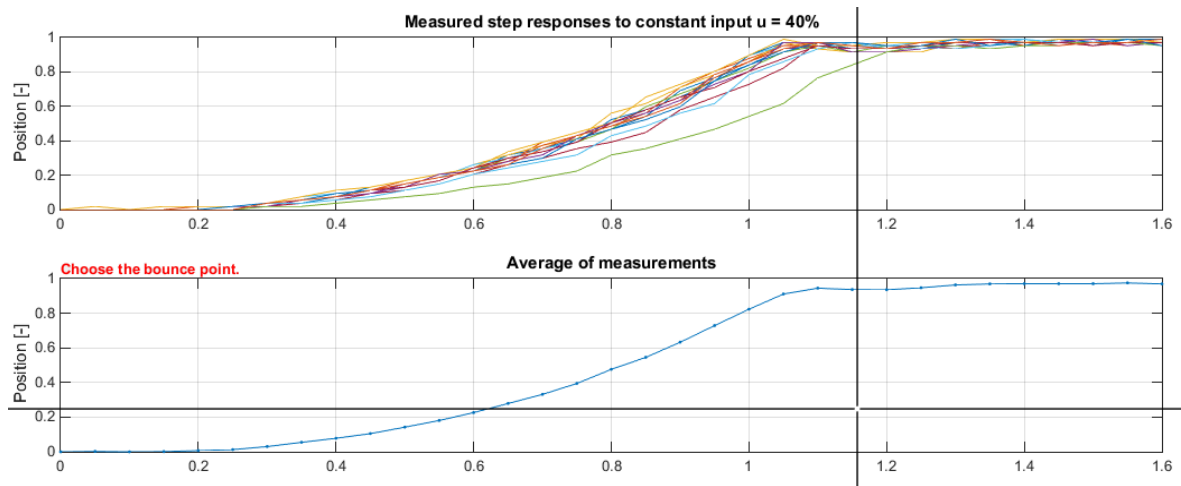


Fig. 9 Choosing the bounce point from the averaged response

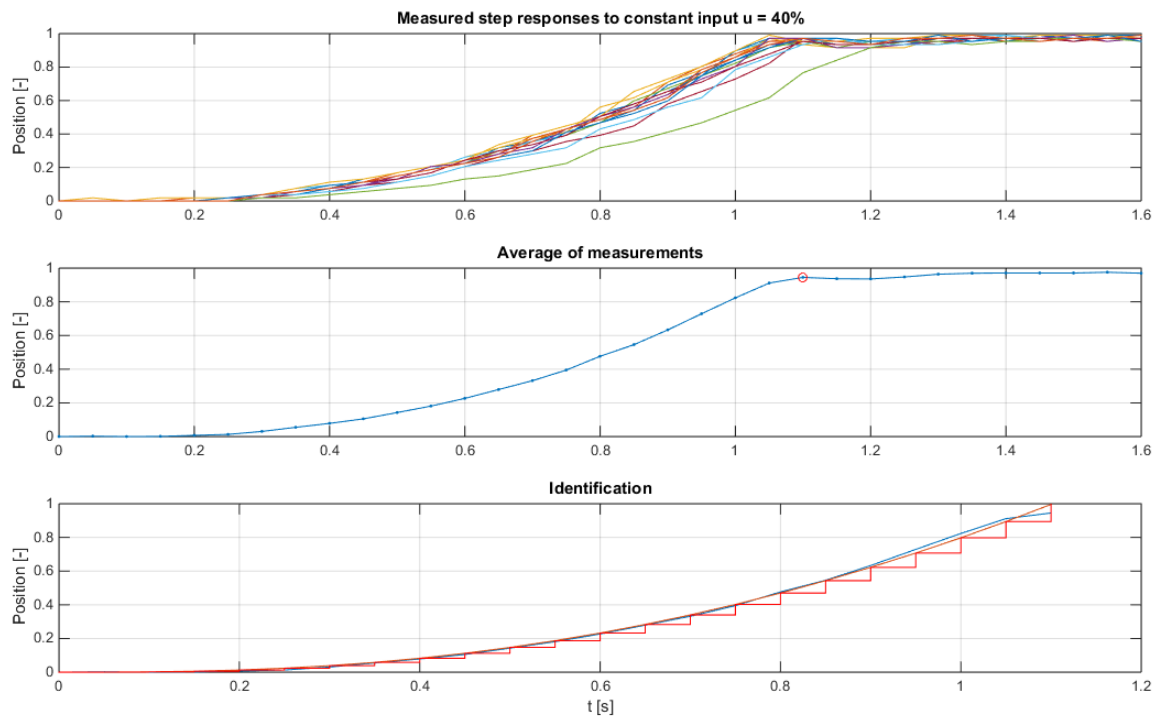


Fig. 10 Identification process

Identified parameters (3.1) from all measured step responses were gathered into one scatter plot (Fig. 11) to show their distribution and dependence on the ball type (table tennis ball and ball from mechanical computer mouse). The raising character of parameters is caused by the range of step inputs used (20-80%). Data are also normalized to better show their clustering instead of actual values.

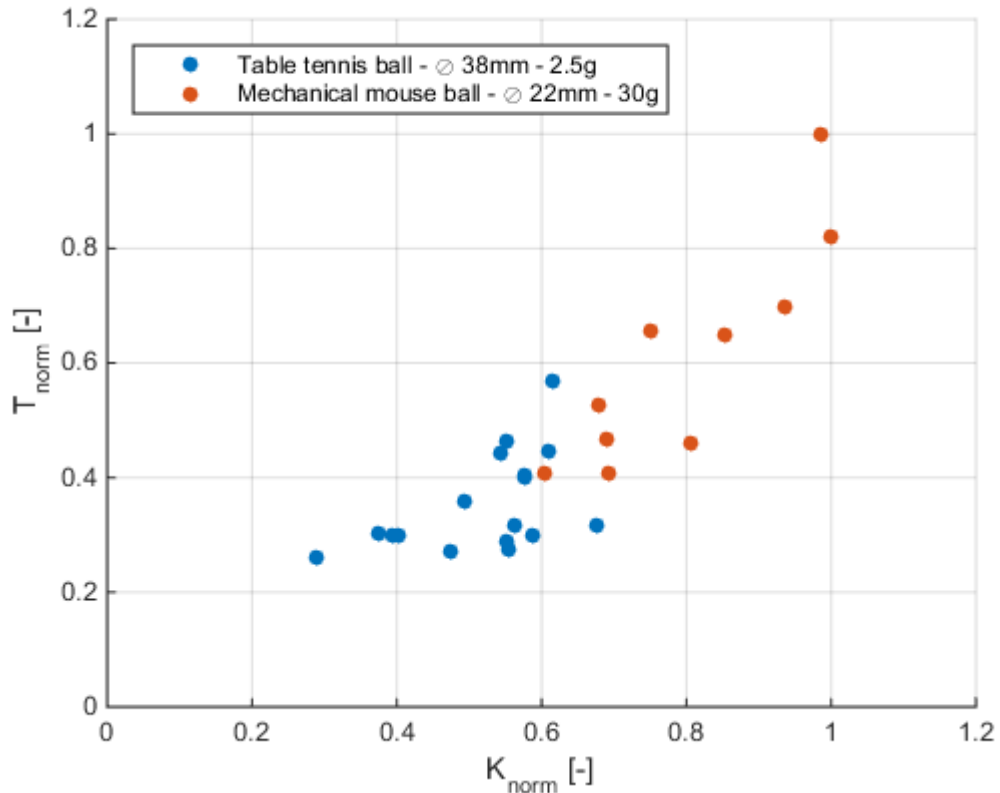


Fig. 11 Distribution of identified parameters

Note that the model is linearized around the small inclination angles, thus preferring parameters for small inputs would better describe the model. For the controller design, inputs for 40% inclination were chosen as the best option.

5 SIMULATION

5.1 Simulink model

The model according to equations (2.17) and (2.18) was constructed in the Simulink block diagram environment (Fig. 12). The motor model was simply replaced by ideal first order transfer function mentioned in [1], where it is derived from dynamics of the servo system used in the form of (2.32) as $G_m(s) = \frac{0.1878}{0.187s+1}$.

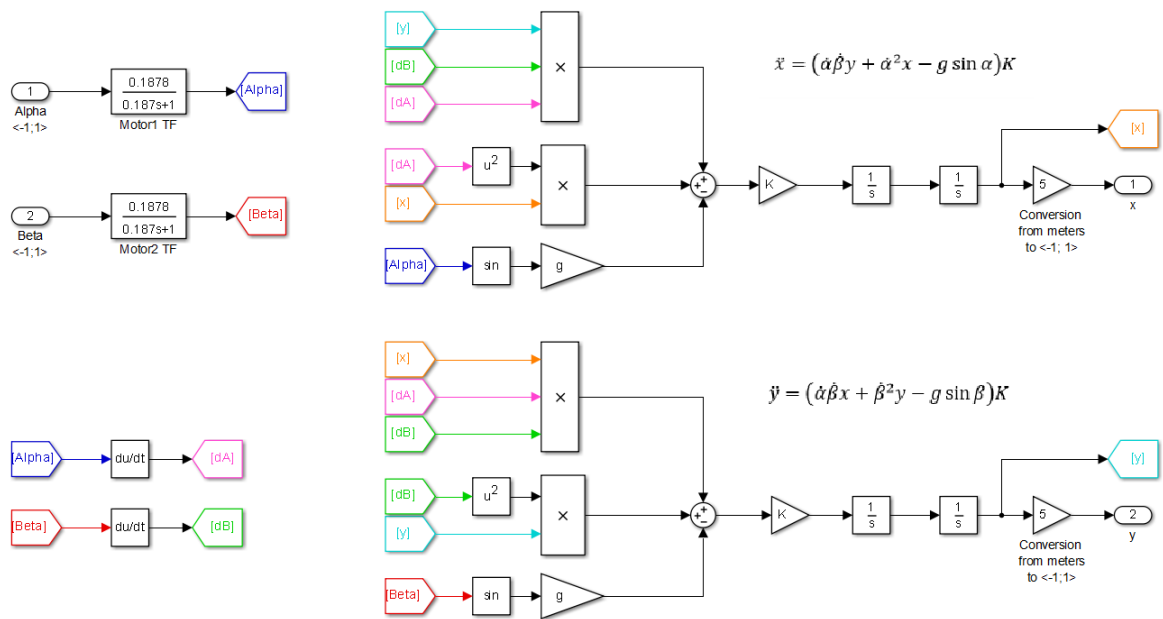


Fig. 12 Nonlinear Simulink model

To make the Simulink model more readable, feedback connections of state variables were replaced using GOTO and FROM blocks. The conversion from meters to normalized units is done using gain of 5, because the maximum position on the plate is 0.2 in meters (5 is the reciprocal of 0.2). This model is in masked subsystem with two inputs and two outputs as shown in Fig. 13 and its simple step responses are shown in Fig. 14.

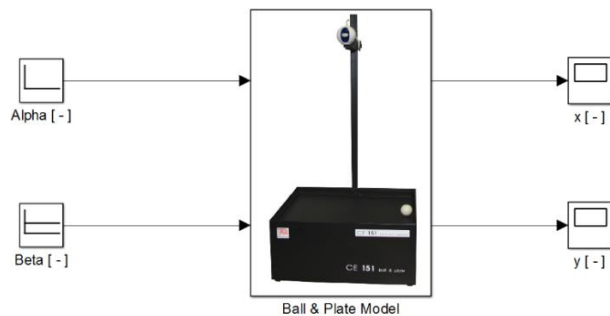


Fig. 13 Masked subsystem model

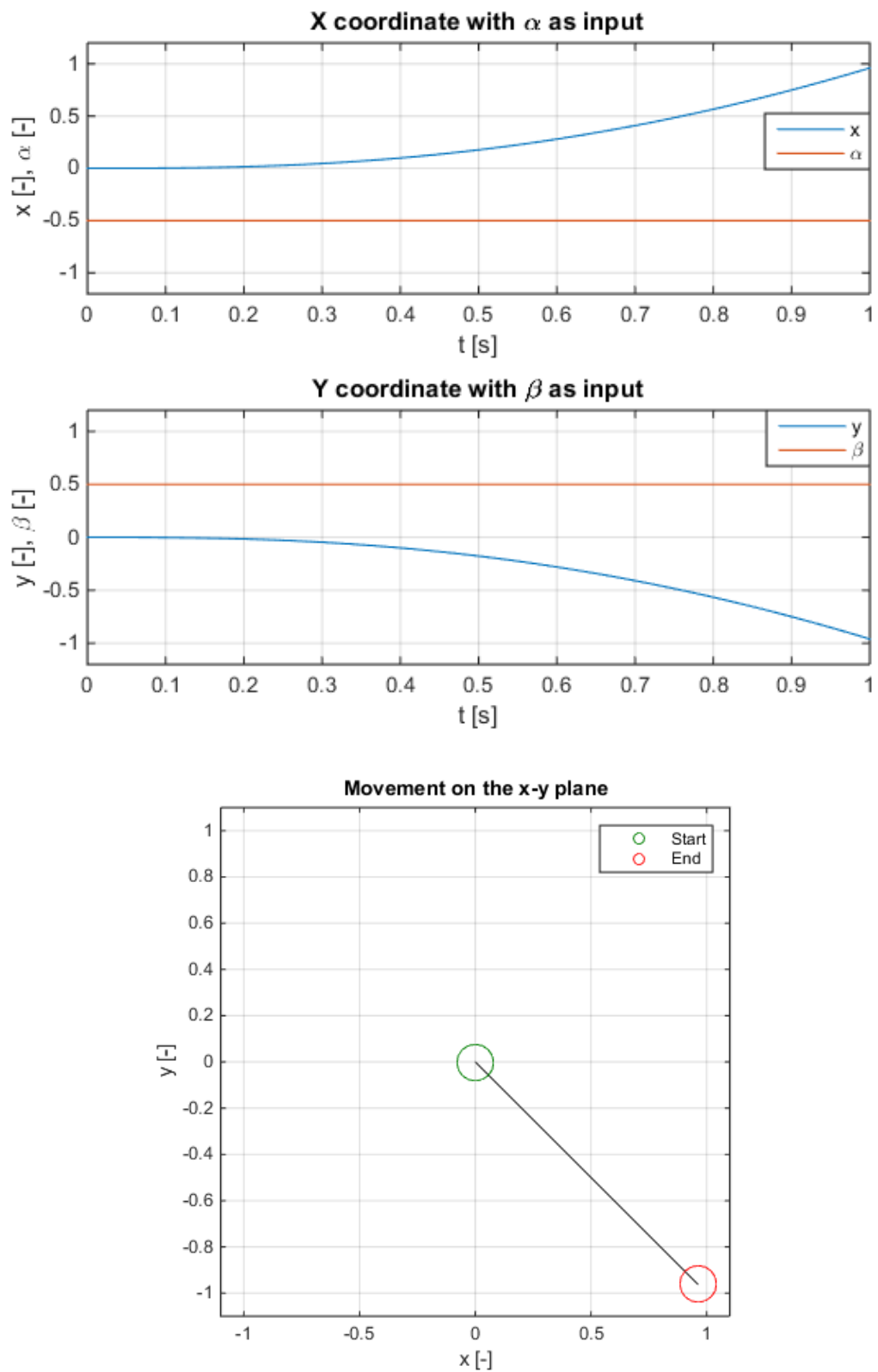


Fig. 14 Simulated response to step

5.2 Controller design

The 2DOF controller will be designed for the following plant model:

$$G(s) = \frac{-Kg}{s^2} \frac{K_m}{\tau_m s + 1} C_x = \frac{-KgK_m C_x}{s^2(\tau_m s + 1)} = \frac{-5.0706}{s^2(0.187s + 1)} \quad (5.1)$$

Where the first term is from equation (2.30), second term is the motor transfer function (2.32) and C_x is the conversion coefficient for normalization of position of the ball. This coefficient is inverted value of half of the plate's side length $C_x = \frac{1}{0.2} = 5 \text{ m}^{-1}$. As stated in the previous chapter, the motor transfer function was obtained from [1] and its coefficients have values $K_m = 0.1878$ and $\tau_m = 0.187$. The ball is assumed to be a hollow sphere (i.e. spherical shell), thus $K = \frac{3}{5}$ as derived in (2.29) and g is the gravitational acceleration.

