VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TELES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

# MODELOVÁNÍ, IDENTIFIKACE A ŘÍZENÍ ROBOTICKÉHO MANIPULÁTORU

MODELLING, IDENTIFICATION AND CONTROL OF ROBOTIC MANIPULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MICHAL ŠURANSKÝ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ROBERT GREPL, Ph.D.
SUPERVISOR

BRNO 2013

# Abstrakt

Táto diplomová práca sa zaoberá identifikáciou, modelovaním a riadením manipulátoru s tromi stupňami voľnosti. Diplomová práca je súčasťou projektu [17], ktorého cieľom je vytvorenie edukačnej platformy. V práci je testované riadenie modelu kyvadla pomocou PID regulátoru a pomocou PID regulátoru s doprednou kompenzáciou. Ďalej je vytvorený model jednosmerných motorov, ktoré boli použité na výrobu manipulátoru a taktiež sa vytvoril model inverznej dynamiky celého manipulátoru. Tento model sa následne použil na riadenie manipulátoru metódou riadenia s doprednou kompenzáciou. V záverečnej časti bola vytvorená aplikácia v ktorá umožňuje manipulátoru naučiť sa a neskôr vykonávať rôzne pohyby. Pre jednoduchšie ovládanie aplikácie bolo naprogramované používateľské rozhranie.



# Kľúčové slová

Model, inverzná kinematika, inverzná dynamika, riadenie, dopredná kompenzácia, motor, grafické používateľské rozhranie



# Abstract

Main aim of this master's thesis is to identify, model and control robotic manipulator with three degrees of freedom. The thesis is a part of major project [17], the aim of which is to create an educational platform. In the thesis the simple PID control and the PID with feedforward compensation control is tested on the model of simple pendulum. In the next part models of DC motors, which are used for construction of the manipulator, are developed and the inverse dynamics model of manipulator is developed. This model is used for feedforward control of the manipulator. In the final part the application was developed, which allows the manipulator to be taught some movements, which can be later on, executed. For the simple control of the application the graphical user interface was programmed.



# Keywords

Model, inverse kinematics, inverse dynamics, control, feedforward compensation, motor, graphical user interface

# Acknowledgement

I would like to thank to my supervisor doc. Ing. Robert Grepl Ph.D. for the leadership of the whole project and to Ing. Josef Vejlupek for patient answering my questions.

# Statutory declaration

With this statement I affirm, that this thesis is my genuine authorial work, which was created under the leadership of my tutor and with the support of mentioned literature.

Michal Šuranský, Brno, 2013

# Bibliographic reference

ŠURANSKÝ, M. *Modelling, identification and control of robotic manipulator*. Brno: University of Technology, Faculty of mechanical engineering, 2013. 72 pages. Tutor: Doc. Ing. Robert Grepl, Ph.D..

# Table of Contents

# 1. Introduction

Modeling and analysis enable engineers to test whether specifications are met. System can be analyzed at different levels such as for example model for design of mechatronic system, model for control synthesis, finite element based model and many more. For control synthesis a control engineering based system representation is needed. This can be either inverse dynamics model for feedforward compensation or linearization of equations of motion to obtain state space representation. In the thesis state space control is not tested.[10]

Besides feedforward control, models can be used for model based design of products. This means that engineers use an executable model to find out every bit of information for validation of the product. Model based design can significantly reduce product development costs. In addition, the model can be easily shared with other engineers, which creates a possibility to outsource research and development. Therefore creating a model of the system is meaningful in many ways for sustainable development in the future.

Control based on FFC algorithm always involves PID regulator with feedback loop. A feedback control loop is required to track required set point and to suppress unmeasured disturbances which are always present in the real world system. Even though the most of industrial applications can be controlled by simple PID regulator with feedback loop, combined feedforward plus feedback loop can significantly improve performance. In ideal situations a feedforward loop can almost entirely eliminate effect of disturbance on output of the process. Even with some model inaccuracies feedforward control is often able to reduce measured disturbance better than a simple feedback loop could achieve alone. Of course, the decision whether to use a feedforward control depends on whether the improvement in the response is good enough compared to additional development and maintenance costs. [4], [11]

Another possibility how to ensure sustainable development is to create user friendly applications. For purpose of this, many different frameworks for graphical user interfaces development were created. One of these is implemented in Matlab and is called Matlab graphical user interface – GUI. When a GUI application for machine is developed its readability and popularity rapidly increases. In this thesis a GUI application for manipulator control is developed.

# 2. Goals Description

## 2.1. Project for educational platform

This Master Thesis is based on the project ‚Platform for education of modeling, identification and control of dynamic systems [17]. The aim of the project is to create an educational platform for identification, simulation and control of three axis manipulators, which are supposed to be similar to the commercially produced ones. The base of the manipulator is an electromechanical system, which is built from common components since the aim is to create platform, which is easy to reproduce.

One of the basic tasks is dynamics requirements of the manipulator. Designed platform is supposed to be able to execute movements dynamic enough to show influence of inertia torques of parts.

Another goal is a possibility to demonstrate effects of gravity load. Therefore construction of manipulator must be designed so that joints are not self-locking.

## 2.2. Motivation for developing the manipulator

The construction is motivated by choosing final functionalities, which are supposed to be attractive for students. The manipulator construction is therefore designed such a way that the manipulator is able to write on tablet with the end effector or draw simple pictures. The manipulator should be able to do typical manipulator actions as to grasp the object and move the object.

## 2.3. Partial project tasks

1) Fundamental parameters suggestion based on simple models of manipulator (following motivation tasks)
2) Design and construction of manipulator parts
3) Design and realization of power and control electronics
4) Identification of manipulator parameters. Model estimation based on the real manipulator
5) Control design: PID control, Feedforward control
6) Application creation. Applications use the manipulator with complementary sensors such as encoders, accelerometers and vision toolbox

## 2.4. Goals of the Master Thesis

In the first part of the master thesis control with feedforward compensation is examined on the simple model of pendulum. Realization of the model and comparison of control methods is done. This is afterwards used in manipulator regulator design.

In the next part of thesis motors of manipulator and manipulator itself is modeled. Inverse Dynamics model of the manipulator is done by three different methods and the final model is extended with model of friction. Some of the parameters (masses, dimensions, inertia matrices) are gained from Solidworks software and other (friction coefficients) are estimated. The final model is used for feedforward compensation of dynamic torque.

When the feedforward compensation control is ready it is compared with 3xPID control and the comparison is evaluated.

In the final part a user friendly interface is created for the application to test manipulator control. The application is TEACH – EXECUTE application and the user interface is supposed to make it more readable and popular for the end user.

## 2.5. Team structure

In this part a distribution of partial tasks between members of development team is shown.

Josef Vejlupek
: A team leader. Project goals specification and basic suggestions.

Tomáš Ripel
: A 3D model design. Design and Construction of mechanical part of manipulator.

David Klimeš
: KLIMEŠ, D. [7] Hardware and software solution for diagnostics and safety operation for robotic manipulator. Brno: University of Technology, Faculty of Mechanical Engineering, 2013. 64 p. Master's Thesis supervisor doc. Ing. Robert Grepl, Ph.D.

# 3. Literature Review

## 3.1. Feedforward and Feedback control

### 3.1.1. Feedback control

In the process industries feedback control is widely used technique. It has certain advantages.

1. Corrective action takes place only when control variable differs from set point regardless on type of disturbance.
2. Feedback control requires only minimal knowledge of controlled plant. Basically a mathematical model is not required.

And it has certain disadvantages.

1. If it is required that PID controller is universal and robust, then each time the process conditions change, controller must be retuned.
2. Feedback control does not calculate a predictive control action to compensate a measureable or a known control disturbances.
3. It is not suitable for systems with large time constant.
4. Controlled variable must be measured online.

### 3.1.2. Feedforward control

The basic concept of feedforward control is to calculate or measure important disturbance variables and execute corrective action. [4]

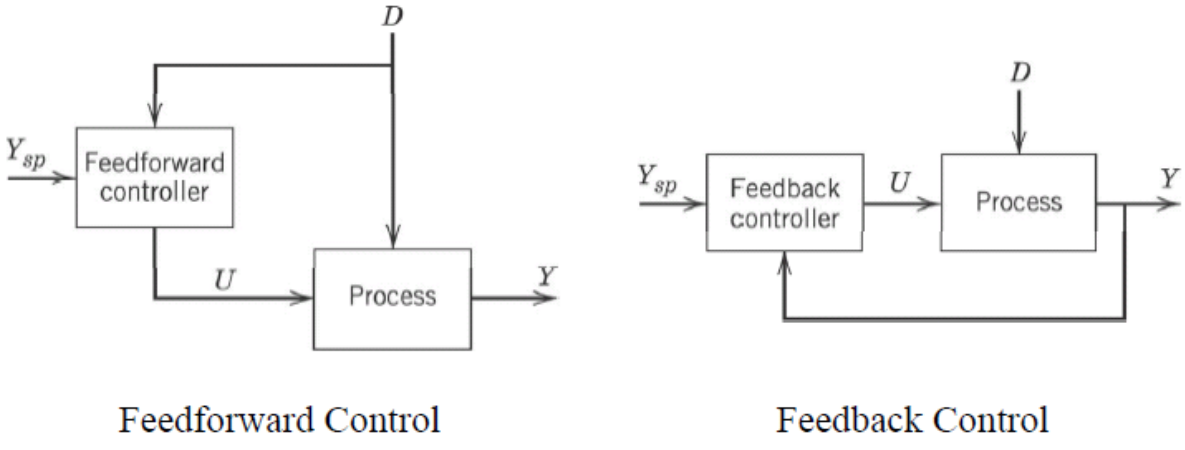



**Figure 1, Basic diagram of Feedforward and Feedback control [4]**

$Y_{sp}$ *is required value of controlled input*

$U$ *is taken action*

$D$ *represents disturbancies*

$Y$ *is system output*

Feedforward control has certain attributes.