A sampling period has to be determined in order to acquire discrete transfer function. As the camera used in the real model has maximum sampling frequency of 30 fps, it is pointless to choose a sampling period smaller than $\frac{1}{30}$ seconds. Additionally, the personal computer used to control real model is not fast enough to process camera images as quickly, thus the camera's sampling frequency was set to 10 fps. The sampling period was chosen accordingly to this limitation to be $T_s = 0.1 \text{ s}$. Nevertheless, the smaller sampling period was picking up all jiggling movements of the ball which led to unnecessary corrections from the controller. Discretized plant model is in equation (5.2) and its pole-zero map in Fig. 15.

$$G(z^{-1})_{T_s=0.1s} = \frac{0.00396z^{-1} + 0.01394z^{-2} + 0.00304z^{-3}}{1 - 2.5871z^{-1} + 2.1743z^{-2} - 0.5871z^{-3}} \quad (5.2)$$

Following the plant and controller structures specified in chapter 3 in equations (3.2), (3.9), (3.10) and characteristic polynomial $D_6(z^{-1})$ in (3.12), it is possible to determine three optimal roots of this polynomial using spectral factorization. The Polynomial toolbox in MATLAB [9] and its function $spf(A*qu*A' + B*B')$ were used to do spectral factorization of equation (3.14). Optimal roots are $0.8477 \pm 0.1409i$ and 0.5821 for $q_u = 1$. Remaining three roots were chosen to be $[0.8 \ 0.8 \ 0.8]$. Finally, using equations (3.15) and (3.16):

$$C_b(z^{-1}) = \frac{-2.3556 + 5.8538z^{-1} - 4.7512z^{-2} + 1.2489z^{-3}}{1 - 1.1796z^{-1} + 0.4187z^{-2}} \quad (5.3)$$

$$C_f(z^{-1}) = \frac{-0.0041}{1 - 1.1796z^{-1} + 0.4187z^{-2}}$$

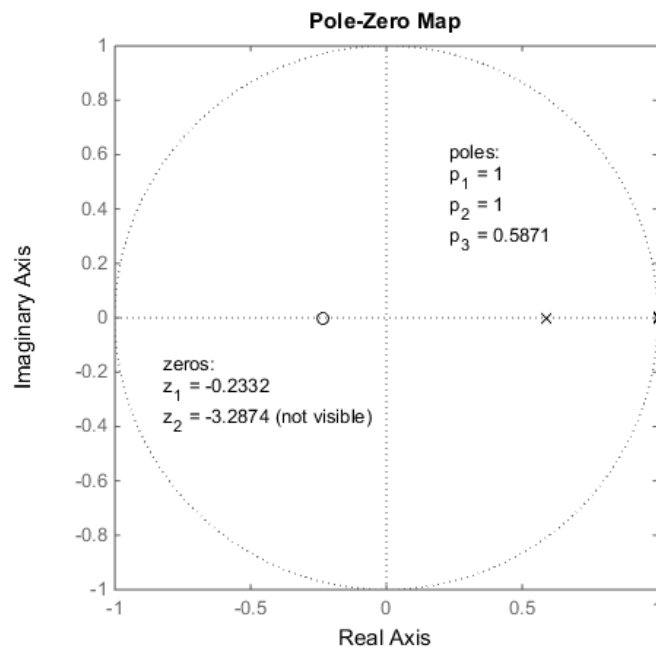


Fig. 15 Pole-zero map of the plant

5.3 Simulation results

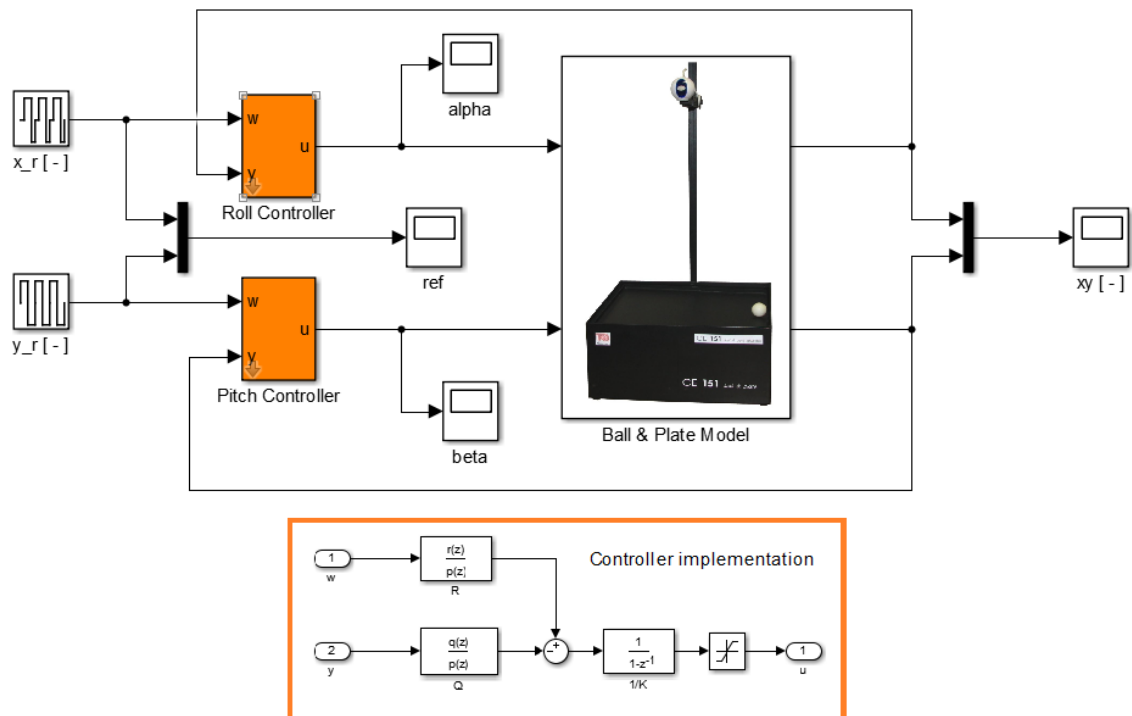


Fig. 16 Ball & Plate control simulation model

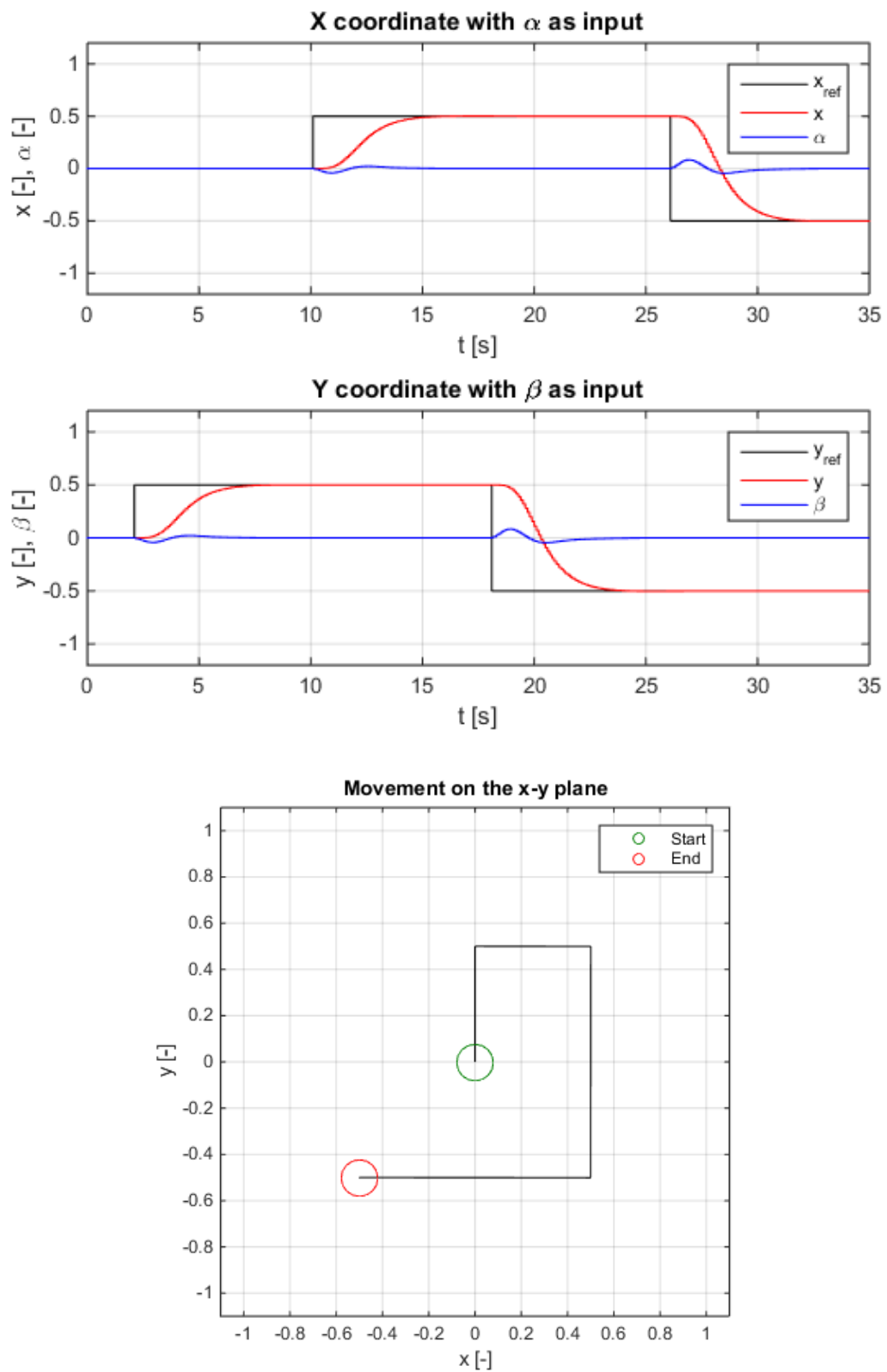


Fig. 17 Simulated step reference tracking

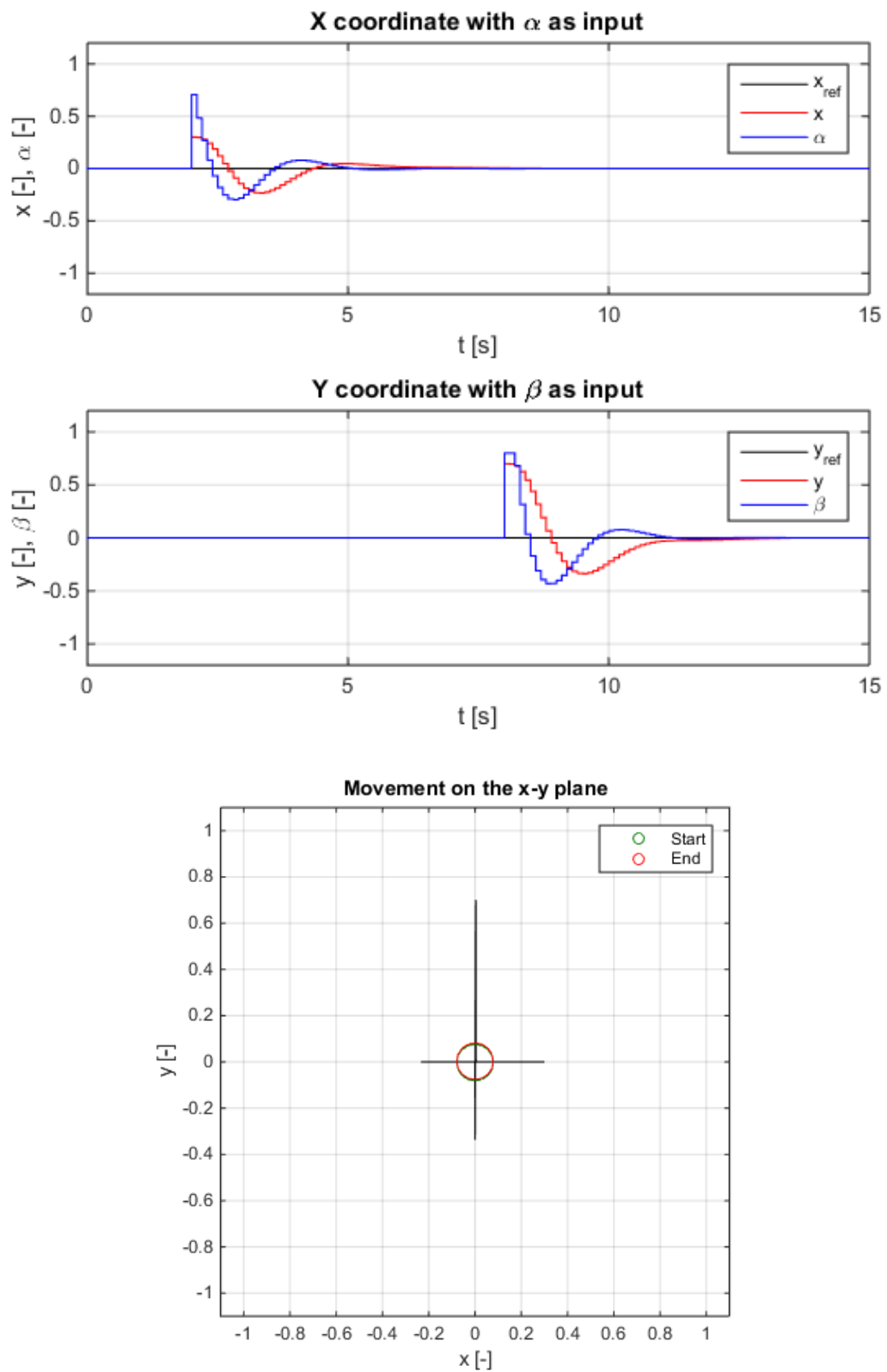


Fig. 18 Simulated step disturbance rejection

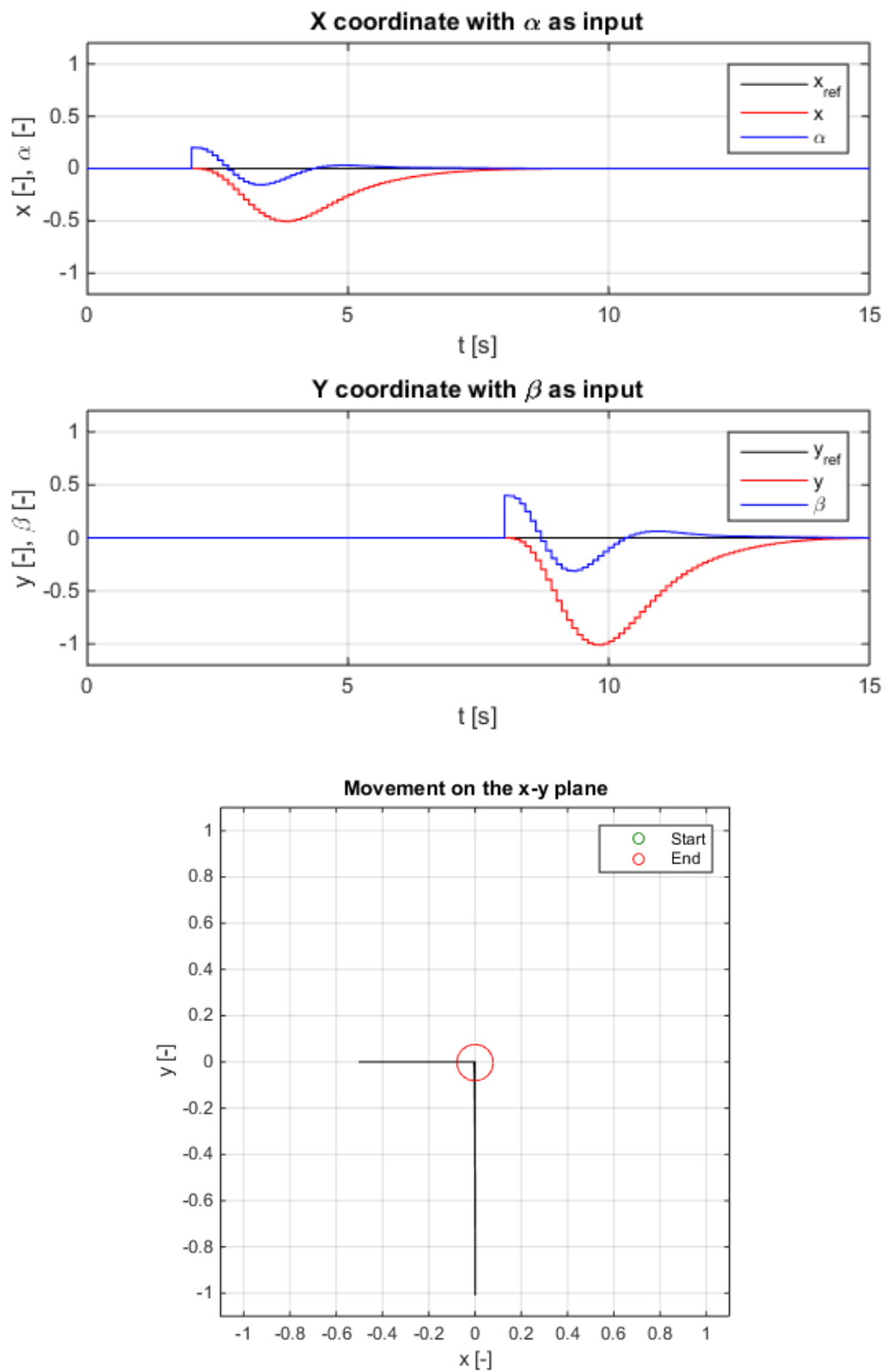


Fig. 19 Simulated step load disturbance rejection

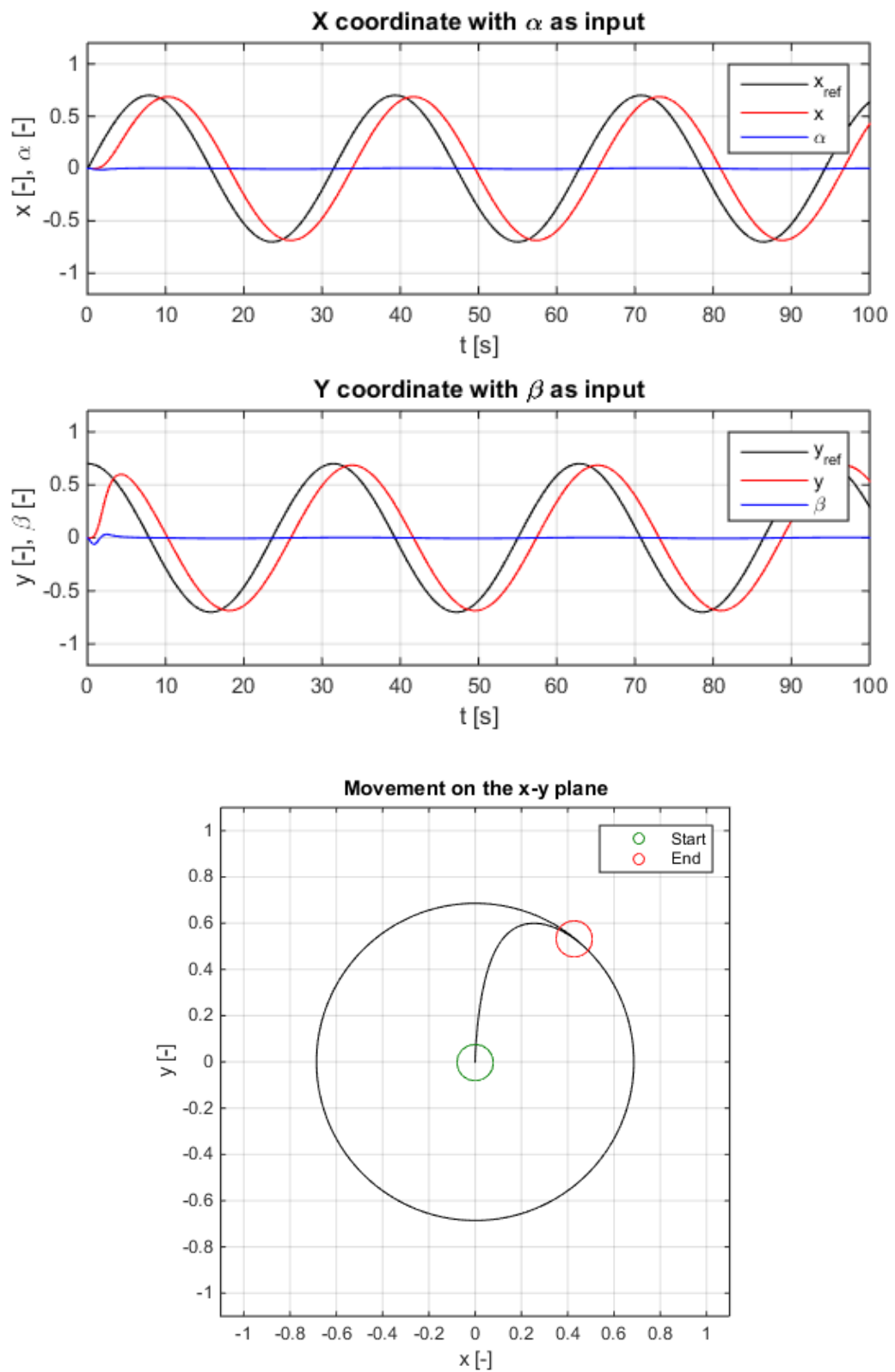


Fig. 20 Simulated circular reference tracking – relatively low frequency

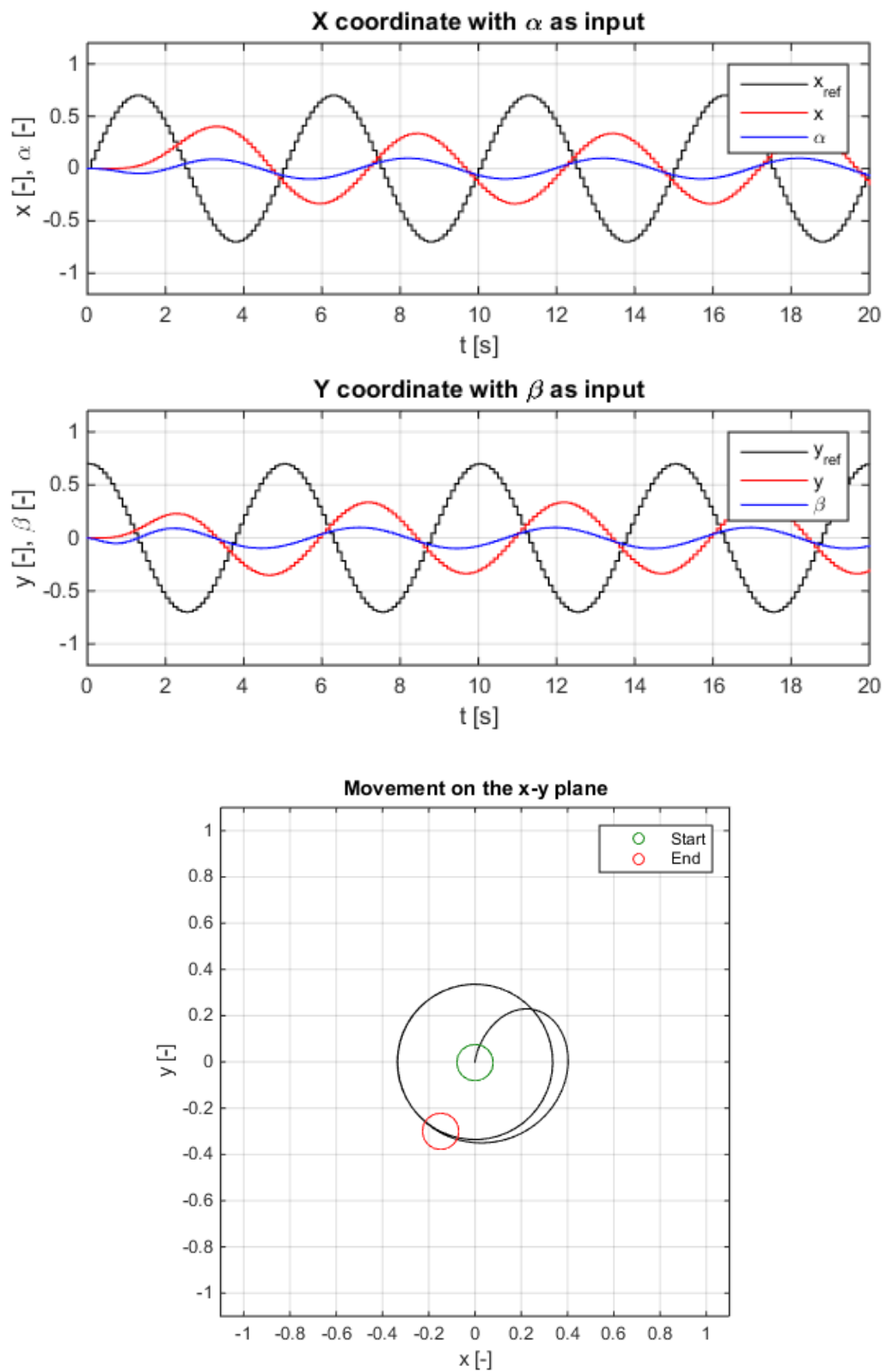


Fig. 21 Simulated circular reference tracking – relatively high frequency

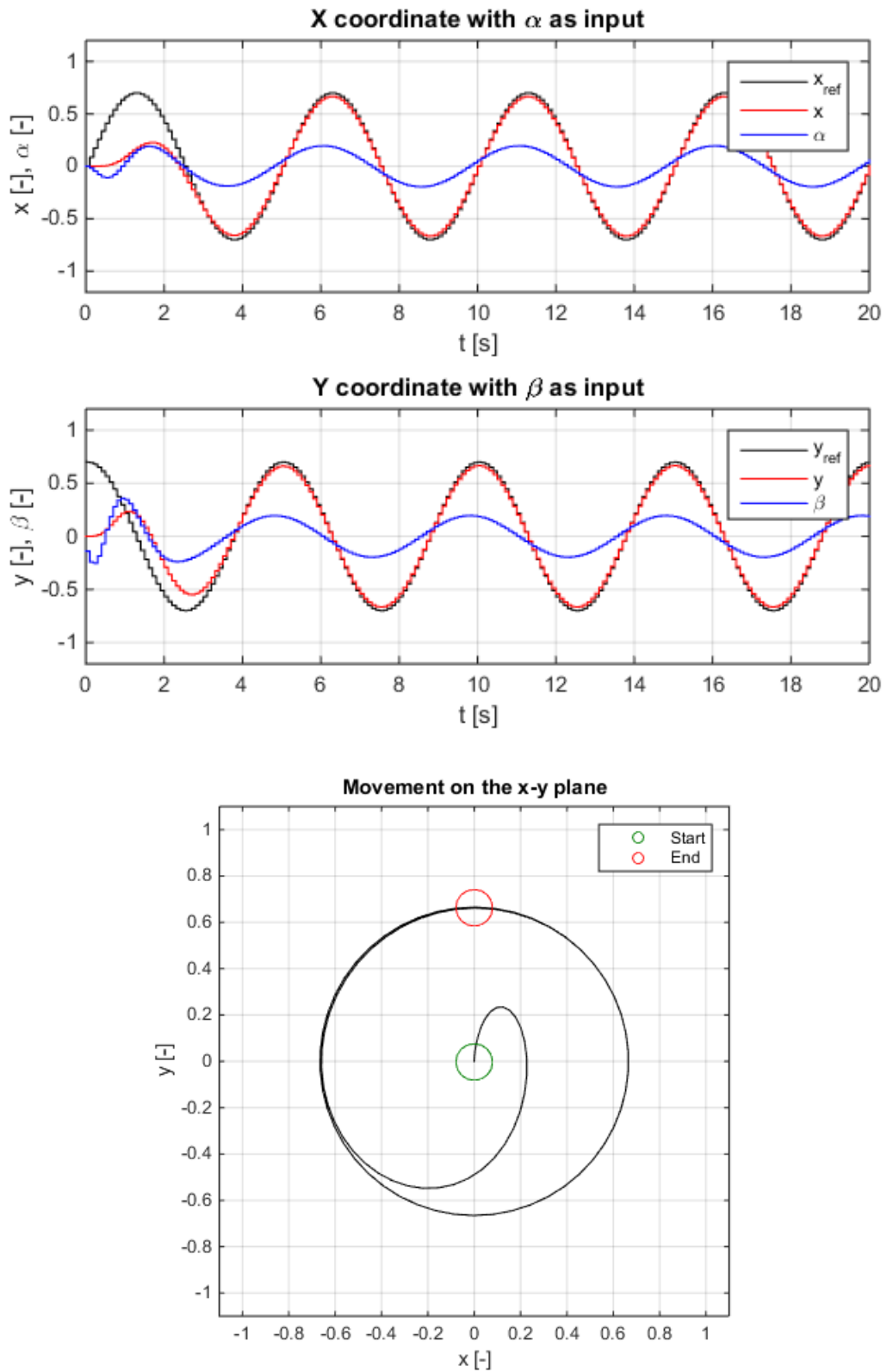


Fig. 22 Simulated circular reference tracking with sinusoidal controller design

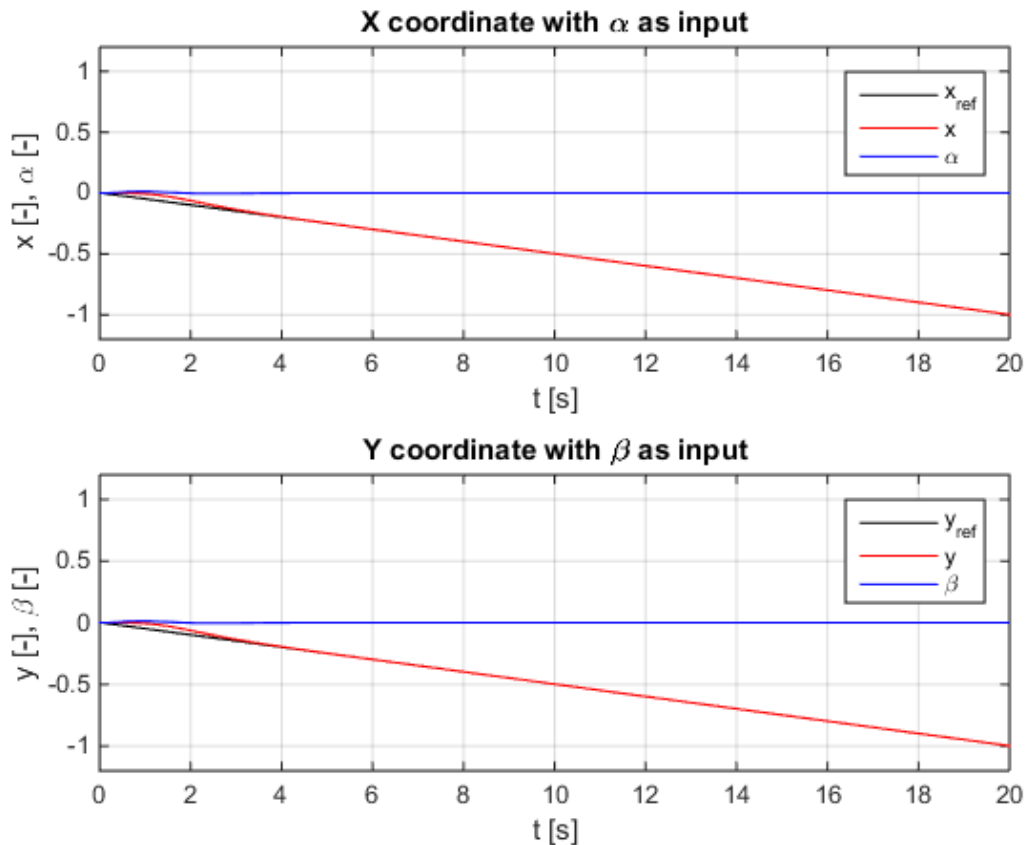


Fig. 23 Simulated ramp reference tracking with ramp controller design

Controllers are able to track sinusoidal reference value with relatively low frequency (apart to the phase), but as the frequency goes higher, controllers don't keep up and although the motion is still circular, the amplitude is much smaller, as presented in Fig. 20 and Fig. 21. This could be easily compensated by moving three chosen poles closer to the center, but this could lead to instability. The better solution is to design controller for sinusoidal reference, instead of step reference, which removes the phase lag (Fig. 22). Controllers were also designed for ramp reference (Fig. 23) to remove evident permanent error for ramp signal.