1. To make feedforward controller work properly at least an approximate model of the process must be known. Quality of feedforward control depends on accuracy of plant‘s model. Basically it is important to know how measured variables respond to the change of controlled variables and disturbances.
2. Perfect feedforward controllers which are able of perfect control are not physically realizable. Even though approximations of these ideal controllers often provide very powerful control.

In a real world applications feedforward control is usually used in a combination with feedback control. The feedforward control is used to minimize effects of measureable disturbances. The feedback control compensates unmeasurable disturbances and inaccuracies in plant‘s model. [4]

### 3.1.3. Example of feedforward control

Common example used in system control education is inverted pendulum mounted on a motorized cart. Dynamic model of this system is needed for FFC of dynamic torque. [15] Free body diagram of inverted pendulum and cart is shown in Fig. 2. To get dynamic equations of inverted pendulum in horizontal and vertical direction following steps have to be done.

Summing the forces of the free body diagram of the cart in the horizontal direction gives equation (3.1) and summing the forces of free body diagram of the pendulum in the horizontal direction gives equation (3.2). [15]

$$M\ddot{x} + b\dot{x} + N = F \tag{3.1}$$

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{3.2}$$

Substituting equation (3.2) into (3.1) brings the first one of two governing equations of the system (3.3). [15]

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{3.3}$$

To get the second equation of motion, sum the forces perpendicular to the pendulum is made (3.4). [15]

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{3.4}$$

(3.4) is then combined with equation (3.5), which represents the sum of moments about the centroid of the pendulum and the second governing equation of the system is obtained (3.6). [15]

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{3.5}$$

$$\left(I + ml^2\right)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{3.6}$$

Control design technique in this example applies only to linear systems and therefor the set of governing equations needs to be linearized, specifically about the upward equilibrium position. In this position, which is when $\theta = \pi$. For linearization equations (3.7) are taken into consideration. [15]

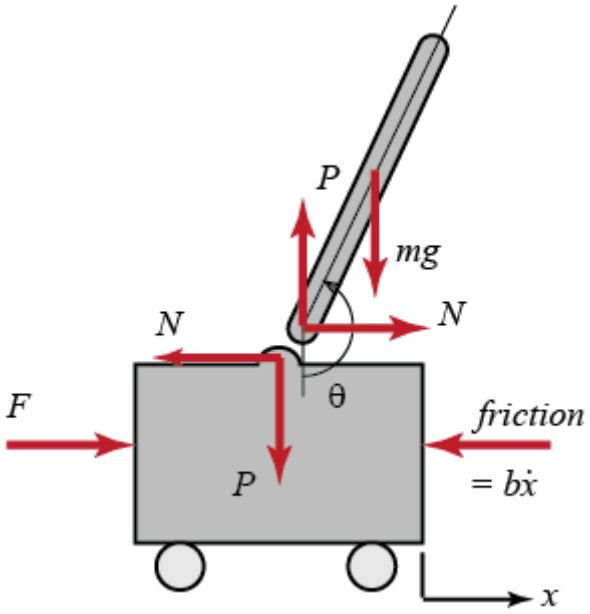



**Figure 2, Inverted pendulum model for FFC compensation, where M is mass of the cart in [kg], m is mass of the pendulum in [kg], b is friction coefficient for cart in [N/m/sec], l is length to pendulum center of mass [m], I is mass moment of inertia of the pendulum in [kgm^2], F is force applied to the cart, x is cart position coordinate, angles are in radians [15]**

$$\begin{aligned}\cos\theta &= \cos\left(\pi + \phi\right) \approx -1\\ \sin\theta &= \sin\left(\pi + \phi\right) \approx -\phi\\ \dot{\theta}^2 &= \dot{\phi}^2 \approx 0\end{aligned} \tag{3.7}$$

After approximations (3.7) are submitted into nonlinear governing equations (3.3) and (3.6), two linearized equations of motion are gained. Mark that $u$ has been substituted for the input $F$. [15]

$$\begin{aligned}(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} &= u\\ \left(I + ml^2\right)\ddot{\phi} - mgl\phi &= ml\ddot{x}\end{aligned} \tag{3.8}$$

Linearized equations of motion can be used for calculation of compensation torque around the set point in feedforward control. Control block diagram can be seen in Fig. 3.

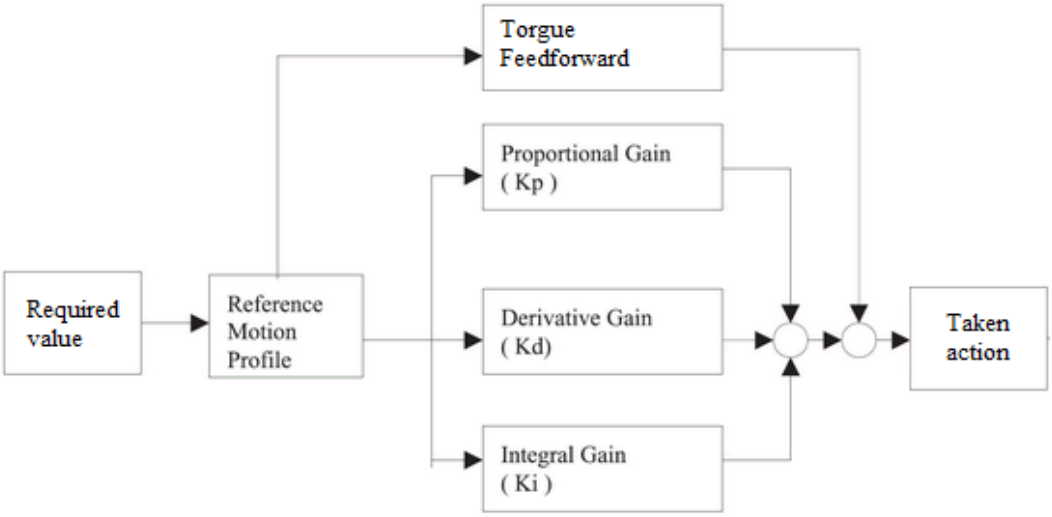



**Figure 3, Control with FFC of inverted pendulum [16]**

## 3.2. Trajectory and path planning

**Path** is a set of points, which define positions of the system on the way from a starting point to ending point. **Trajectory** is a path and for each position is defined time, when system is in the position.

### 3.2.1. Trajectory classification

Trajectories can be classified from different points of view. The first viewpoint is space of definition, which can be either Cartesian or joint. Next is point of view is the task type, which is trajectory planning or trajectory tracking. Then path geometry can be considered, which can be either rectilinear or polynomial or exponential. Last point of view is, if a coordinated or independent trajectory is planned. Trajectory is coordinated if trajectory of all joints starts and ends at the same time.

### 3.2.2. Trajectory planning in joint space

One of the basic tasks in manipulator control is to define a trajectory of the end effector from starting to desired position as a function of time. Some possible functions are cubic polynomial, fifth order polynomial, 4-3-4 polynomial and higher order polynomials. [5]

### 3.2.3. Cubic polynomial

A cubic polynomial has the form $q(t) = a + bt + ct^2 + dt^3$ and can be used to calculate trajectory only if four conditions are known. These conditions are

$$\begin{aligned} q(0) &= q_{in} & q(T) &= q_{fin} \\ \dot{q}(0) &= v_{in} & \dot{q}(T) &= v_{fin} \end{aligned} \tag{3.9}$$

When above conditions are known a system of four equations with four unknowns can be solved. [1][5] There is a problem with cubic polynomial when a second derivative is supposed to be calculated, because the second derivative is a straight-line equation. This can cause step changes in planned acceleration, which are not physically realizable.

$$\begin{aligned} q_{in} &= a + bt_{in} + ct_{in}^2 + dt_{in}^3 \\ q_{fin} &= a + bt_{fin} + ct_{fin}^2 + dt_{fin}^3 \\ v_{in} &= b + 2ct_{in} + 3dt_{in}^2 \\ v_{fin} &= b + 2ct_{in} + 3dt_{fin}^2 \end{aligned} \tag{3.10}$$

### 3.2.4. Fifth order polynomial

Fifth order polynomial has following form $q(t) = a + bt + ct^2 + dt^3 + et^4 + ft^5$ and six conditions must be defined. These conditions are equations (3.11).

$$\begin{aligned} q(0) &= q_{in} \quad & q(T) &= q_{fin} \\ \dot{q}(0) &= v_{in} \quad & \dot{q}(T) &= v_{fin} \\ \ddot{q}(0) &= a_{in} \quad & \ddot{q}(T) &= a_{fin} \end{aligned} \tag{3.11}$$

After conditions (3.11) are known a system of six equations with six unknowns can be solved (3.12). If the second derivative is done, resulting equation is a third order polynomial equation. [1][5]

$$\begin{aligned} q_{in} &= a + bt_{in} + ct_{in}^2 + dt_{in}^3 + et_{in}^4 + ft_{in}^5 \\ q_{fin} &= a + bt_{fin} + ct_{fin}^2 + dt_{fin}^3 + et_{fin}^4 + ft_{fin}^5 \\ v_{in} &= b + 2ct_{in} + 3dt_{in}^2 + 4et_{in}^3 + 5ft_{in}^4 \\ v_{fin} &= b + 2ct_{fin} + 3dt_{fin}^2 + 4et_{fin}^3 + 5ft_{fin}^4 \\ a_{in} &= 2c + 6dt_{in} + 12et_{in}^2 + 20ft_{in}^3 \\ a_{fin} &= 2c + 6dt_{fin} + 12et_{fin}^2 + 20ft_{fin}^3 \end{aligned} \tag{3.12}$$

.

### 3.2.5. Online and offline planning

Online trajectory planning is a method when the controlled variable is calculated during the program execution and the device operation. In the case of moving arm it means that the arm is being moved, while trajectory is being calculated. This means that trajectory calculation algorithm has to be fast enough. To calculate the fifth order polynomial in a real time requires demanding algorithms.

Offline trajectory planning is a method when the controlled variable is calculated before the program starts its execution. As a result of this more complicated algorithms can be used.

### 3.3. Inverse kinematics Model

Inverse kinematics model is used to calculate joint coordinates as a function of Cartesian coordinates of end effector position. A 3xDOF manipulator is relatively simple and it is possible to make inverse kinematics model analytically. It means to find three equations, each for one joint coordinate as functions of Cartesian coordinates. [1]

$$\begin{aligned} q_1 &= f(x, y, z) \\ q_2 &= f(x, y, z) \\ q_3 &= f(x, y, z) \end{aligned} \tag{3.13}$$

### 3.4. Inverse dynamics Model

Inverse dynamics model is a model of the system which calculates applied torques needed to achieve given joints positions, velocities and accelerations. Mathematically inverse dynamics model is a system of equations in the following form (3.14). [1]

$$Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) \tag{3.14}$$

Symbols in equation (3.14) are described below.

- $q, \dot{q}, \ddot{q}$ are vectors of joint coordinates, velocities and accelerations respectively
- $M(q)$ is a joint space inertia matrix. Diagonal elements in this matrix describe inertia on the joint $j$ and corresponding torque on the joint is equal to $Q_j = I_{jj}\ddot{q}_j$. Non diagonal elements describe coupling of acceleration from joint $j$ to generalized force on joint $i$. [3]
- $C(q, \dot{q})$ is Coriolis matrix. Coriolis matrix is used to calculate centripetal torques and Coriolis torques. [3]
- $F(\dot{q})$ is friction force. For most electric drives friction is after gravity load the next most dominant joint load. [3]
- $G(q)$ is gravity load, which is generally dominant term, it is present even when robot is stationary or moving slowly [3]
- $Q$ is vector of generalized actuator torques [3]

Inverse dynamics model can be calculated with several methods. Some possible ways to calculate inverse dynamics model are

1) Use Simulink-Simmechanics module to physically model a robot.
2) Describe arm robot with denavit hartenberg parameters.
3) Derive equations of motion using automated calculation method based on Lagrange equations.

Simmechanics method does not show equations of motion, but it is the least complicated and easy to set up. Denavit Hartenberg (further referred as DH) parameters is a very common way how to describe robotic systems. DH parameters method is a more complicated, but offers greater flexibility. Deriving Lagrange equations can be done either by hand or automatically using software. Lagrange equations of the second kind are one of the most used methods in analytical dynamics when constructing equations of motion for bounded systems. Progress describing calculation of Lagrange equations of the second kind in four steps is shown below. [1]

1) Select *n* independent generalized coordinates $q_i$. Considered number of an independent generalized coordinates is equal to number of system‘s degrees of freedom.

2) Form equation for kinetic and potential energy and form it to be a function of generalized coordinates and their derivatives

3) Get n equations of motion by derivation according to Lagrange equations of the second kind.

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{q}_i}\right)-\frac{\partial E_k}{\partial q_i}+\frac{\partial E_p}{\partial q_i}=Q_i \tag{3.15}$$

4) Form equations of motion into matrix to get equation (3.14)

Considering that 3xDOF manipulator is normally too complicated to be derived by hand an automatic approach is preferred. With automatic method all matrices from equation (3.14) can be calculated directly.

### 3.4.1. Calculation of matrices

Matrix $M(q)$ from equation (3.14) can be calculated using equation (3.16). [1]

$$M(q)=\sum_{i=1}^{n} m_i J_P^{(T_i)^T} J_P^{(T_i)} + J_R^{(T_i)^T} R_{i0} I_i^i R_{i0}^T J_R^{(T_i)} \tag{3.16}$$

Symbol in equation (3.16) are described below. [1]

$m_i$ is mass of i'th arm.

$J_P^{(T_i)}$ , $J_R^{(T_i)}$ are geometric Jacobians.

$R_{i0}$ is rotation matrix from system *i* to system *0*.

$I_i^i$ is inertia matrix of i'th arm in system *i* which is fixed with the arm.

Geometric jacobians for rotational joint can be calculated according to (3.16.1). [1]

$$J_{Pj}^{(T_i)} = z_{j-1} \times \left( r^{(Ti)} - r_{j-1} \right)$$
$$J_{Rj}^{(T_i)} = z_{j-1} \tag{3.16.1}$$

Symbols in equation (3.16.1) are described below. [1]

$z_{j-1}$ is axis of rotation of (j-1)'th joint.

$r^{(Ti)}$ is position vector of CoG of $i'th$ arm in coordinate system 0.

$r_{j-1}$ is position vector of origin of coordinate system (j-1) in CS 0.

Matrix $C(q,\dot{q})$ and its coefficients can be calculated according to equation (3.16.2). [1]

$$c_{ijk} = \frac{1}{2}\left( \frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right)$$
$$c_{ij} = \sum_{k=1}^{n} c_{ijk}\dot{q}_k \tag{3.16.2}$$

Symbols in equation (3.16.2) are described below. [1]

$M_{xx}$ is corresponding element of matrix $M(q)$.

$q_x$ are steering angles.

$c_{ijk}$ are Christoffel symbols.

$c_{ij}$ are corresponding elements of matrix $C(q,\dot{q})$.

Matrix $G(q)$ can be calculated according to equation (3.16.3), which calculates i'th element of the matrix. [1]

$$g_i = -\sum_{j=1}^{n} m_j g_0^T j_{P_i}^{(T_j)} \tag{3.16.3}$$

Symbols in equation (3.16.3) are described below. [1]

$g_0$ is vector of gravitational acceleration and is equal to $g_0 = [0,0,-g]$.

$m_j$ is weight of j'th arm.

$j_{P_i}^{(T_j)}$ are jacobians as shown in calculation of matrix $M(q)$.

### 3.4.2. Friction

For rotating machinery or gearboxes a characteristics of friction torque versus speed is similar to that shown in Fig. 5. On the zero speed we observe stiction effect. Value of the torque has to be over a certain level in order to the rotation can start. Once the rotation has started, stiction torque decreases and viscous friction dominates. Viscous friction (straight and dotted line) is normally modeled by equation (3.17). [3]

$$F(\dot{q}) = B\dot{q} + Q_C \tag{3.17}$$

In (3.17) B is viscous friction coefficient, which represents slope of the line. Offset $Q_C$ is Coulomb friction coefficient. In general the friction value depends on direction of rotation, but this is more due to coulomb friction than to viscous friction. Viscous friction is generally a constant, in some cases provided by motor manufacturer. [3]

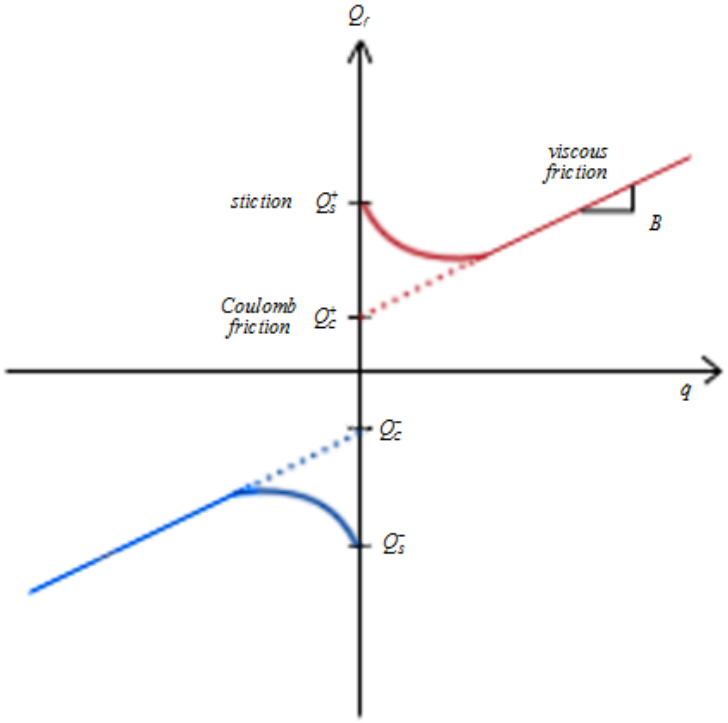



**Figure 4, Dependence of friction torque vs. speed [3]**

When friction coefficients are known, then equation (3.17) is simply added to equation (3.14).