Three chosen poles of characteristic polynomial are relatively far from the center, so that output of controllers is steady. This results in longer rise time, but also in smaller and more subtle plate inclinations.

Inputs in graphs are show on the scale from -1 to 1, although they rarely go near these limits. Often the input is so much out of scale that is looks like straight line (e.g. in Fig. 20), but the purpose of this is to show how relatively very small angles affect greatly the ball's position. In addition, when the input is very small, its ratio is more interesting than its actual value.

6 REAL MODEL CONTROL

6.1 Simulink model

The Simulink model and its subsystems (Fig. 24) will be presented in this chapter. The Real Time Toolbox (*RT I/O*) is needed to communicate with MF624 driver card from Humusoft and the Image Acquisition Toolbox (*From Video Device*) to communicate with the camera. It also uses s-function supplied with the model for finding and determining ball's position (using mex-file and c-file) and *Trajectory Graph* for results monitoring and control. All other subsystems were created for the purpose of this thesis or are built-in Simulink blocks.

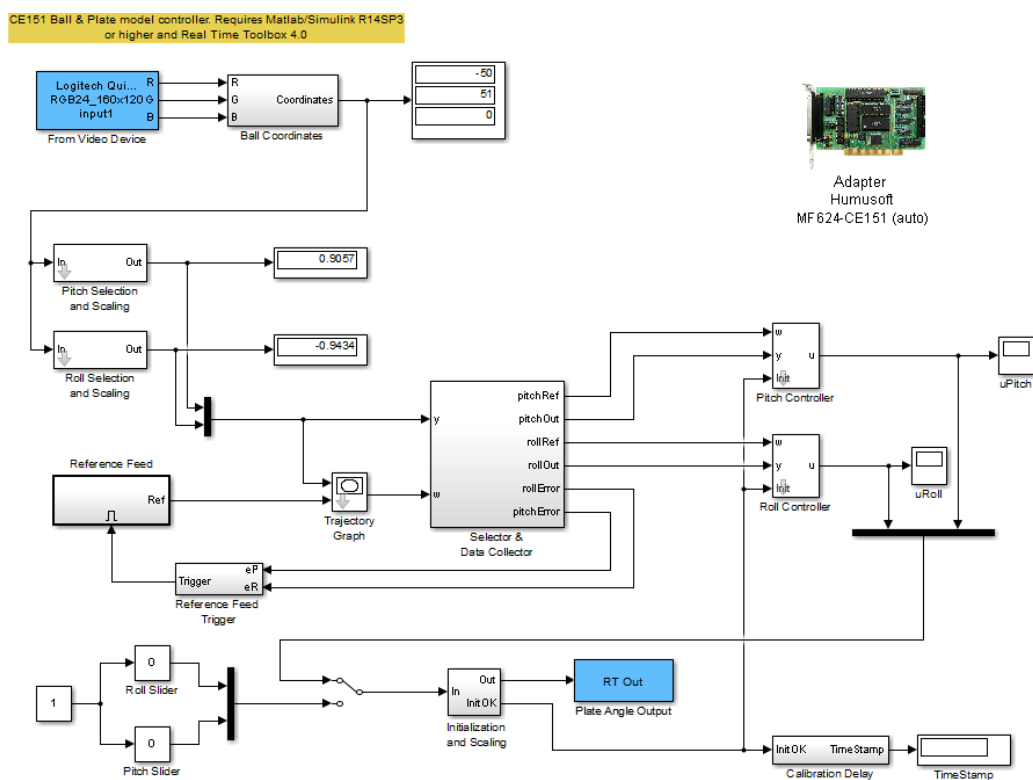


Fig. 24 Simulink scheme for real model control

The first subsystem (Fig. 25) called *Ball Coordinates* has 3 inputs (RGB components) and one 3-dimensional output (x position, y position and ball state – 0 for OK, 1 for NOT FOUND and 2 for BALL TOO BIG). RGB components are averaged and the whole image is trimmed and reshaped, after which the s-function is used to find ball's position. Ball position values are normalized and split in *Selection and Scaling* block (Fig. 26), which results in ball's coordinates ranging from -1 to 1 (borders of the plate), thus number 1 corresponds to 20 cm on the plate. Signals are then rerouted and collected (Fig. 27).

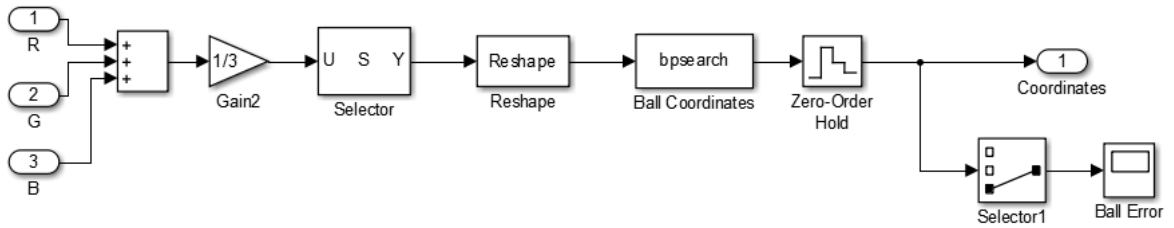


Fig. 25 Ball Coordinates Subsystem

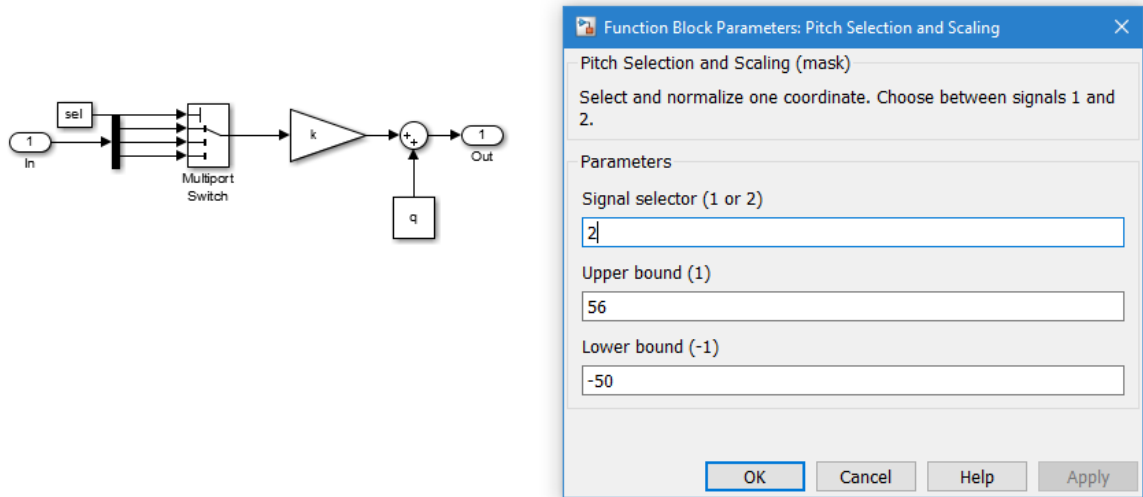


Fig. 26 Selection and Scaling subsystem with mask options

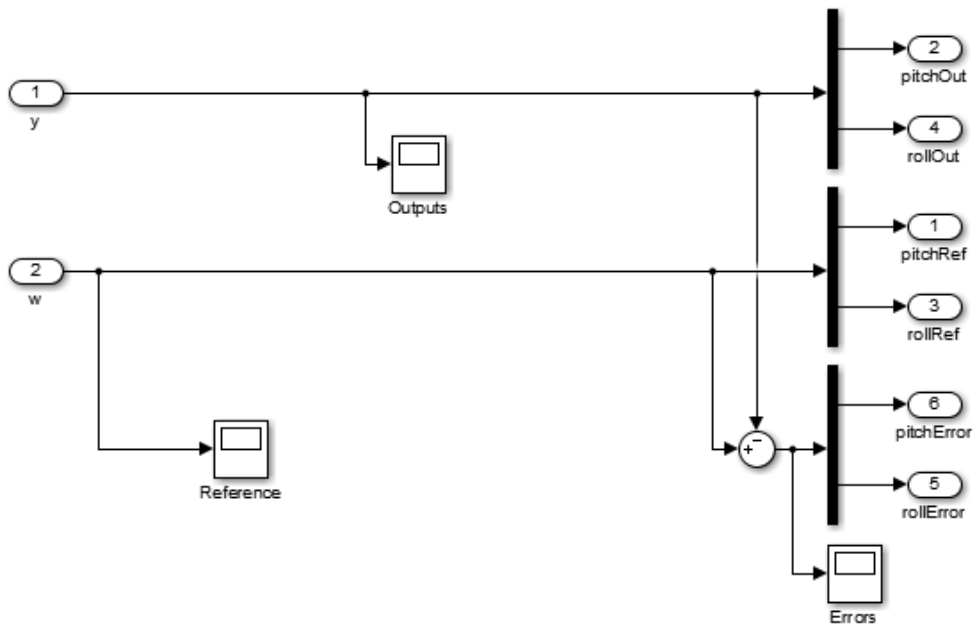


Fig. 27 Selector & Data Collector subsystem

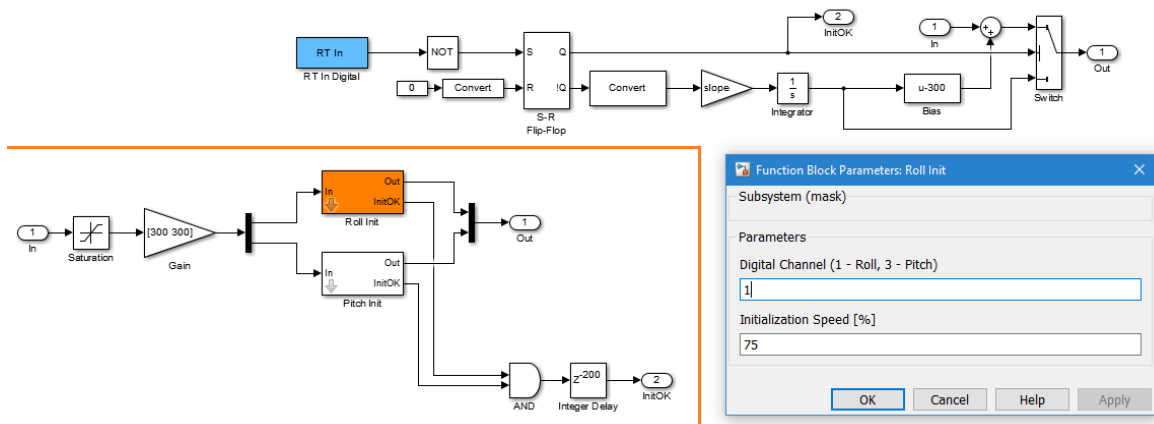


Fig. 28 Initialization and Scaling (left) and its Roll Init subsystem (right)

The real model has to be initialized first, so that it always starts in horizontal position (zero inclination angles). During initialization all controllers are switched off because they would be influenced by this “low level” plate control. The initialization uses digital outputs of the model, which are four switches located under the plate in centers of casing sides. With certain plate angle, the switch is activated and this signal is sent to *RT In Digital* block as logical one. Initialization consists of slowly raising angle until the switch is activated and angle at that moment is used as a bias value. Outputs of this subsystem (Fig. 28) are scaled input and logical *InitOK* signal which controls start of the regulation. To know the time when the regulation starts, the *Calibration Delay* subsystem was created (Fig. 29), which stores and outputs this time. The controller subsystem is shown in Fig. 30.

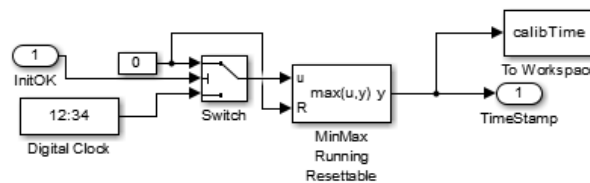


Fig. 29 Calibration Delay subsystem

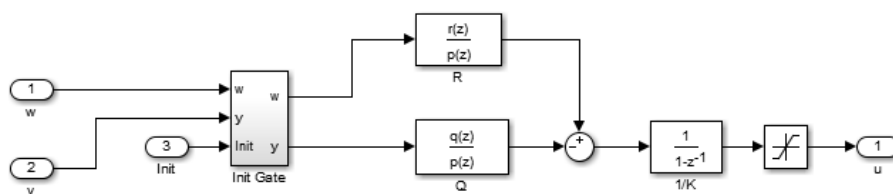


Fig. 30 Controller subsystem

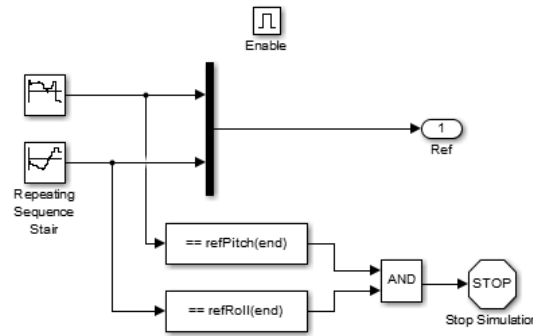


Fig. 31 Reference Feed subsystem

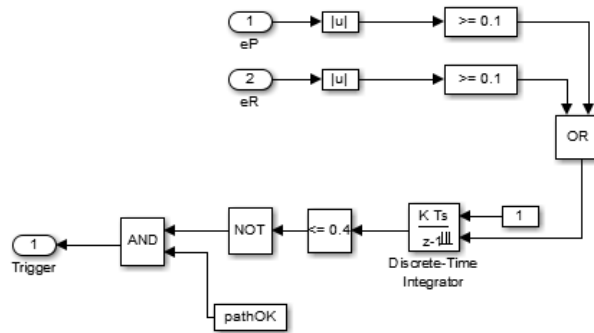


Fig. 32 Reference Feed Trigger subsystem

The reference value can be chosen manually by clicking with mouse on the desired position in *Trajectory Graph* block or the input can be custom and *Trajectory Graph* just passes this input to next block. For the purpose of chapter 6.4, the subsystem that would feed desired reference value as needed had to be created. The subsystem *Reference Feed* (Fig. 31) contains reference vector in *Repeating Sequence Stair* block, which is simply passed to its output. The *Enable* block is dealing with actual reference feeding. It freezes the whole subsystem if the enable input to the *Reference Feed* subsystem is not true. Whole subsystem is thus triggered by the output of the *Reference Feed Trigger* subsystem (Fig. 32). This trigger is fired only if errors of both coordinates are less than 0.1 for the duration of 0.4 s.

6.2 Controller design

The controllers for the real model were designed in the similar fashion as in the chapter 5.2, but the used plant was from the identification in chapter 4 for step angle 40% of the maximum, showed in Fig. 10. This identified plant is presented in the following expression:

$$G(s) = \frac{5.7402}{s^2(0.1877s + 1)} \quad (6.1)$$

Parameters obtained from experimental identification, modelling and the manual page [1] are compared in Fig. 33. The parameter K is negative because of inverted x and y coordinates in the real model. Parameter T is the same for the modelling part and manual page, because this parameter was obtained from there as the time constant of motors.