## 3.5. Methodology for inverse dynamics model development

3xDOF manipulator is modeled by all three different ways without friction. To calculate motion equations in form of equation (3.14) a matrix method based on kinematic description and Lagrange equations of the second kind is used. Using kinematic description is possible to calculate velocities. Using geometric Jacobian, Inertia matrices, friction coefficients and masses of links it is possible to calculate matrices ***M, C, F, G*** from equation (3.14). [1] Than gained equations of motion will be used in feedforward control to drive the manipulator. Because the friction is ignored, it is assumed that the movement will be too ‚weak'. Then

friction coefficients will be experimentally tuned to get better robot movement based only on model.

## 3.6. DC motor modeling

### 3.6.1. Model description

To identify system ,engine and gear we need to know every constant in the equation (3.18), which represent mechanical equation of the system.

$$I\ddot{\varphi} = m_e - b\dot{\varphi} - Tsign\left(\dot{\varphi}\right) + M_1 \tag{3.18}$$

In equation (3.18) $I$ is rotor inertia, $m_e$ is electric torque of drive, $b$ is viscous friction constant, $T$ is Coulomb friction constant, $M_1$ is external torque load and $\dot{\varphi}, \ddot{\varphi}$ are angular velocity and acceleration respectively. The schematic representation of DC motor is given in Fig. 5. For DC motor in steady state, the current $i_s$ is constant and the torque $M_S$ generated at the shaft is given by (3.19).

$$M_S = k_\phi i_s \tag{3.19}$$

In (3.19) $k_\varphi$ is motor torque constant. DC motor voltage induced in the armature due to armature rotation is proportional to speed of rotation. This relationship can be described by equation (3.20).

$$v_s = c_\phi \dot{\varphi} \tag{3.20}$$

In equation (3.20) $c_\varphi$ is motor voltage constant. Both friction constants depend only on angular velocity of the motor shaft.

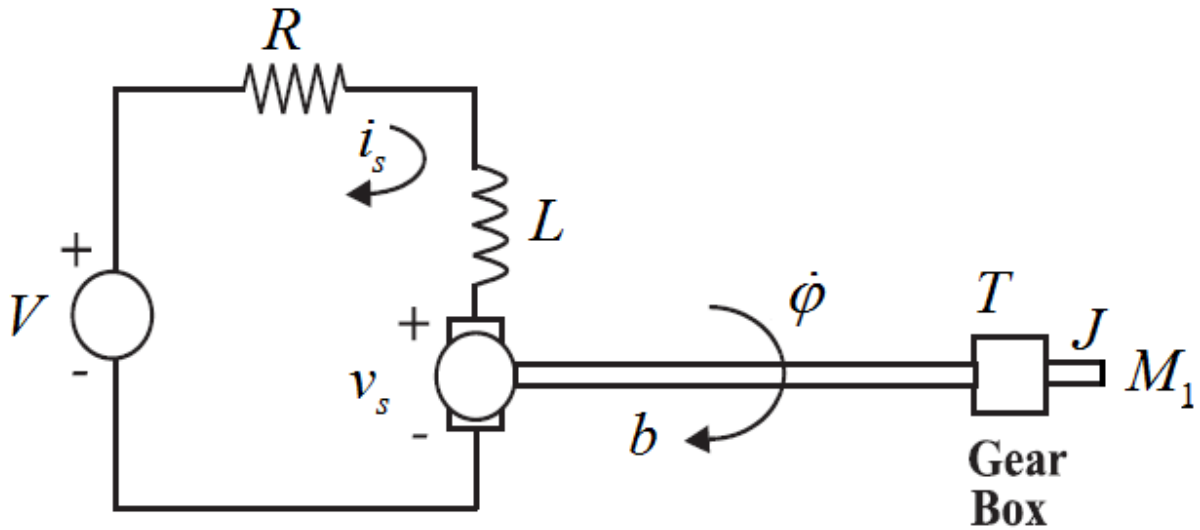



**Figure 5, Schematic representation of DC motor [6]**

## 3.7. Methodology for drive modeling

Constants $c_\varphi$ and $k_\varphi$ will be calculated according to (3.20) and (3.19). Friction constants will be calculated according to formula (3.17) and the fact that they are affected only by velocity of rotation. Inertia of rotor is fitted according to transient state measurements.

## 3.8. Communication between Matlab GUI and Simulink

There are more possibilities how to set communication between a Matlab graphical user interface and Simulink model. Some of these are:

1) Change model parameter with SET_PARAM commands.
2) Use Matlab S-Function.
3) Use Simulink event listeners.

Each of these approaches connects GUI to Simulink model differently. [12]

### 3.8.1. SET_PARAM API approach

Some of the basic attributes of these approaches are listed below.

1) It does not require another Simulink block.
2) It needs callback function to be programmed.
3) It allows only limited access to signal values during simulation.

The control of parameters and file exchange from GUI to Simulink blocks is handled by SET_PARAM command in Matlab GUI m-file.

```
set_param([bdroot '/Manual Switch'],'Manual Switch',value)
```

And control form Simulink blocks to Matlab GUI by block's callback functions. [13]

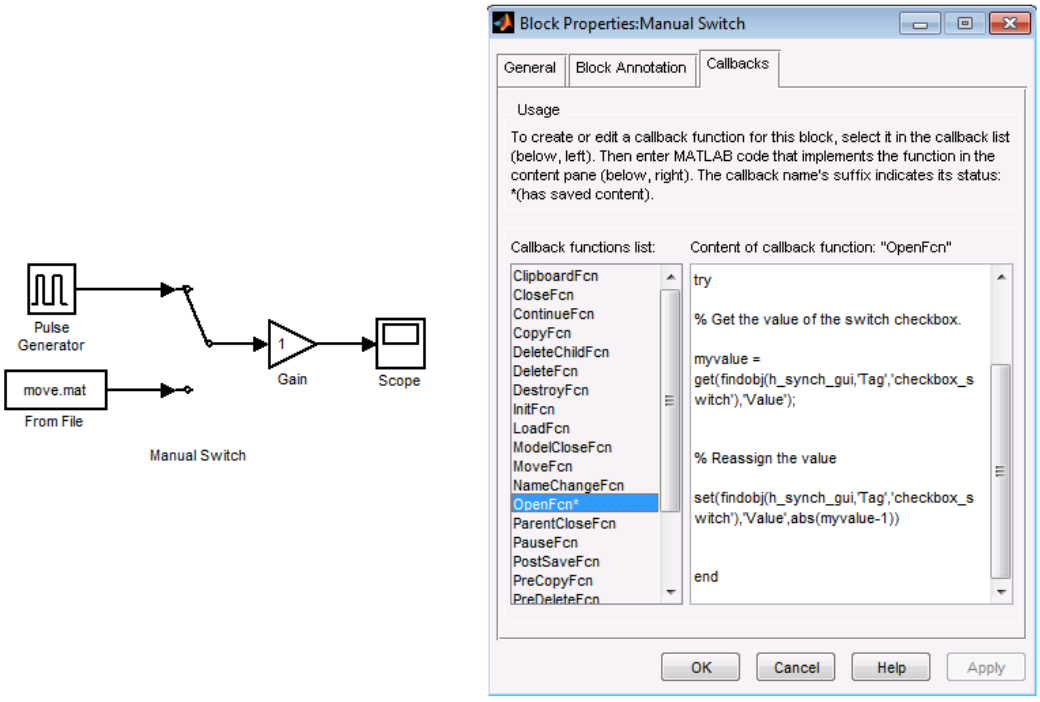



**Figure 6, Block properties and callbacks [13]**

### 3.8.2. Alter Matlab S-Function

In this method GUI is programmed as Simulink sink or source block using S-Function. Execution of the GUI process occurs during Simulink simulation.

Some of the basic attributes of this approach are listed below.

1) It does require another Simulink block.
2) It only sometimes needs callback function to be programmed.
3) It allows access to signal values during simulation.

Each figure has a property called ‚*Userdata*‘, which keeps all stored information even after figure is closed and each Simulink block has a field in Object Parameters called ‚*Userdata*‘, which keeps all stored information even when block is closed. The figures have a handle called ‚*hObject*‘, by means of which all figure properties can be accessed and each Simulink block has a handle by which all block properties can be accessed. Following figure shows how the data flow takes place. [14]

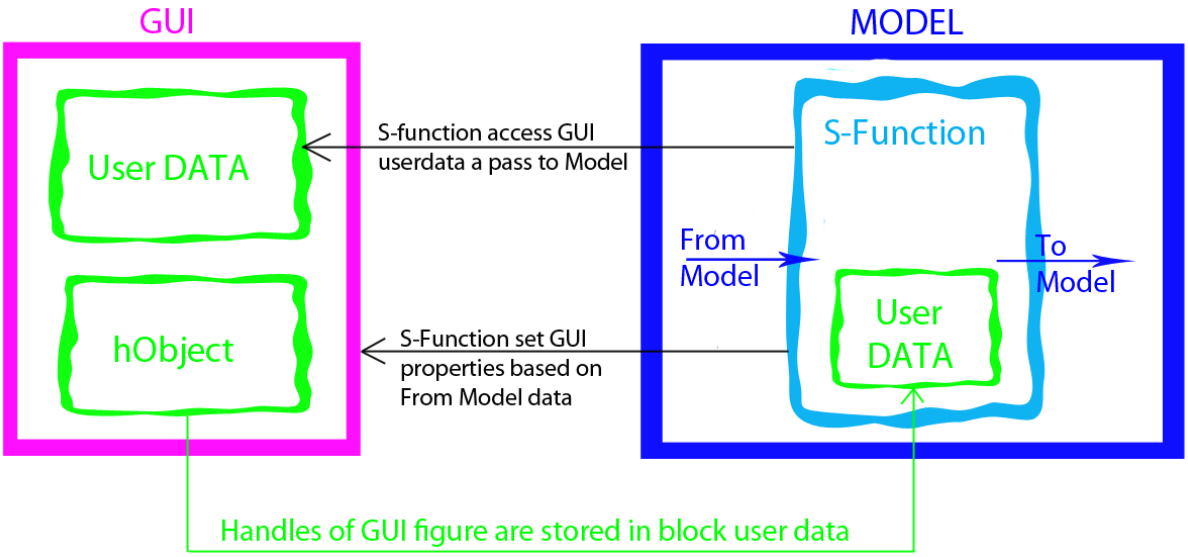



**Figure 7, S-Function data flow diagram [14]**

### 3.8.3. Use Simulink event listeners

Event listeners execute a function when actual event in the model happens. This method allows setting very tight connection between GUI and model without the need to program callback functions. However to set up event listeners is much more complicated than other two methods. [12]

## 3.9. Methodology to set up GUI and Simulink model communication

SET_PARAM commands are used, because they provide higher flexibility for the application.

# 4. Testing of feedforward compensation

## 4.1. Testing Device

The motor used to drive the FFC testing device is Transmotec DC motor. Detailed of the motor description is shown in Table 1. On the motor a universal testing holder is fastened. This can be seen in Fig. 8. After holder is fastened on the motor, whole system is fastened in the vise.

| Testing Motor parameters | |
|---|---|
| Brand | Transmotec |
| Model | WLD4383-OE |
| Voltage | 24 V |
| No Load | 4000 RPM |
| Gear Ratio | 1/31 |

**Table 1, Testing motor parameters**


**Figure 8, Testing device: 1 - holder, 2 - vise, 3 - DC motor**

## 4.2. Control without compensation

At first a control without compensation is set up and a PID controller is programmed in Simulink. For a real time communication with the testing device MF 624 card is used. Required position is calculated offline and stored as a structure. Simulink model can be seen in Fig. 9.

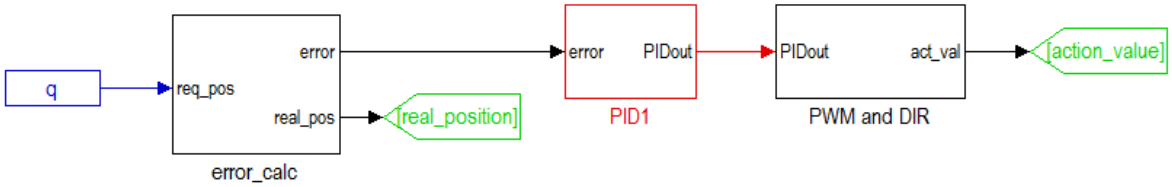



**Figure 9, Control without compensation**

PID constants are tuned experimentally and used with same values in all tests. An unusual is that D constant is ten times bigger than I constant. The constants can be seen in the table 2. When constants are tuned the testing movement can be chosen.

| PID constants | |
|---|---|
| P | 10 |
| I | 0,03 |
| D | 0,3 |

**Table 2, PID Constants for testing device**

The Testing movement is prepared to start from zero position, what is when the holder points down to the ground and to continue up to position one radian in turn, then two and three radians and back. The accuracy of control only with PID regulator can be seen in the Fig. 10.

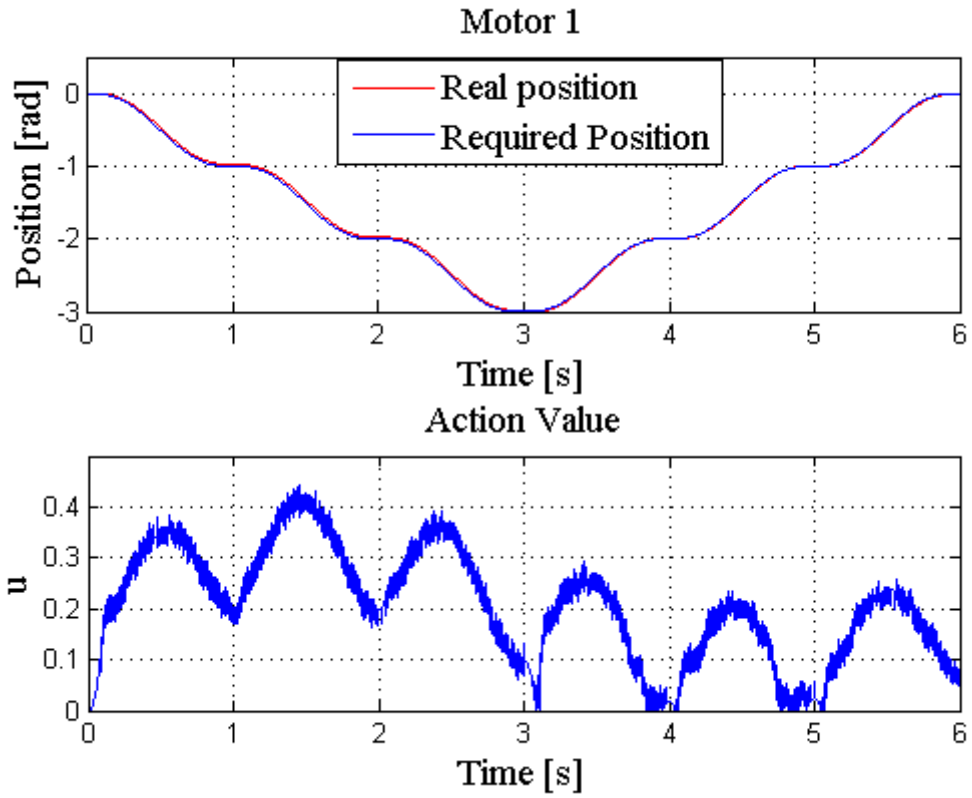



**Figure 10, Control with PID regulator only**

## 4.3. Control with static compensation

A static compensator is set to compensate gravity load of the holder. Its Simulink implementation can be seen in the Fig. 11.

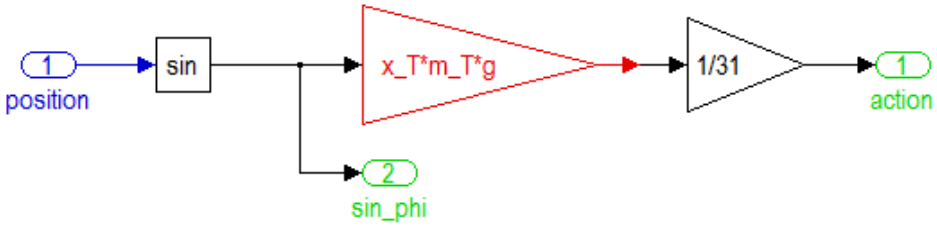



**Figure 11, Static compensator Simulink model**

Unknown constants in Fig. 11 are described in (4.1).

$$\begin{aligned} g &= 9{,}81\, ms^{-2} \\ m\_T &= 0{,}884\, kg \\ x\_T &= 0{,}075\, m \end{aligned} \tag{4.1}$$

Expression x_T is the x coordinate of the Center of Gravity and the gain block with value 1/31 reduces compensation torque in ratio of gear constant. This compensation is able to keep the holder in the set position without any other effort. Its connection to PID regulator is shown in the Fig. 12.

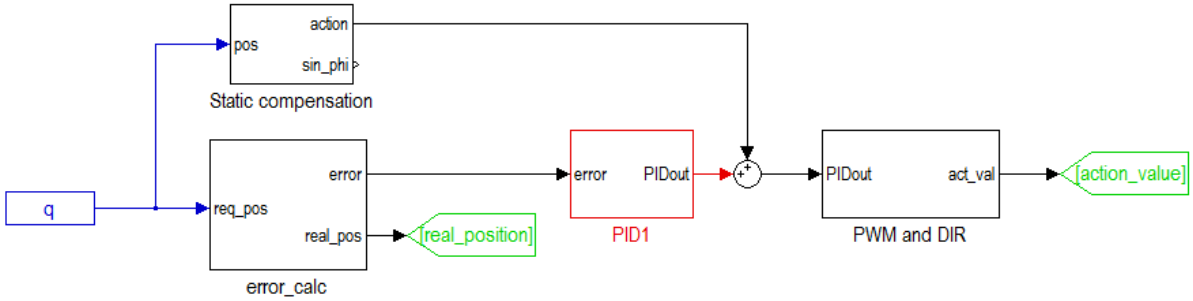



**Figure 12, Control with static compensation**

Testing movement is the same as for control without compensation and the accuracy of control with static compensation is shown in Fig. 16.