$G(s) = \frac{K}{s^2(Ts+1)}$	K	T
Manual Page	4.803	0.187
Modelling	-5.0706	0.187
Identification	5.7402	0.1877

Fig. 33 Comparison of models

The discretized model of identified plant:

$$G(z^{-1})_{T_s=0.1s} = \frac{0.00449z^{-1} + 0.01579z^{-2} + 0.00344z^{-3}}{1 - 2.5870z^{-1} + 2.1741z^{-2} - 0.5870z^{-3}} \quad (6.2)$$

Optimal roots are $0.8391 \pm 0.1491i$ and 0.5811 for $q_u = 1$. Remaining three roots were chosen to be also $[0.8 \ 0.8 \ 0.88]$. Resulting controllers:

$$C_b(z^{-1}) = \frac{2.2372 - 5.5540z^{-1} + 4.5040z^{-2} - 1.1831z^{-3}}{1 - 1.1796z^{-1} + 0.4187z^{-2}} \quad (6.3)$$

$$C_f(z^{-1}) = \frac{0.0041}{1 - 1.1623z^{-1} + 0.4118z^{-2}}$$

6.3 Real model control results

The original idea was to use approximation of the model with first order astatism and time delay, because lower order of the system simplified the model. The approximation was quite precise and also controller with time delay compensation using digital Smith Predictor [12] was designed. Its usage was proven to be difficult, because stepper motors were losing steps (breaking the assumption in 2.3). Lost steps are load disturbance that cannot be measured and because Smith Predictor relies on knowing the input to the system (which is unknown due to lost steps), a permanent error occurred that couldn't be removed. One of the options was to use adaptive controller, however it was not possible due to small sampling period and slow personal computer used to control the model. Lost steps compensated by controllers can be seen in Fig. 34 as non-zero controller outputs α and β .

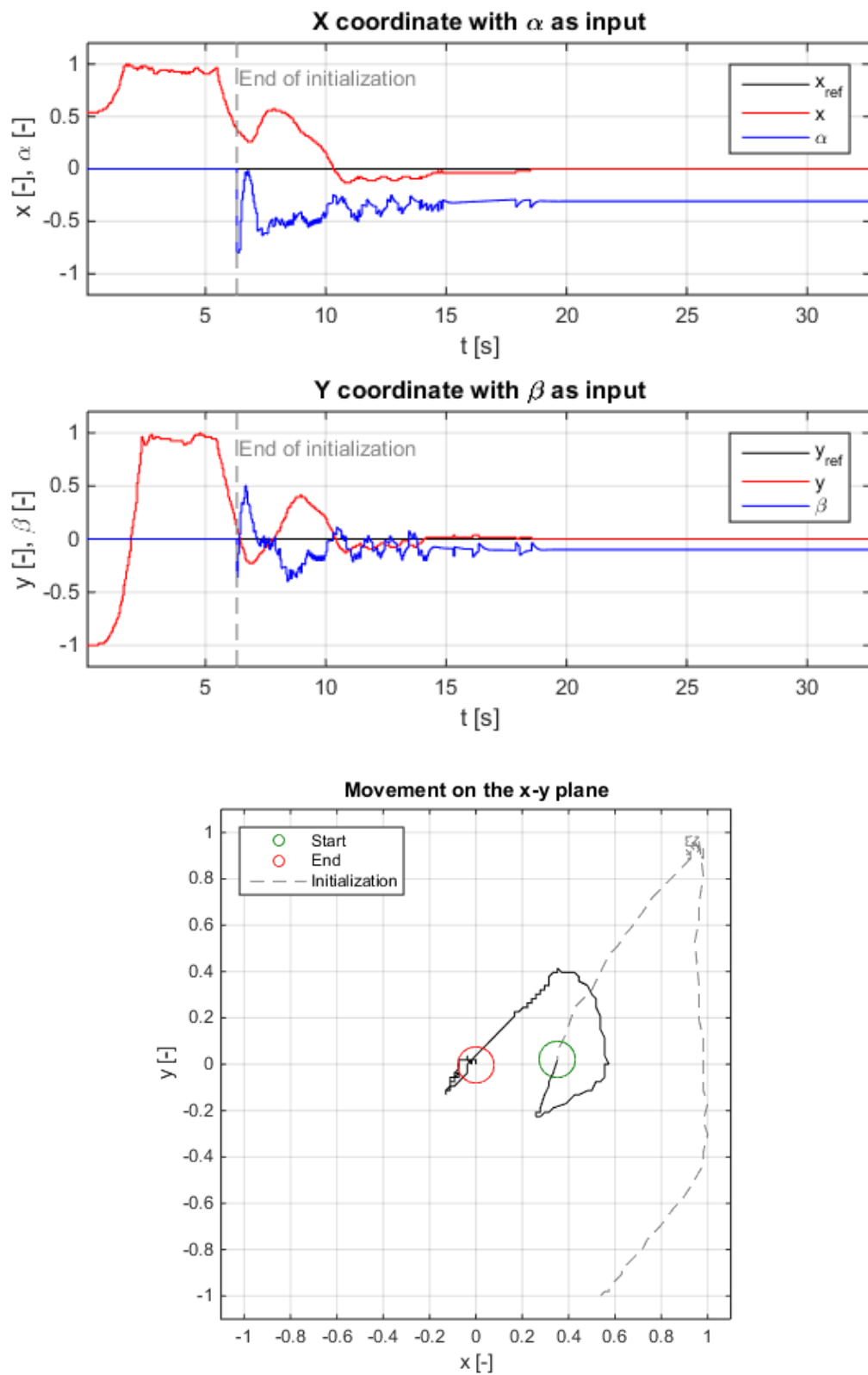


Fig. 34 Reference tracking

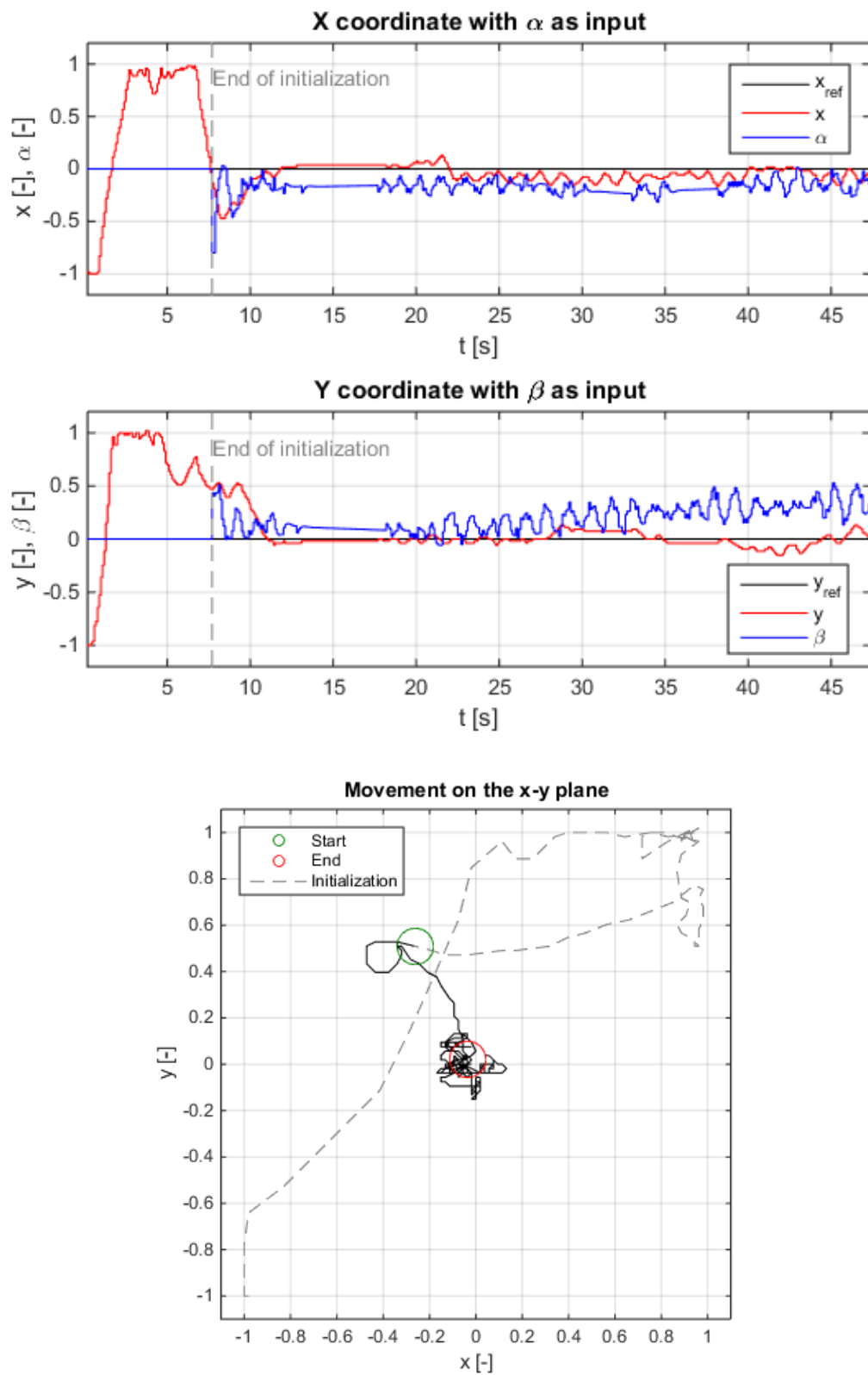


Fig. 35 Disturbance rejection (blowing to the ball)

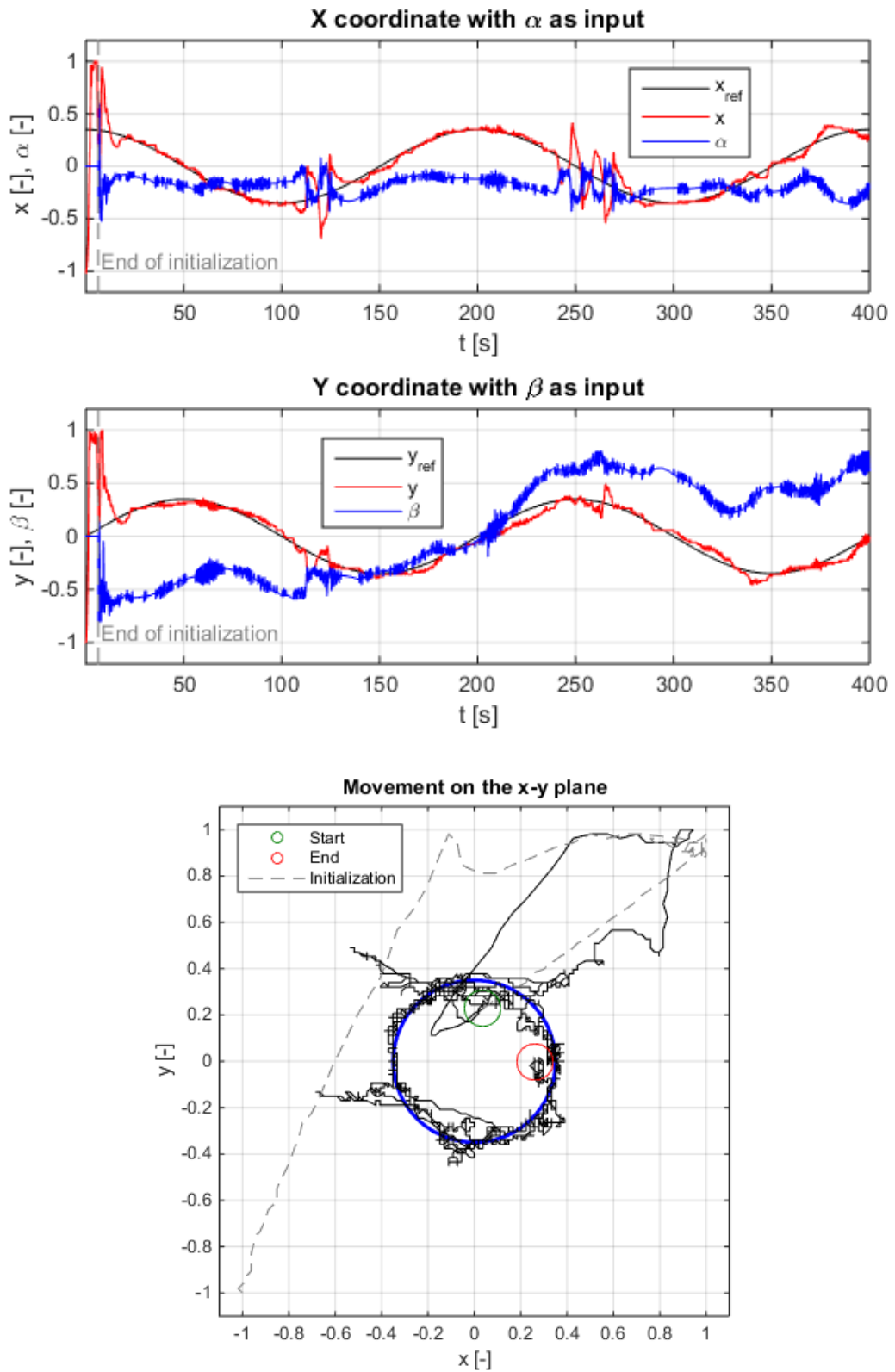


Fig. 36 Circular reference tracking

6.4 Navigating the maze

6.4.1 Automatic path determination

In addition to classical reference values, the maze was constructed on the plate using a blue tape (Fig. 37). The blue RGB component from camera output was removed (although this doesn't mean the blue color was removed), because the blue tape would add unwanted noise.



Fig. 37 Maze

Next, the simple color segmentation was used. Taking only the blue RGB component is not enough, because presence of the blue component doesn't necessarily mean the presence of the blue color as perceived by humans, as shown in Fig. 38. Thus the simple formula for computing the blueness b [13] of the image was used: $b = B - \max(R, G)$.

(255, 0, 0)	→	-255
(0, 255, 0)	→	-255
(0, 0, 255)	→	255
(127, 127, 255)	→	128
(255, 0, 255)	→	0
(0, 255, 255)	→	0

Fig. 38 Color perception [13]

A blueness mask was created from blueness image using appropriate threshold value and the redundant noise was removed, thus only binary mask of walls remains (Fig. 39). The watershed transform (see 6.4.3) and further noise cleaning was used to obtain the solution of the maze [14] shown in Fig. 40.

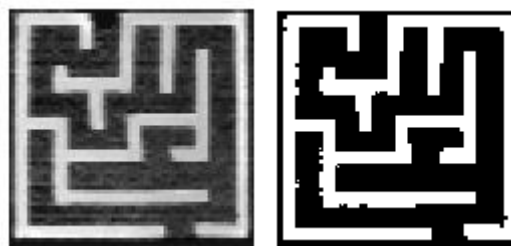


Fig. 39 Blueness picture and its mask

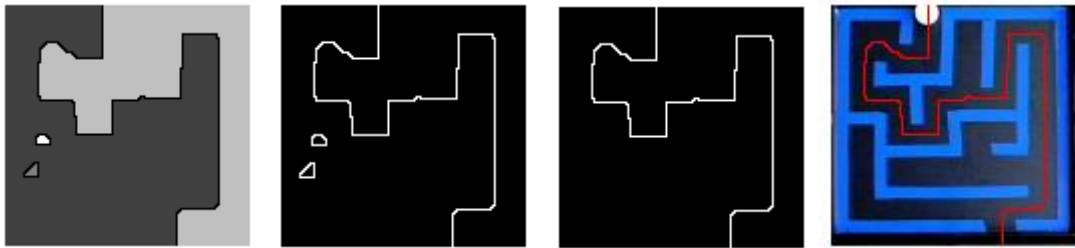


Fig. 40 Watershed transform steps

A sequence of reference values was obtained from the computed path in the form of logical matrix. For reference to be step-changing signal, only corner points of the path were chosen. Various reflections and foreign objects could confuse the algorithm, because this method is based on color segmentation. That's why the user is prompted to check the result (Fig. 41). The only downside of this method is that the maze has to be perfect with one entrance, one exit and without loops (algorithm generates a warning otherwise - Fig. 42).