## 4.4. Control with dynamic compensation

To create a dynamic compensator the static compensator is extended with compensation of viscous friction and compensation of inertial torque. A structure of dynamic compensator can be seen in the Fig. 13. The inertia of holder is calculated with Solidworks software and viscous friction coefficient $b$ is found out experimentally.

$$\begin{aligned} Inertia &= 0{,}0044\, kgm^2 \\ b &= 5{,}5\, \frac{Nm}{rads^{-1}} \end{aligned} \tag{4.2}$$

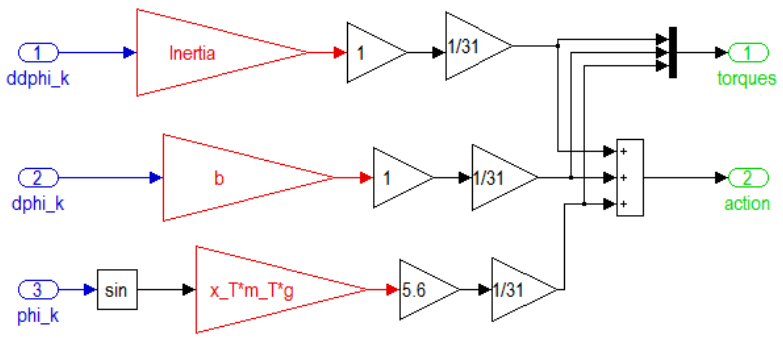



**Figure 13, Dynamic compensator structure**

When dynamic compensator is being tuned, there is a problem when the holder moves in the direction of gravity. Therefore effect of static compensation is strengthened by a factor of 5,6 so that movement shown in Fig. 14 is achieved. Movement shown in Fig. 14 is based only on the model without control. After the satisfactory accuracy of the model is achieved, the dynamic compensator is connected to the PID regulator as shown in Fig. 15. Accuracy of the movement with dynamic compensator can be seen in Fig. 17.

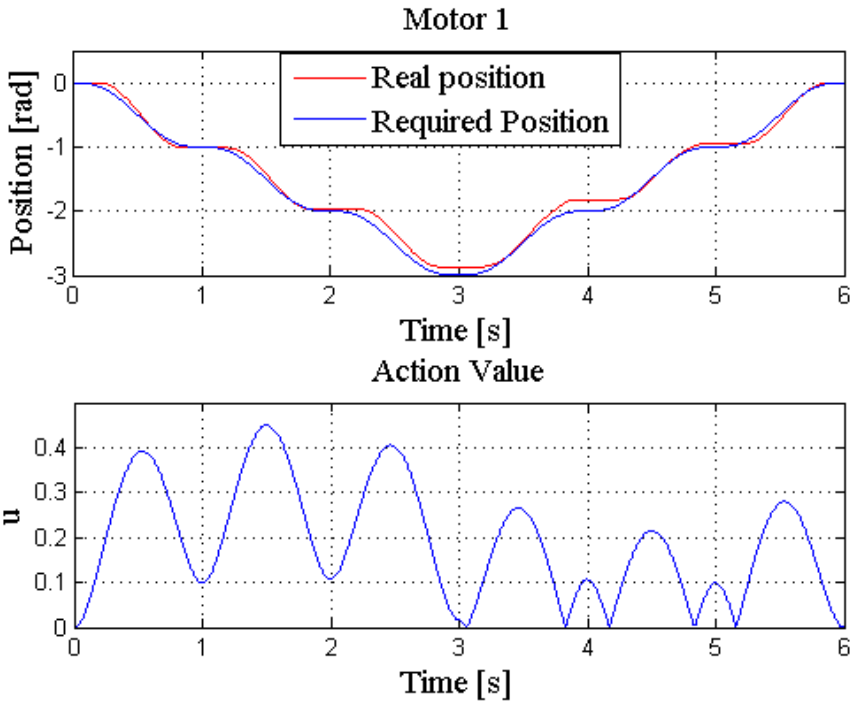



**Figure 14, Movement based only on dynamic compensator**

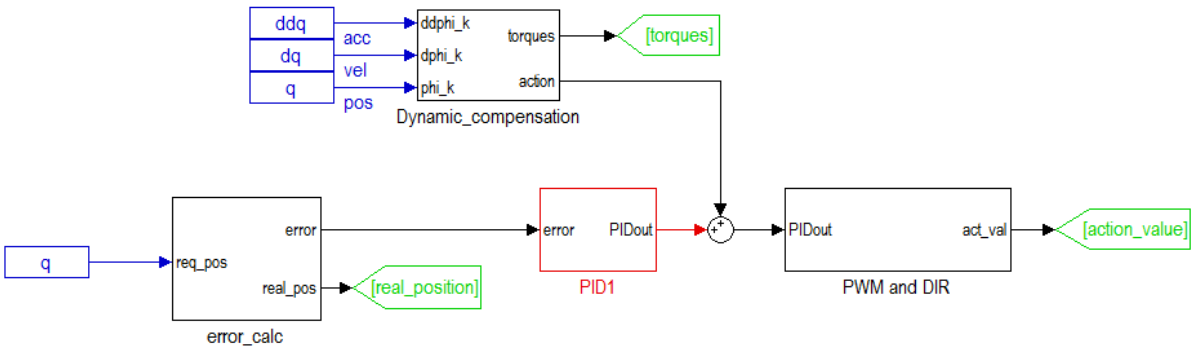



**Figure 15, Control with dynamic compensator**

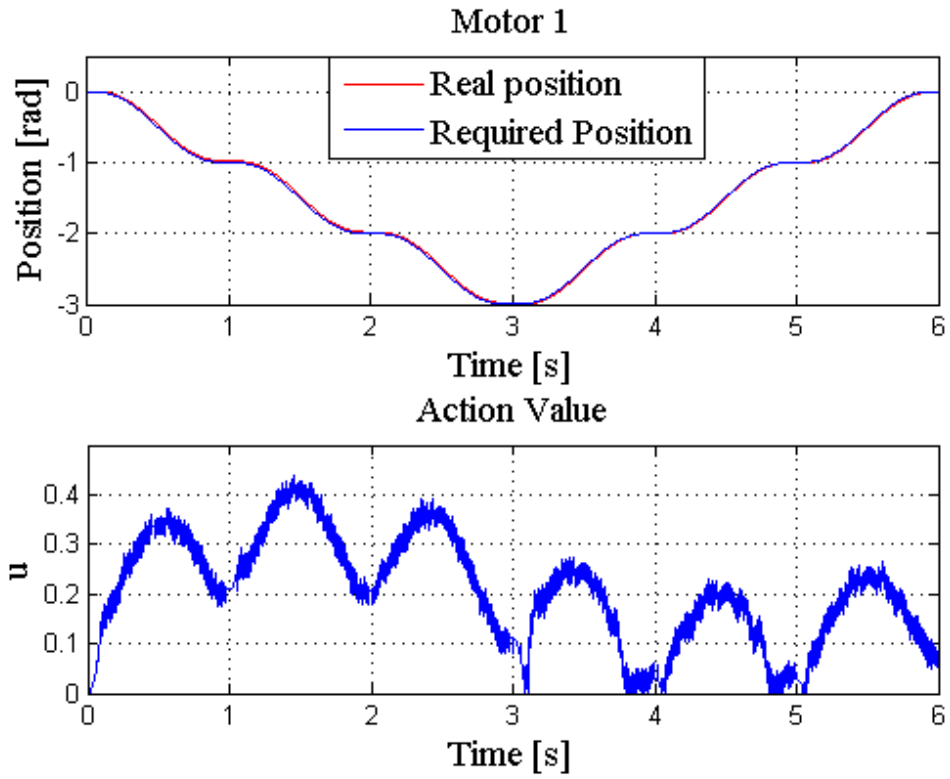



**Figure 16, Control with static compensation**

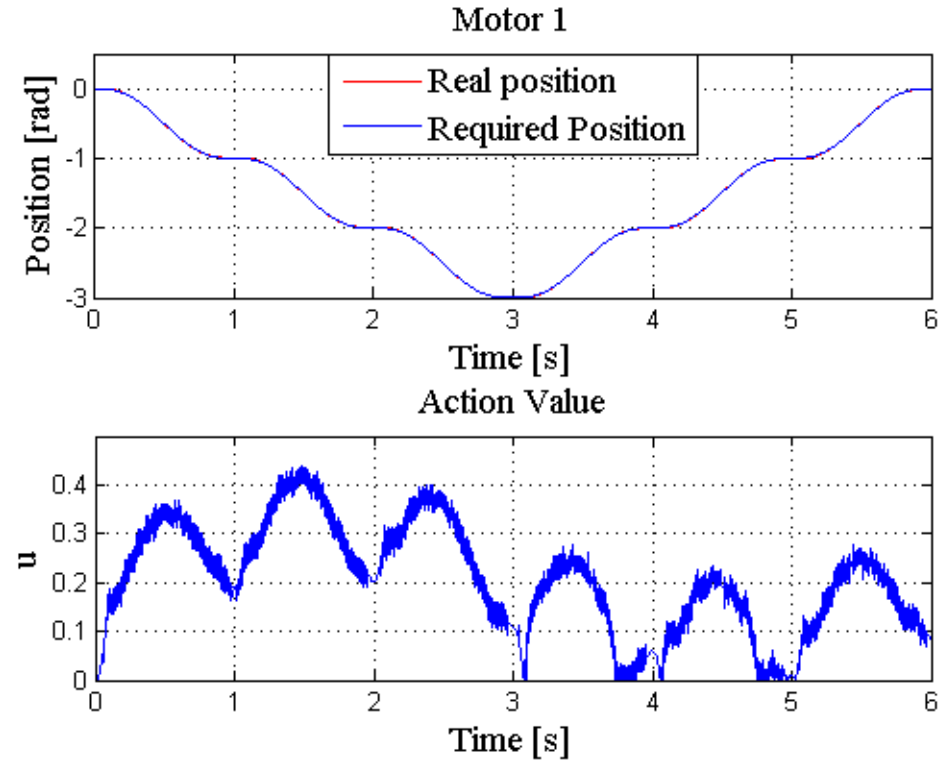



**Figure 17, Control with dynamic compensation**

## 4.5. Comparison of control types

To compare accuracy of movements regulated with pure PID control, with PID control and static compensation and with PID control and dynamic compensation ten movements are recorded for each type of control. Mean square error is calculated for each movement and then overall mean square error is calculated. Results are shown in table 3.

| | MSE in position | | |
|---|---|---|---|
| Regulator | PID | PID with Static Compensation | PID with Dynamic Compensation |
| Measurement 1 | 5,801E-04 | 5,008E-04 | 3,376E-05 |
| Measurement 2 | 5,801E-04 | 4,985E-04 | 3,398E-05 |
| Measurement 3 | 5,805E-04 | 4,934E-04 | 3,455E-05 |
| Measurement 4 | 5,848E-04 | 4,979E-04 | 3,508E-05 |
| Measurement 5 | 5,881E-04 | 5,014E-04 | 3,433E-05 |
| Measurement 6 | 5,898E-04 | 5,017E-04 | 3,290E-05 |
| Measurement 7 | 5,907E-04 | 5,012E-04 | 3,222E-05 |
| Measurement 8 | 5,967E-04 | 4,993E-04 | 3,056E-05 |
| Measurement 9 | 5,964E-04 | 5,038E-04 | 3,015E-05 |
| Measurement 10 | 5,938E-04 | 5,019E-04 | 3,003E-05 |
| **Mean MSE** | **5,881E-04** | **5,003E-04** | **3,276E-05** |

**Table 3, Comparison of movements' accuracies in squared radians**

As can be seen in table 3 the biggest error is produced by pure PID regulator. Error is slightly smaller with static compensation and markedly smaller with dynamic compensation of torque applied to motor. This result proves that when model is modeled correctly, it is possible to achieve better accuracy of movement.

But sadly the biggest advantage of FFC, what is ability to avoid oscillations when system gets unstable could not be demonstrated, because the motor used in the test has too big gear ratio and is to powerful. Therefore it is impossible to make the system oscillate. This proves that for the system **the PID control is absolutely sufficient**. It the test the same motor is used as the one used in manipulator as motor P3. But on the manipulator is the motor much less loaded, what indicates that the motor is going to be stable on the manipulator as well.

# 5. Model of the motors and the Manipulator

## 5.1. Model and parameter estimation of motor P1 and P2

Based on equations (3.18), (3.19) and (3.20), following experiment to obtain a guess of constant $k_\varphi$ is done. A pole of length 720 mm is fastened on the rotor and the motor is fixed to the vise. Different voltages are applied on the motor and forces at the end of the pole are measured. Rotor is not turning, so all elements in equation (3.18) which are dependent on $\dot{\varphi}$ or $\ddot{\varphi}$ are equal to zero. Considering the fact the motor is not turning equations (5.1) and (5.2) can be written.

$$0 = k_\Phi i + M_1 \tag{5.1}$$

$$k_\Phi = \frac{M_1}{i} \tag{5.2}$$

Average guess of constant $k_\Phi$ is equal to **0,905**. Calculation of the guess is shown in table 4.

| Calculation of kphi constant for drive P1 and P2 | | | | |
|---|---|---|---|---|
| I [A] | press [g] | press [kg] | m [Nm] | $k_\Phi$ [Nm/A] |
| 2,1 | 272 | 0,272 | 1,921 | 0,915 |
| 2,2 | 285 | 0,285 | 2,013 | 0,915 |
| 4,6 | 550 | 0,55 | 3,885 | 0,845 |
| 5,2 | 690 | 0,69 | 4,874 | 0,937 |
| 6,9 | 900 | 0,9 | 6,357 | 0,921 |
| | | | **Average** | **0,907** |

**Table 4, Calculation of $k_\Phi$ constant for drive P1 and P2**

Next step is to define constants of viscous and coulomb friction b and T. Therefore position and speed of the turning motor with no load are measured. Motor is turning at a constant speed, what means that elements dependent on $\ddot{\varphi}$ equal to zero. So from the equation (3.18) the following equation is obtained.

$$0 = k_\Phi i - b\dot{\varphi} - Tsign(\dot{\varphi}) \tag{5.3}$$

Sign function is function which changes its sign with respect to direction of rotation. Its sign must always act in opposite manner to electric moment. If we consider, that motor is turned only in one direction, sign function can be omitted and simply its sign is minus. So the equation (5.3) alters to (5.4).

$$0 = k_\Phi i - b\dot{\varphi} - T \tag{5.4}$$

Now we need to measure at two different voltage levels to obtain coefficients b and T. In equations (5.5) measured velocities from motor rotation with no load and corresponding torques are substituted and so coefficients b and T can be calculated according to (5.6) and (5.7). These can be calculated with any variation of two measurements. Done calculations can be seen in table 5.

$$M_1 = b\omega_1 + T$$
$$M_2 = b\omega_2 + T \tag{5.5}$$

$$b = \frac{(M_1 - M_2)}{(\omega_1 - \omega_2)} \tag{5.6}$$

$$T = M_2 - b\omega_2 \tag{5.7}$$

| Calculation of friction constant b and T for drives P1 and P2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| w [rads-1] | | I[A] | | U[V] | M[Nm] | | b[Nm/rad/s] | T [Nm] |
| Equation system 1 | | | | | | | | |
| 14,50 | difference | 1,05 | difference | 18 | 0,950 | difference | 0,032 | 0,486 |
| 21,30 | 6,80 | 1,29 | -0,24 | 26 | 1,167 | 0,217 | | |
| Equation system 2 | | | | | | | | |
| 12,80 | difference | 0,99 | difference | 16 | 0,896 | difference | 0,029 | 0,477 |
| 16,20 | 3,40 | 1,10 | -0,11 | 20 | 0,996 | 0,100 | | |
| Equation system 3 | | | | | | | | |
| 12,80 | difference | 0,98 | difference | 16 | 0,887 | difference | 0,035 | 0,468 |
| 9,40 | -3,40 | 0,85 | 0,13 | 12 | 0,769 | -0,118 | | |
| Equation system 4 | | | | | | | | |
| 17,90 | difference | 1,17 | difference | 22 | 1,059 | difference | 0,032 | 0,486 |
| 21,30 | 3,40 | 1,29 | -0,12 | 26 | 1,167 | 0,109 | | |
| | | | | | | **Average** | **0,032** | **0,479** |

**Table 5, Calculation of friction constants of drive P1 and P2**

From equation (3.18) we see that we need only one more coefficient and it is inertia of rotor I. This coefficient is going to be guessed by trying different values until calculated curve fits measured curve. Of course it is needed to measure curve while motor accelerates. In equation (3.18) function sign is going to be omitted again and torque $M_1$ is going be zero. Equation (5.8) is obtained.

$$I\ddot{\varphi} = m_e - b\dot{\varphi} - T \tag{5.8}$$

If we think of electric torque according to equation (3.18), it is needed to calculate $c_\varphi$ constant in order to be able to calculate current continuance. To calculate $c_\varphi$ from measurements of motor rotation following formulas can be used:

$$u = Ri + c_\Phi \dot{\varphi} \tag{5.9}$$

$$c_\Phi = \frac{u - Ri}{\dot{\varphi}} \tag{5.10}$$

In equations (5.9) and (5.10) R is rotor resistance and is equal to $0{,}8\Omega$. Results of calculation of $c_\varphi$ are shown in table 6.

| cphi constant calculation | | | | |
|---|---|---|---|---|
| w [rads-1] | I[A] | U[V] | R [ohm] | $c_\Phi$ [V/rad/s] |
| 9,40 | 0,85 | 12,00 | 0,80 | 1,204 |
| 12,80 | 0,98 | 16,00 | | 1,189 |
| 14,50 | 1,05 | 18,00 | | 1,183 |
| 16,20 | 1,10 | 20,00 | | 1,180 |
| 17,90 | 1,17 | 22,00 | | 1,177 |
| 21,30 | 1,29 | 26,00 | | 1,172 |
| | | | **Average** | **1,184** |

**Table 6, Calculation of $C_\Phi$ constant of drive P1 and P2**

Current flow is described by equation (5.11).

$$i = \frac{u - c_\Phi \dot{\varphi}}{R} \tag{5.11}$$

$$m_e = k_\Phi i = k_\Phi \frac{u - c_\Phi \dot{\varphi}}{R} \tag{5.12}$$

Using equation (5.11) the equation (5.12) can be altered into final form (5.13). Motor can be simulated in Simulink using equation (5.13).

$$\ddot{\varphi} = k_\Phi \frac{u - c_\Phi \dot{\varphi}}{I \cdot R} - \frac{b\dot{\varphi}}{I} - \frac{T}{I} \tag{5.13}$$

Best inertia guess for motors P1 and P2 is **0.4 kgm$^2$**. Calculated parameters are shown in table 7 and comparison of real and simulated run ups is shown in Fig. 18. To improve model accuracy all calculated guesses of parameters are improved in parameter estimation toolbox in Simulink and more accurate results are achieved. Simulated movement with more precisely estimated parameters is shown in figure 19. Improved estimates of parameters are show in table 7. To check the quality of the model, steady state speeds of the motor at different voltage levels are compared to steady state speeds of rotation at same voltage levels achieved by the model. Comparison can be seen in table 7 on the right.