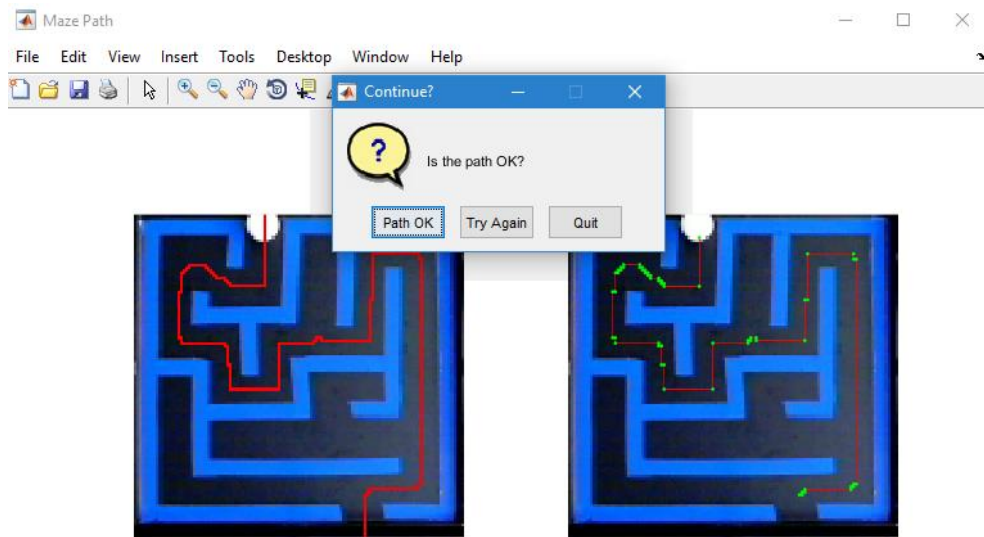


Fig. 41 Algorithm completion with user prompt

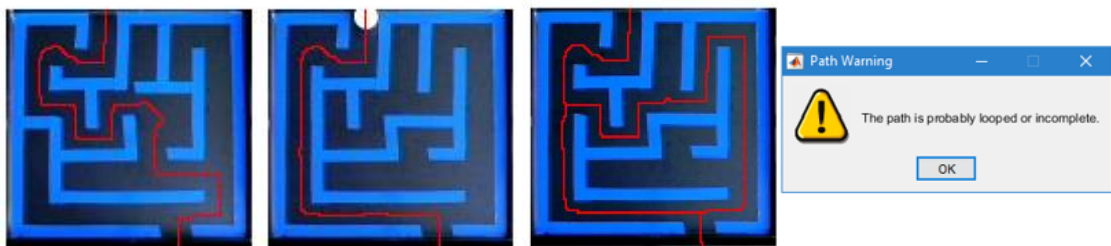


Fig. 42 Different paths test and looped path with warning

6.4.2 Maze navigation results

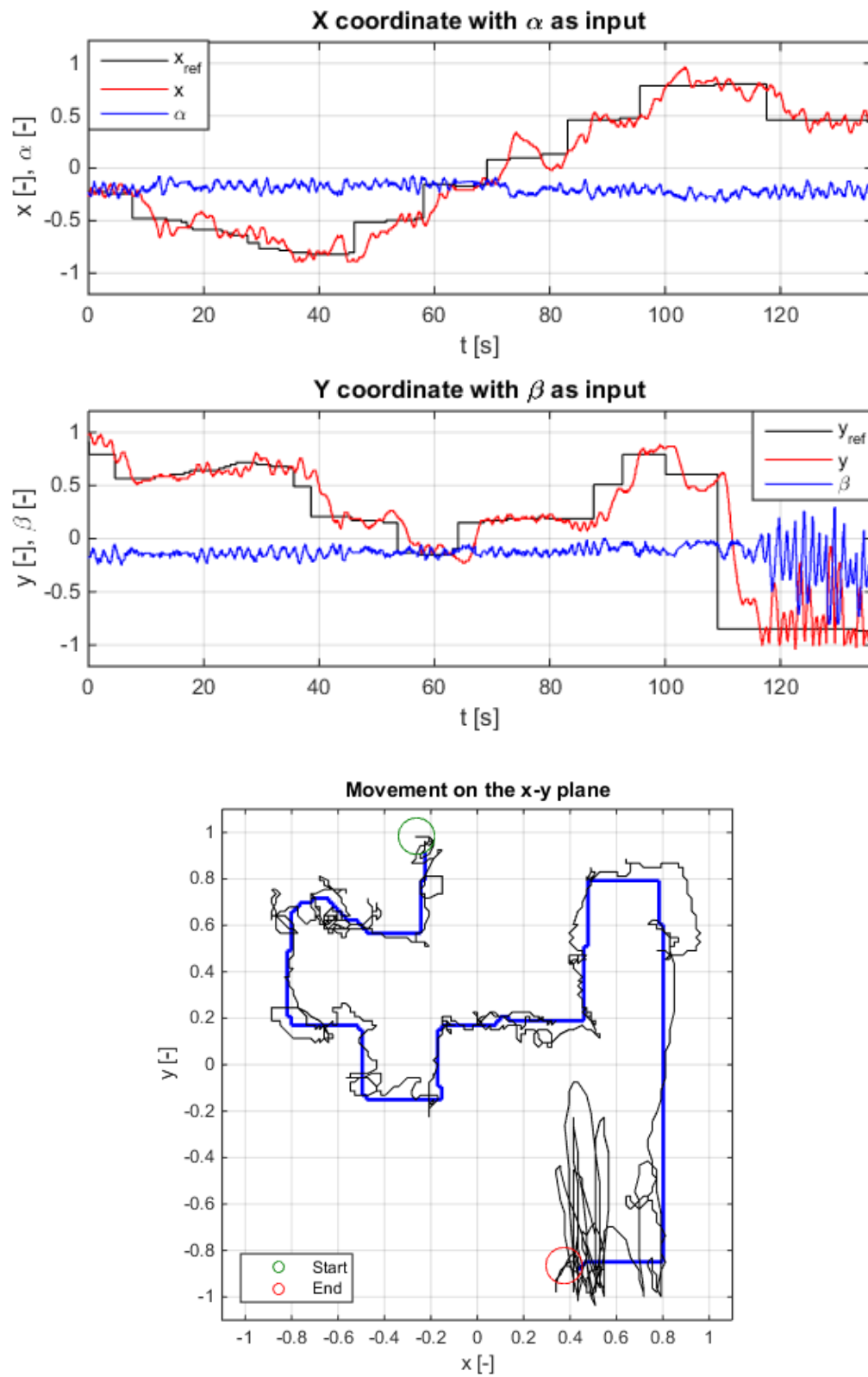


Fig. 43 Maze navigation – first run

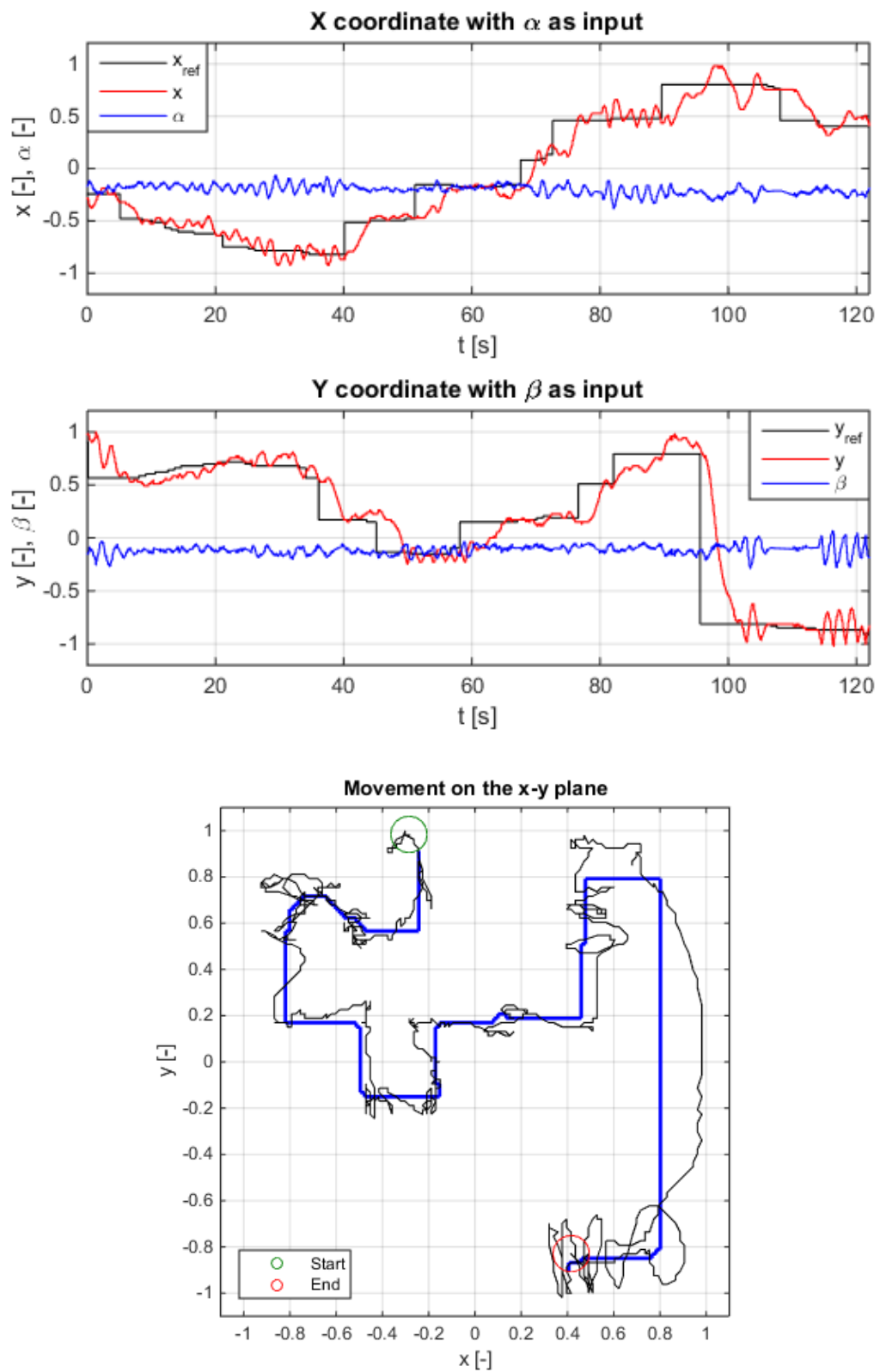


Fig. 44 Maze navigation – second run

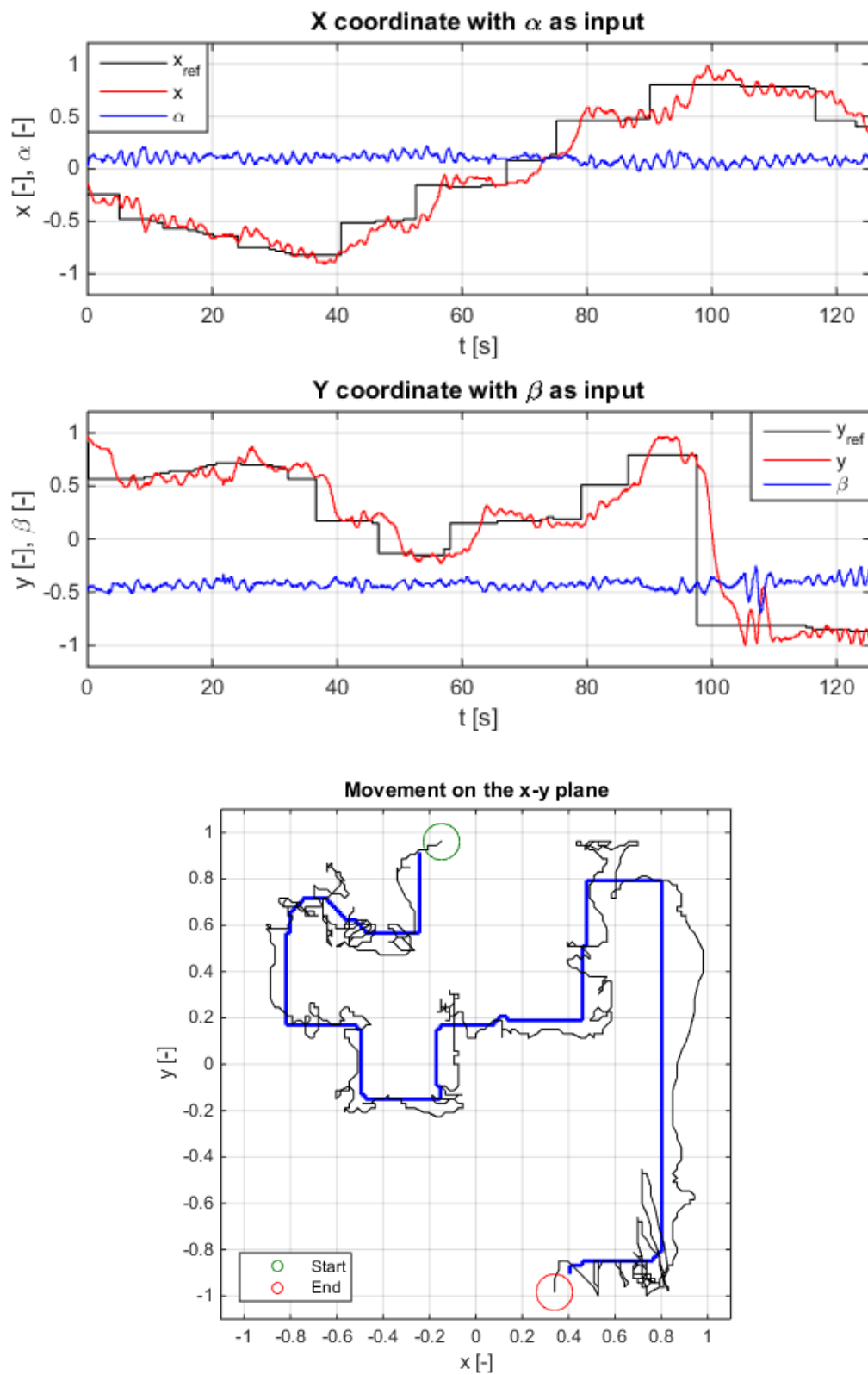


Fig. 45 Maze navigation – third run

6.4.3 Watershed transform

The term watershed refers to a ridge that divides areas drained by different river systems as shown in Fig. 46. A catchment basin is the geographical area draining into a river or reservoir. [15] In image processing, it was introduced as a tool for segmenting grayscale images by S. Beucher and C. Lantuéjoul in the late 70's. It considers a grayscale image as a topographical relief (the grey level of a pixel represents the elevation of a point, where dark areas are "low" and bright areas are "high"). [16] The example grayscale image and its 3D surface is in Fig. 47. The *watershed* function in MATLAB detects these watershed regions and outputs them in the matrix of the same size as the input image (Fig. 40 - left).

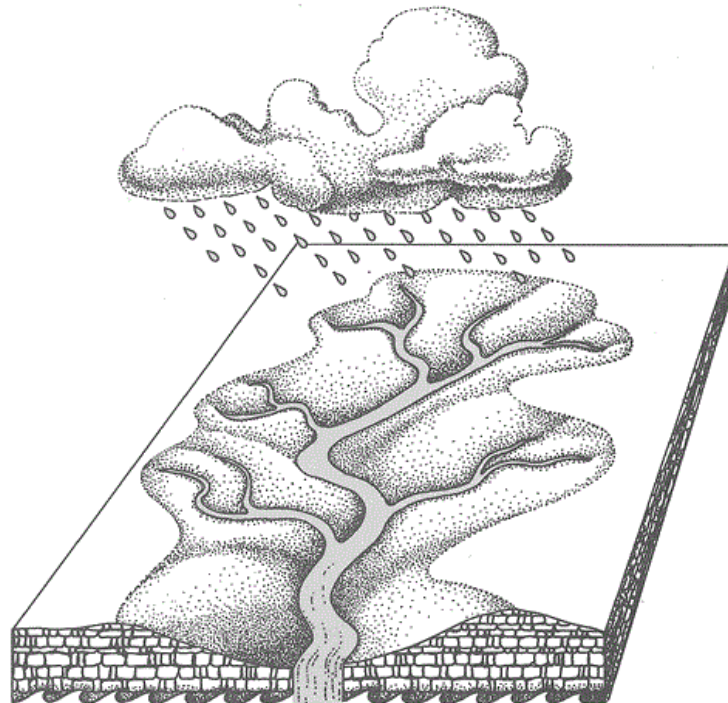


Fig. 46 Watershed drainage [17]

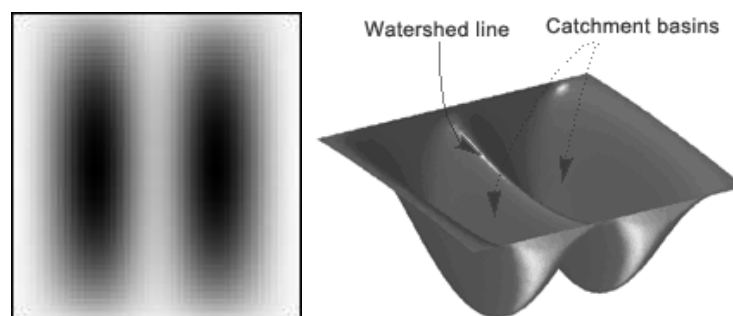


Fig. 47 Grayscale image and its topological relief [15]

7 GRAPHICAL USER INTERFACE

The graphical user interface (GUI) was designed using MATLAB's GUIDE design environment to control simulation or the real model. As mentioned on few occasions, the personal computer used to control the model is slow and it barely managed to run GUI on top of the Simulink application. Because of this, the real model control is not supported, although its layout was prepared for such option. Only simulation mode is fully operational and the GUI is directly connected to the Simulink simulation model (modified version of model in Fig. 16). It is straightforward to use this GUI and doesn't need detailed explanation. Warning and error dialogs were implemented to prevent user from choosing parameters and inputs that are not expected. The designed GUI implements 4 sources of the reference value as seen in Fig. 48: manual input using edit boxes X_{ref} and Y_{ref} , circular reference, maze navigation and MAT-file. By choosing either of these sources, the user is prompted accordingly. Sliders to the right and above the axes have no actual use in simulation mode as their purpose is to manually control plate inclination. Note also that the classical MATLAB's gray figure background is removed only for the sake of this document.

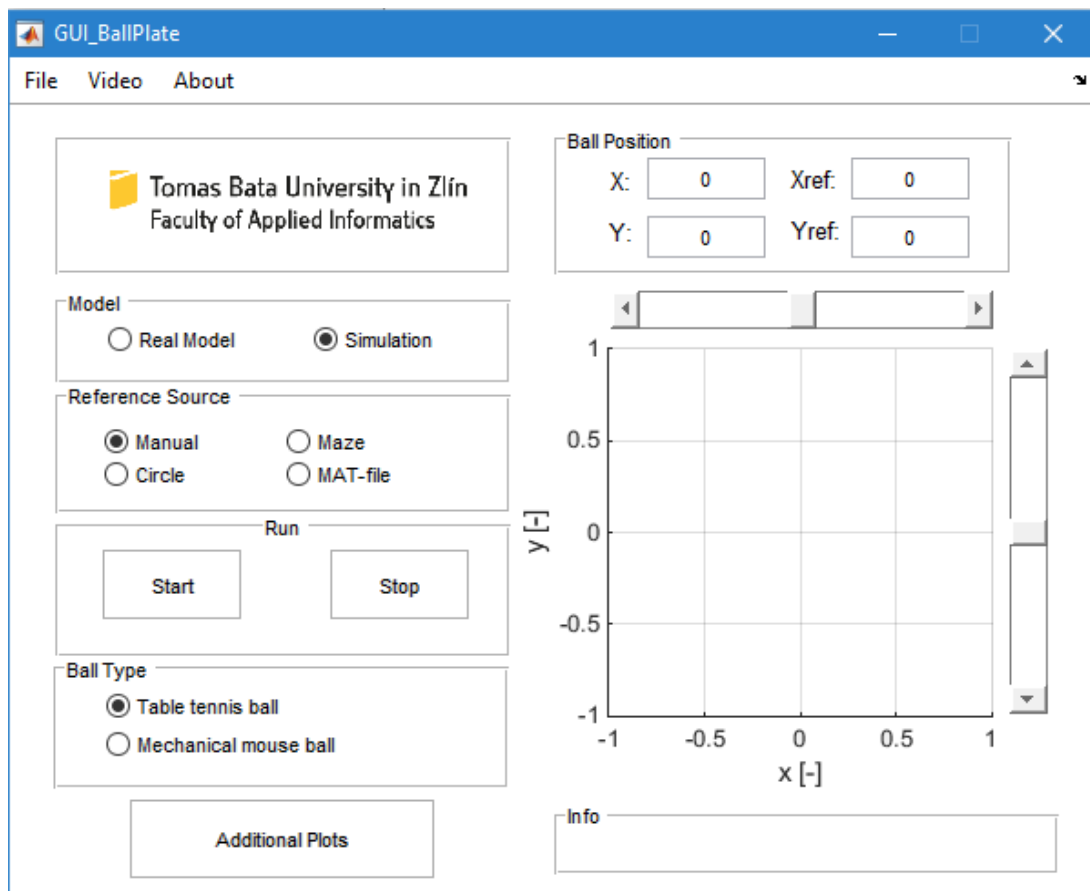


Fig. 48 Ball & Plate model GUI

As seen in Fig. 49, by clicking the *Additional Plots* toggle button, extra plots appear with radio buttons to choose desired plot. Zoom in/out and Data Cursor palettes also appear under the context menu. The Fig. 49 shows a maze simulation example where all line and marker colors correspond to colors in chapters 5.3, 6.3 and 6.4.2 (except of red color representing ball position in x-y plane).

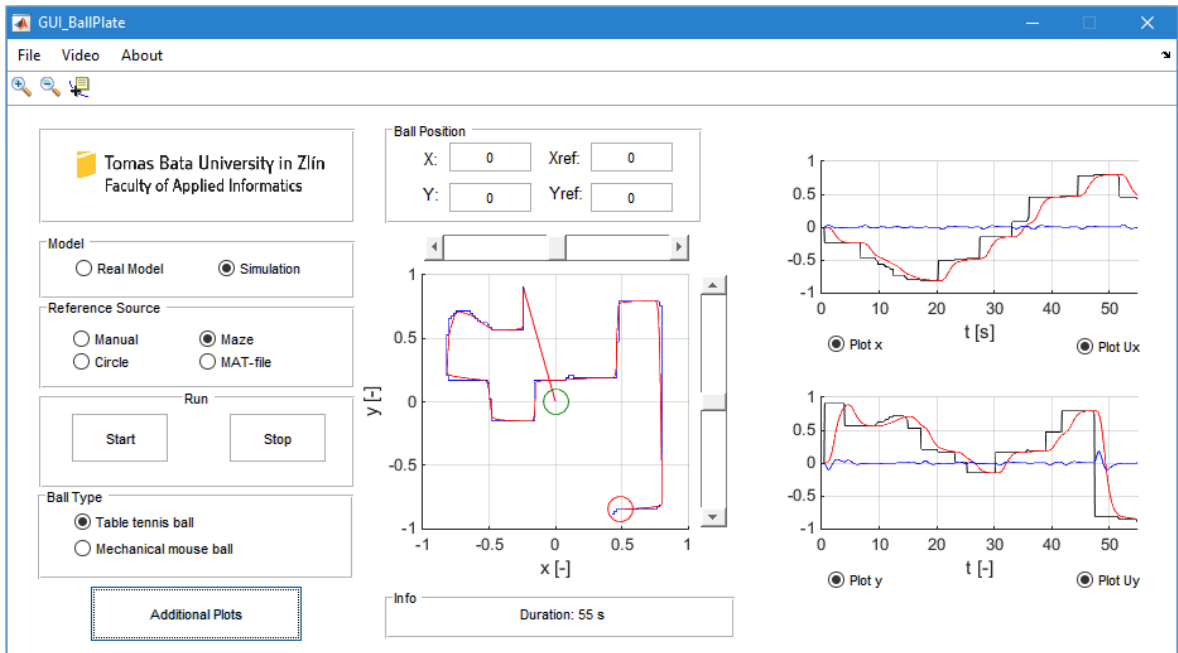


Fig. 49 Additional plots in GUI with maze navigation simulation results

In Fig. 50, the circle parameters user prompt (left) and the context menu content (right) are shown to introduce the complexity of the GUI and its preparedness for the real model. All other buttons and GUI elements are self-explanatory and don't require further attention.

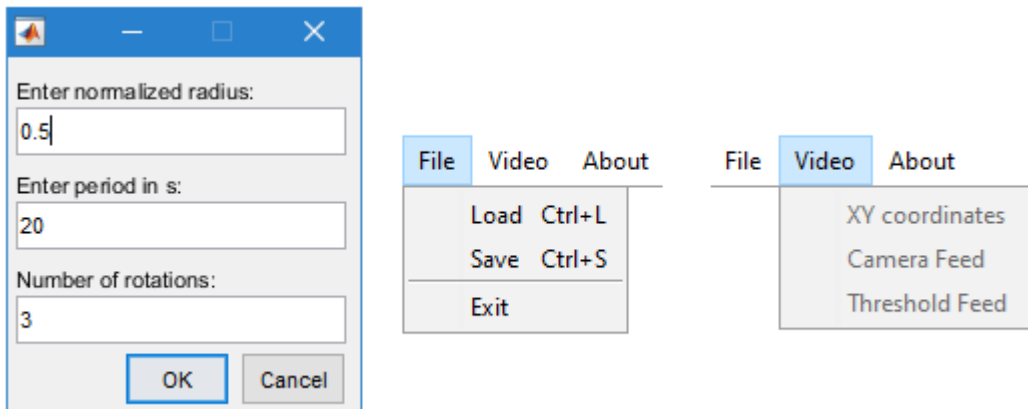


Fig. 50 Circle parameters user prompt and context menu content

CONCLUSION

This thesis introduced the CE 151 Ball & Plate model from Humusoft, its structure, parameters, downsides, mathematical description and the design of suitable digital controller. The model was not perfect in terms of an ideal Ball & Plate system, mostly because of its configuration using steel wires. This caused motors to lose steps, which ultimately led to random unmeasurable load disturbance. Possible solutions were to directly access the hardware and electronics or change the driver's software, neither of which was the aim of this thesis. The computational limitations of personal computer used to run Simulink environment had to be also taken into account as it contained the communication card and thus could not be changed to faster one.

The LQ controller design proved to be easy to use after appropriate algorithms were created and optimal poles computed. It was surprisingly easy to place remaining poles as they had to be further from the center of complex plane to make the controller output bounded. Together with optimal poles, the resulting controller was able to track the reference and reject disturbances. This solution was robust enough to even deal with switching the ball "on the flight", although the rise time and settling time were clearly worse.

The model was experimentally identified for different inputs (plate inclinations) and two different balls, resulting in 26 identified models. The chosen model was the one for small plate inclinations, as it is the assumed linearization point. As the model is assumed to be symmetrical, the same model was used to design both controllers.

After initial preparations, the simulated model was controlled for different reference signals and disturbances to show its ideal behavior. The real model was controlled afterwards in the same manner. Additionally, the maze navigation was implemented to make the reference signal selection process more interesting and algorithmic. To encapsulate all tested reference signals, the graphical user interface (GUI) was designed, although for simulation purposes only. This proved to be a valuable tool to quickly change reference values and see the final outcome of the simulated model. The layout of the GUI was also prepared for the real model control, but due to mentioned PC limitations, it was not implemented as a functional solution.

BIBLIOGRAPHY

- [1] HUMUSOFT. CE 151 Ball & Plate Apparatus User's Manual. Prague, 2006.
- [2] ČUPIĆ, Grgo. *Implementacija nelinearnih algoritama upravljanja platformom s dva stupnja slobode* [online]. Zagreb, 2011 [cit. 2016-04-14]. Available from: http://bib.irb.hr/datoteka/534227.gcupic_diplomski_rad_2011.pdf
- [3] BRUCE, Jonathan, Chris KEELING a Ronald RODRIGUEZ. *Four Degree of Freedom Control System Using a Ball on a Plate* [online]. Southern Polytechnic State University, 2011 [cit. 2016-04-23]. Available from: <https://users.soe.ucsc.edu/~jbruce/webfiles/BPFinalReport.pdf>
- [4] BOHIGAS, Oriol, Montserrat MANUBENS, Lluís ROS. A Linear Relaxation Method for Computing Workspace Slices of the Stewart Platform. *Journal of Mechanisms and Robotics* [online]. 2013, **5**(1), 011005- [cit. 2016-04-15]. DOI: 10.1115/1.4007706. ISSN 1942-4302. Available from: <http://mechanismsrobotics.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4007706>
- [5] NOKHBEH, Mohammad and Daniel KHASHABI. *Modelling and Control of Ball-Plate System* [online]. Amirkabir University of Technology, 2011 [cit. 2016-04-23]. Available from: http://web.engr.illinois.edu/~khashab2/files/2011_LinearControl/16.pdf. Under the supervision of H.A. Talebi.
- [6] MATUŠŮ, Radek and Roman PROKOP. Algebraic design of controllers for two-degree-of-freedom control structure. *International Journal of Mathematical Models and Methods in Applied Sciences* [online]. 2013, vol. 7, iss. 6, s. 630-637 [cit. 2016-04-28]. ISSN 1998-0140. Available from: <http://www.naun.org/cms.action?id=5358>.
- [7] BOBÁL, V. *Adaptivní a prediktivní řízení*. Zlín: Univerzita Tomáše Bati ve Zlíně. Akademické centrum, 2008, ISBN 978-80-7318-662-3.
- [8] *PolyX* [online]. Prague, 2001 [cit. 2016-04-30]. Available from: <http://www.polyx.com/>
- [9] Šebek, M.: *Polynomial Toolbox for MATLAB, Version 3.0*. PolyX, Prague, 2014
- [10] BOBÁL, V. *Identifikace systémů*. Zlín: Univerzita Tomáše Bati ve Zlíně. Akademické centrum, 2009, ISBN 978-80-7318-888-3.
- [11] LANDAU, Ioan D. *Digital control systems design, identification and implementation*. 1st ed. London: Springer, 2007. ISBN 978-184-6280-566.

- [12] BOBÁL, Vladimír, Petr CHALUPA, Petr DOSTÁL and Marek KUBALČÍK. Digital control of unstable and integrating time-delay processes. *International Journal of Circuits, Systems and Signal Processing* [online]. 2014, vol. 8, s. 424-432 [cit. 2016-04-28]. ISSN 1998-4464. Available from: <http://naun.org/cms.action?id=7621>.
- [13] A simple image segmentation example in MATLAB. In: *MATLABtricks* [online]. Zoltan Fegyver, 2014 [cit. 2016-05-09]. Available from: <http://matlabtricks.com/post-35/a-simple-image-segmentation-example-in-matlab>
- [14] Solving mazes with the watershed transform. In: *Mathworks Blogs* [online]. Spandan Tiwari, 2014 [cit. 2016-05-09]. Available from: <http://blogs.mathworks.com/steve/2014/01/21/solving-mazes-with-the-watershed-transform/>
- [15] The Watershed Transform: Strategies for Image Segmentation. In: *Mathworks Technical Articles* [online]. Steve Eddins, 2014 [cit. 2016-05-09]. Available from: <http://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html>
- [16] COUPRIE, Michel, Laurent NAJMAN a Gilles BERTRAND. Algorithms for the Topological Watershed [online]. s. 172 [cit. 2016-05-09]. DOI: 10.1007/978-3-540-31965-8_17. Available from: http://link.springer.com/10.1007/978-3-540-31965-8_17
- [17] The Long Island Sound Watershed. In: *The SoundBook Online* [online]. [cit. 2016-05-09]. Available from: http://soundbook.soundkeeper.org/chapter_ContentID_210_SectionID_6.htm

LIST OF FIGURES

Fig. 1 CE151 Ball & Plate model diagram [1]10

Fig. 2 Alternative Ball & Plate model [2].....12

Fig. 3 Frame pivoting layout [3] and 6DOF Stewart platform [4]13

Fig. 4 Mathematical model setup [5]14

Fig. 5 Structure of 2DOF controller.....21

Fig. 6 Measured step responses26

Fig. 7 Average of measured step responses27

Fig. 8 Identification using Recursive Least Squares method.....27

Fig. 9 Choosing the bounce point from the averaged response28

Fig. 10 Identification process.....28

Fig. 11 Distribution of identified parameters.....29

Fig. 12 Nonlinear Simulink model30

Fig. 13 Masked subsystem model30

Fig. 14 Simulated response to step31

Fig. 15 Pole-zero map of the plant.....33

Fig. 16 Ball & Plate control simulation model33

Fig. 17 Simulated step reference tracking34

Fig. 18 Simulated step disturbance rejection35

Fig. 19 Simulated step load disturbance rejection36

Fig. 20 Simulated circular reference tracking – relatively low frequency.....37

Fig. 21 Simulated circular reference tracking – relatively high frequency38

Fig. 22 Simulated circular reference tracking with sinusoidal controller design39

Fig. 23 Simulated ramp reference tracking with ramp controller design40

Fig. 24 Simulink scheme for real model control41

Fig. 25 Ball Coordinates Subsystem.....42

Fig. 26 *Selection and Scaling* subsystem with mask options42

Fig. 27 *Selector & Data Collector* subsystem42

Fig. 28 *Initialization and Scaling* (left) and its *Roll Init* subsystem (right).....43

Fig. 29 Calibration Delay subsystem43

Fig. 30 Controller subsystem43

Fig. 31 Reference Feed subsystem44

Fig. 32 Reference Feed Trigger subsystem44

Fig. 33 Comparison of models.....	45
Fig. 34 Reference tracking.....	46
Fig. 35 Disturbance rejection (blowing to the ball).....	47
Fig. 36 Circular reference tracking.....	48
Fig. 37 Maze.....	49
Fig. 38 Color perception [13].....	49
Fig. 39 Blueness picture and its mask.....	49
Fig. 40 Watershed transform steps.....	50
Fig. 41 Algorithm completion with user prompt.....	50
Fig. 42 Different paths test and looped path with warning.....	50
Fig. 43 Maze navigation – first run.....	51
Fig. 44 Maze navigation – second run.....	52
Fig. 45 Maze navigation – third run.....	53
Fig. 46 Watershed drainage [17].....	54
Fig. 47 Grayscale image and its topological relief [15].....	54
Fig. 48 Ball & Plate model GUI.....	55
Fig. 49 Additional plots in GUI with maze navigation simulation results.....	56
Fig. 50 Circle parameters user prompt and context menu content.....	56

APPENDICES

- P I Function for maze solving
- P II Function for identification
- P III Function for LQ controller design
- P IV Script for spectral factorization

APPENDIX P I: FUNCTION FOR MAZE SOLVING

```
function [refPitch, refRoll, pathOK] = mazeSolve(imgPath)
%*****
% mazeSolve.m - function to solve mazes with blue walls. The call without
%               agruments results in taking the snapshot from camera.
%
% Return:
%   refPitch  (Nx1 double)   reference values of pitch (y axis for CE151)
%   refRoll   (Nx1 double)   reference values of roll (x axis for CE151)
%   pathOK    (1x1 logical)  flag to tell if the path is OK
%
% Arguments:
%   imgPath   (string)       path to the maze image (image or MAT-file)
%
% Examples:
%   [y, x, pathOK] = mazeSolve('maze.mat');
%   [y, x, pathOK] = mazeSolve('maze.jpg');
%   [y, x, pathOK] = mazeSolve();
%
% Author: Lubos Spacek, ll_spacek@fai.utb.cz, 2016
%*****

if nargin == 0
    useCam = 1;
else
    useCam = 0;
end

f = figure('Units','Normalized','Position',[0.3 0.3 0.4 0.4],...
           'Name', 'Maze Path', 'NumberTitle','off');

% threshold value for blueness (empirical)
thresholdValue = 75;

% number of pixels to crop horizontally from each side (empirical)
cropValue = 20;

% get the video object
if useCam
    vid = videoinput('winvideo',1,'RGB24_160x120');
end

pathOK = false;

while ~pathOK && ishandle(f)

    % take a snapshot
    if useCam
        img = getsnapshot(vid);
    elseif strcmp(imgPath(end-2:end), 'mat')
        try
            S = load(imgPath);
            img = cell2mat(struct2cell(S));
        catch
            errordlg('Select maze picture stored in one matrix!');
            refPitch = 0; refRoll = 0;
            return;
        end
    end
end
```

```

else
    try
        img = imread(imgPath);
    catch
        errordlg('Select a picture!');
        refPitch = 0; refRoll = 0;
        return;
    end
end
imgSize = size(img);

% crop the snapshot
I = img(:,cropValue:imgSize(2)-cropValue+1,:); imgSize = size(I);

% create and show red image as the base for trajectory alpha mask
redImg = zeros(imgSize); redImg(:,:,1) = ones(imgSize(1:2));

% plot the image
h1 = subplot(121);
imshow(I); hold on;
h = imshow(redImg); hold off;

% get the R-G-B values
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);

% calculate the blueness of each pixel  $b = B - \max(R,G)$ 
blueness = double(B) - max(double(R),double(G));

% create a label image, where all pixels having the same value
% belong to the same object, example:
% 1 1 0 1 1 0      1 1 0 2 2 0
% 0 1 0 0 0 0      0 1 0 0 0 0
% 0 0 0 1 1 0  -> 0 0 0 3 3 0
% 0 0 1 1 1 0      0 0 3 3 3 0
% 1 0 0 0 1 0      4 0 0 0 3 0
labels = bwlabel(blueness > tresholdValue);

% choose label with maximum occurence
id = mode(labels(:));

% get the mask containing only the maze walls
maze = (labels == id);

% apply the watershed transform to get the trajectory
L = watershed(maze);

% remove all connected objects that have fewer than 120 pixels
traj = bwareaopen(L == 0, 120);

% apply the trajectory alpha mask to add red trajectory to image
set(h, 'AlphaData', traj);

% corner points north to east (empirical)
cPts = struct('N',4,'S',110,'W',7,'E',118);

% init

```

```

row = 1; col = find(traj(1,:)==1);
trajLength = sum(traj(:));
try
    walk = zeros(trajLength,2); walk(1,:) = [row col];

    % direction of movement 1-NORTH, 2-SOUTH, 3-EAST, 4-WEST
    DIR = zeros(trajLength,1);
    % get the row and column indices of the trajectory in order
    for i = 2:trajLength

        if DIR(i-1) ~= 2 && traj(row+1,col) == 1
            row = row + 1;
            DIR(i) = 1;
        elseif row > 1 && DIR(i-1) ~= 1 && traj(row-1,col) == 1
            row = row - 1;
            DIR(i) = 2;
        elseif DIR(i-1) ~= 4 && traj(row,col+1) == 1
            col = col + 1;
            DIR(i) = 3;
        elseif col > 1 && DIR(i-1) ~= 3 && traj(row,col-1) == 1
            col = col - 1;
            DIR(i) = 4;
        else
            disp('The trajectory is not continuous!');
            break;
        end

        walk(i,:) = [row col];
    end

catch
    DIR = 0;
    wH = warndlg('The path is probably looped or incomplete.', ...
        'Path Warning');
    uiwait(wH);
end

% get the corner points
walk = walk(logical(diff([DIR;1])),:);

% saturate to avoid touching the walls
walk(walk(:,1) < cPts.N+5) = cPts.N+5; % start offset
walk(walk(:,1) > cPts.S-5) = cPts.S-5; % end offset
walk(walk(:,2) < cPts.W) = cPts.W;
walk(walk(:,2) > cPts.E) = cPts.E;

% plot path
h2 = subplot(122);
imshow(I); hold on;
plot(walk(:,2),walk(:,1),'r',walk(:,2),walk(:,1),'.g');

button = questdlg('Is the path OK?', 'Continue?', ...
    'Path OK', 'Try Again', 'Quit', 'Path OK');
pathOK = strcmpi(button, 'Path OK');

if strcmpi(button, 'Quit')
    break;
end
end

```

```

    if ~pathOK
        arrayfun(@cla,[h1 h2]);
    end

end

% normalize to <-1,1> range
if pathOK
    refPitch = (walk(:,1) - cPts.N)*(-2/(cPts.S-cPts.N)) + 1;
    refRoll = (walk(:,2) - cPts.W)*(2/(cPts.E-cPts.W)) - 1;
else
    refPitch = 0; refRoll = 0;
end

% maze from maze generator solution - http://www.mazegenerator.net/
% I = imread('maze 20 by 20 orthogonal.png');
% I = logical(I(:,:,1));
% imshow(I); figure;
% L = watershed(I);
% imshow(L,[]); figure;
%
% S = L == 0;
% D = I - S;
% imshow(D);

```

APPENDIX P II: FUNCTION FOR IDENTIFICATION

```
function [X, fG] = ident_bp(fG, guess)
%*****
% ident_bp.m - function to identify measured step responses for B&P model
%
% Return:
%   X      (1xN double)   identified parameters
%   fG     (func. handle) handle for identified anonymous function
%
% Arguments:
%   fG     (func. handle) handle for anonymous function to identify
%   guess  (1xN double)   initial guesses for identification
%
% Examples:
%   [X, fG] = ident_bp(@(X) tf(X(1), [X(2) 1 0 0]), [5 0.2]);
%   X = ident_bp();
%
% Author: Lubos Spacek, ll_spacek@fai.utb.cz, 2016
%*****

% set default transfer function and initial guess
if nargin < 2
    fG = @(X) tf(X(1), [X(2) 1 0 0]);
    % fG = @(X) tf(X(1), [X(2) 1 0], 'iodelay', X(3));
    guess = [5 0.2];
end

% get the file containing identification data
[filename, pathname] = uigetfile('ident*.mat','Pick a File');
if pathname == 0
    X = []; fG = [];
    return;
end
S = load([pathname filename]);

u = S.input(:,2);
if strcmp(S.desc(1:7), 'rollOut')
    dataPos = S.rollOut;
elseif strcmp(S.desc(1:8), 'pitchOut')
    dataPos = S.pitchOut;
end

% choose only step-responses parts (where u > 0)
ud = diff([0; abs(u)>0; 0]);
sI = find(ud > 0); % start indices
len = find(ud < 0)-sI;
lenMin = min(len);
t = dataPos(sI(1):sI(1)+lenMin-1,1) - dataPos(sI(1),1);

dataY = zeros(lenMin,length(sI)); dataU = dataY;
figure('Units','Normal','Position',[0.14 0.14 0.66 0.66],...
        'Name','Identification','NumberTitle','off');

%-----
subplot(311);
for i = 1:length(sI)
    y = dataPos(sI(i):sI(i)+lenMin-1,2) - dataPos(sI(i),2);
    u = S.input(sI(i):sI(i)+lenMin-1,2);
```

```

        dataY(:,i) = y;
        dataU(:,i) = u;
        plot(t, y); hold on;
end

if any((dataU-dataU(1)) ~= 0)
    error('Input vector si not consistent.');
```

```

end
title(['Measured step responses to constant input u = ', ...
        num2str(u(1)*100), '%']);
ylabel('Position [-]'); grid on;

%-----
subplot(312);
y = mean(dataY,2);
plot(t, y, '-'); hold on;
ylabel('Position [-]'); grid on;
title('Average of measurements');
hText = text(0, 1.05, 'Choose the bounce point.',...
             'color','r','fontweight','bold');

[xm,ym] = ginput(1);    % mouse input
delete(hText);
[ind,ind] = min(abs(xm-t));
plot(t(ind), y(ind), 'ro');
y = y(1:ind); t = t(1:ind); u = u(1:ind);

%-----
subplot(313);
X = fminsearch(@(x) krit(x, fG, t, u, y,'plot'), guess);
Ts = t(2);
Gz = c2d(fG(X),Ts);
ys = lsim(Gz,u,t);
hold on; stairs(t, ys, 'r');
xlabel('t [s]'); ylabel('Position [-]'); grid on;
title('Identification','fontweight','bold');
```

APPENDIX P III: FUNCTION FOR LQ CONTROLLER DESIGN

```
function [q,r,p,refPitch,refRoll,pathOK] = controllerDesign_LQ(refTrack)
%*****
% controllerDesign_LQ.m - function to design LQ controller for B&P model
%
% Return:
%   q          (1xK double)   numerator of feedback controller in z^-1
%   r          (1xL double)   numerator of feedforward controller in z^-1
%   p          (1xM double)   denominator of controllers in z^-1
%   refPitch   (Nx1 double)   see mazeSolve.m
%   refRoll    (Nx1 double)   see mazeSolve.m
%   pathOK     (1x1 logical)  see mazeSolve.m
%
% Arguments:
%   refTrack   (string)       reference tracking option (ramp, sin, maze)
%
% Examples:
%   [q,r,p] = controllerDesign_LQ();
%   [q,r,p] = controllerDesign_LQ('sin');
%   [q,r,p,refPitch,refRoll,pathOK] = controllerDesign_LQ('maze');
%
% Author: Lubos Spacek, ll_spacek@fai.utb.cz, 2016
%*****

if nargin == 0
    refTrack = 'step';
end

refPitch = 0; refRoll = 0; pathOK = false; % init

% [X, fG] = ident_bp();

% get pre-identified parameters (also needed to get polynomial D)
% note: the polynomial D is computed using spectral factorization using
%       Polynomial Toolbox (PolyX, Ltd. - http://polyx.com/)
par = 11;
S = load('params.mat'); X = S.params{par,2}; fG = S.params{par,3};

Ts = 0.1;

% Gz = c2d(tf(-5.0706, [0.1878 1 0 0]), Ts);
Gz = c2d(fG(X), Ts);
set(Gz, 'var', 'z^-1');
[num,den] = tfdata(Gz, 'v');
temp = num2cell(den); [a1, a1, a2, a3] = temp{:};
temp = num2cell(num); [b1, b1, b2, b3] = temp{:};

% D = conv([1 -2.2774 1.7252 -0.4298],poly([0.8 0.8 0.8]));
D = conv(S.dataD{par},poly([0.8 0.8 0.8]));
% D = conv(dataD{par},poly([0.9 0.85 0.88])); % maze

A = [b3 0 0 0 -a3 0
      b2 b3 0 0 a3-a2 -a3
      b1 b2 b3 0 a2-a1 a3-a2
      0 b1 b2 b3 a1-1 a2-a1
      0 0 b1 b2 1 a1-1
      0 0 0 b1 0 1];
b = [D(7) D(6) D(5)+a3 D(4)-a3+a2 D(3)-a2+a1 D(2)-a1+1]';
```

```

res = num2cell(A\b);
[q3, q2, q1, q0, p2, p1] = res{:};
r = sum(D)/sum(num);

% reference tracking options
if strcmp(refTrack, 'sin')
    w = 2*pi*0.2;
    C = -2*cos(w*Ts);
    A = [0 0 0 0 0 1
         0 0 0 0 1 C
         0 b3 0 1 C 1
         b3 b2 1 C 1 0
         b2 b1 C 1 0 0
         b1 0 1 0 0 0];
    b = [D(7) D(6) D(5) D(4) D(3)-1 D(2)-C]';

    res = num2cell(A\b);
    [r0, r1, ~, ~, ~, ~] = res{:};
    r = [r0 r1];
elseif strcmp(refTrack, 'ramp')
    A = [0 0 0 0 0 1
         0 0 0 0 1 -2
         0 b3 0 1 -2 1
         b3 b2 1 -2 1 0
         b2 b1 -2 1 0 0
         b1 0 1 0 0 0];
    b = [D(7) D(6) D(5) D(4) D(3)-1 D(2)+2]';

    res = num2cell(A\b);
    [r0, r1, ~, ~, ~, ~] = res{:};
    r = [r0 r1];
elseif strcmp(refTrack, 'maze')
    % [refPitch, refRoll, pathOK] = mazeSolve(); % use camera
    [refPitch, refRoll, pathOK] = mazeSolve('maze.mat');
end

q = [q0 q1 q2 q3]; p = [1 p1 p2];

disp('DONE');

```

APPENDIX P IV: SCRIPT FOR SPECTRAL FACTORIZATION

```
% *****
% Spectral factorization to minimize the LQ criterion
% *****

% check for Polynomial Toolbox (PolyX, Ltd. - http://polyx.com/)
v = ver;
PT = any(strcmp(cellstr(char(v.Name)), 'Polynomial Toolbox'));
if ~PT
    % User does not have the toolbox installed.
    message = sprintf(['Sorry, but you do not seem to have the ' ...
        'Polynomial Toolbox.\nDo you want to try to continue
        anyway?']);
    reply = questdlg(message, 'Toolbox missing', 'Yes', 'No', 'Yes');
    if strcmpi(reply, 'No')
        % User said No, so exit.
        return;
    end
end

gprops zi
pformat SYMB

Gz = c2d(tf(-5.0706, [0.1878 1 0 0]), 0.1);
set(Gz, 'var', 'z^-1');
[num, den] = tfdata(Gz, 'v');
A = pol(den, length(den)-1, 'zi');
B = pol(num, length(num)-1, 'zi');

qu = 1;
D = spf(A*qu*A' + B*B');
Di = pol(D*zi^deg(D));
Di = Di/Di{0};
```