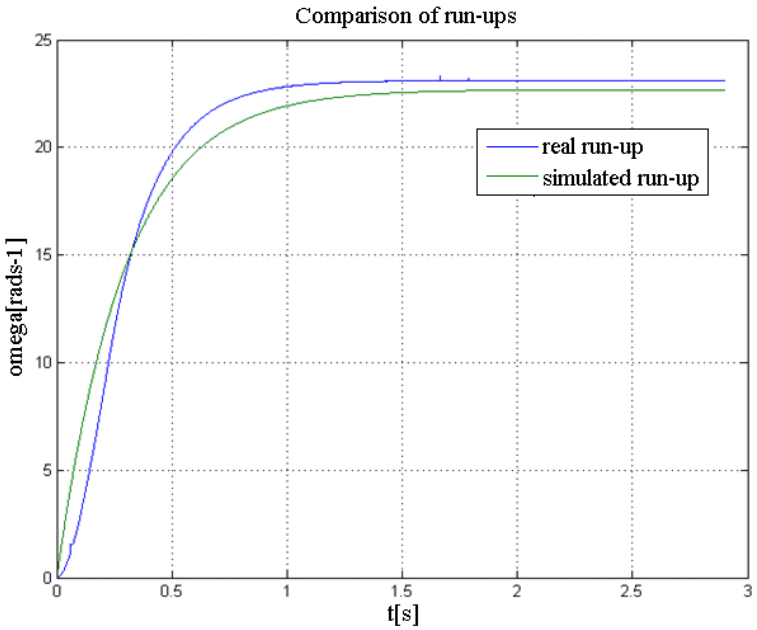



**Figure 18, Comparison of simulated and real drive run-ups of P1 and P2**

| P1, P2 | | |
|---|---|---|
| | original | altered |
| kfi [Nm/A] | 0,905 | 1,07 |
| b [Nm/rads-1] | 0,032 | 0,025 |
| T [Nm] | 0,47 | 0,38 |
| cfi [V/rads-1] | 1,18 | 1,18 |
| I [kgm2] | 0,4 | 0,45 |

| | Speed of Rotation [rad/s] | |
|---|---|---|
| Voltage [V] | meassured | simulated |
| 14 | 11 | 11,2 |
| 18 | 14,5 | 14,6 |
| 22 | 17,9 | 18 |
| 26 | 21,3 | 21,3 |
| 28 | 23,2 | 23 |

**Table 7, Comparison of original and altered parameters on the left and comparison of measured and simulated speed of rotation on the right, More precise estimated parameters are used for comparison.**

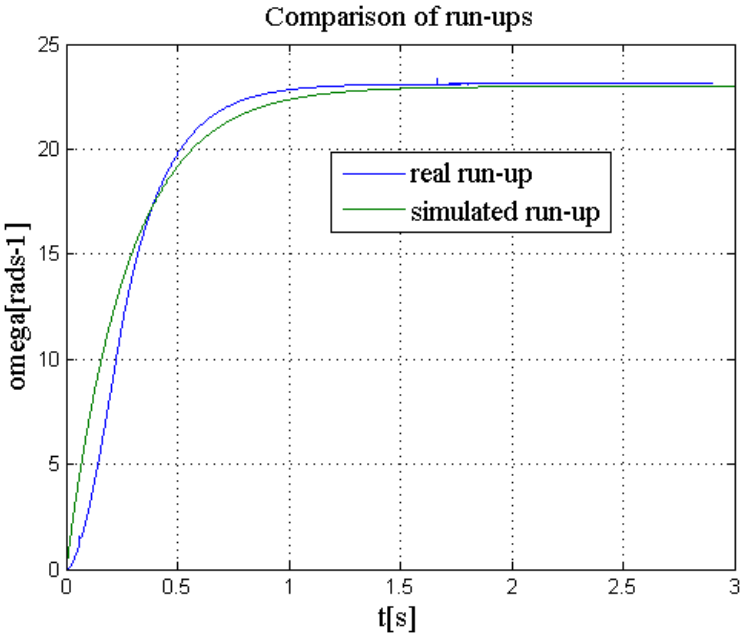



**Figure 19, Comparison of real and simulated run-ups with altered parameters**

## 5.2. Model and parameter estimation of motor P3

To obtain $k_\varphi$ this time a pole of length 240 mm is fastened on the rotor. According to equations (5.1) and (5.2) five measurements are done at voltage level 10V. Results can be seen in table 8.

| Calculation of $k_\Phi$ constant for drive P3 | | | | |
|---|---|---|---|---|
| I [A] | press [g] | press [kg] | m [Nm] | $k_\Phi$ [Nm/A] |
| 1,33 | 1023 | 1,023 | 2,409 | 1,811 |
| 1,42 | 1050 | 1,05 | 2,472 | 1,741 |
| 1,34 | 989 | 0,989 | 2,329 | 1,738 |
| 1,35 | 1020 | 1,02 | 2,401 | 1,779 |
| 1,35 | 1050 | 1,05 | 2,472 | 1,831 |
| | | | **Average** | **1,780** |

**Table 8, Calculation of $k_\Phi$ constant for drive P3**

To define friction coefficients, measurements at ten different voltage levels are done. An average b and T coefficients are calculated. Calculation of b and T constants according to equations (5.5), (5.6) and (5.7) is shown in table 9.

| Calculation of friction constant b and T for drives P3 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| w [rads-1] | | I[A] | | U[V] | M[Nm] | | b[Nm/rad/s] | T [Nm] |
| Equation system 1 | | | | | | | | |
| 2,70 | difference | 0,18 | difference | 6 | 0,320 | difference | 0,028 | 0,281 |
| 3,95 | 1,25 | 0,20 | -0,02 | 8 | 0,356 | 0,036 | | |
| Equation system 2 | | | | | | | | |
| 5,10 | difference | 0,22 | difference | 10 | 0,392 | difference | 0,015 | 0,290 |
| 6,30 | 1,20 | 0,23 | -0,01 | 12 | 0,409 | 0,018 | | |
| Equation system 3 | | | | | | | | |
| 6,30 | difference | 0,23 | difference | 12 | 0,409 | difference | 0,014 | 0,284 |
| 7,55 | 1,25 | 0,24 | -0,01 | 14 | 0,427 | 0,018 | | |
| Equation system 4 | | | | | | | | |
| 7,55 | difference | 0,24 | difference | 14 | 0,427 | difference | 0,015 | 0,280 |
| 8,70 | 1,15 | 0,25 | -0,01 | 16 | 0,445 | 0,018 | | |
| Equation system 5 | | | | | | | | |
| 8,70 | difference | 0,25 | difference | 16 | 0,445 | difference | 0,015 | 0,275 |
| 9,90 | 1,20 | 0,26 | -0,01 | 18 | 0,463 | 0,018 | | |
| Equation system 6 | | | | | | | | |
| 9,90 | difference | 0,26 | difference | 18 | 0,463 | difference | 0,016 | 0,272 |
| 11,00 | 1,10 | 0,27 | -0,01 | 20 | 0,481 | 0,018 | | |
| Equation system 7 | | | | | | | | |
| 11,00 | difference | 0,27 | difference | 20 | 0,481 | difference | 0,015 | 0,267 |
| 12,20 | 1,20 | 0,28 | -0,01 | 22 | 0,498 | 0,018 | | |
| | | | | | | **Average** | **0,017** | **0,278** |

**Table 9, Calculation of friction constants of drive P3**

Calculation of $c_\varphi$ constant and guess of rotor inertia are shown below. The same advance as for motor P1 and P2 is used. But this time motor resistance $R = 12,2\Omega$. Best inertia guess for P3 is **0.05 kgm$^2$**

| cphi constant calculation | | | | |
|---|---|---|---|---|
| w [rad/s] | I[A] | U[V] | R [ohm] | $c_\Phi$ [V/rad/s] |
| 2,70 | 0,18 | 6 | 12,2 | 1,409 |
| 3,95 | 0,20 | 8 | | 1,408 |
| 5,10 | 0,22 | 10 | | 1,435 |
| 6,30 | 0,23 | 12 | | 1,459 |
| 7,55 | 0,24 | 14 | | 1,466 |
| 8,70 | 0,25 | 16 | | 1,489 |
| 9,90 | 0,26 | 18 | | 1,498 |
| 11,00 | 0,27 | 20 | | 1,519 |
| 12,20 | 0,28 | 22 | | 1,523 |
| 13,30 | 0,31 | 24 | | 1,520 |
| | | | **Average** | **1,473** |

**Table 10, Calculation $C_\Phi$ constant of drive P3**

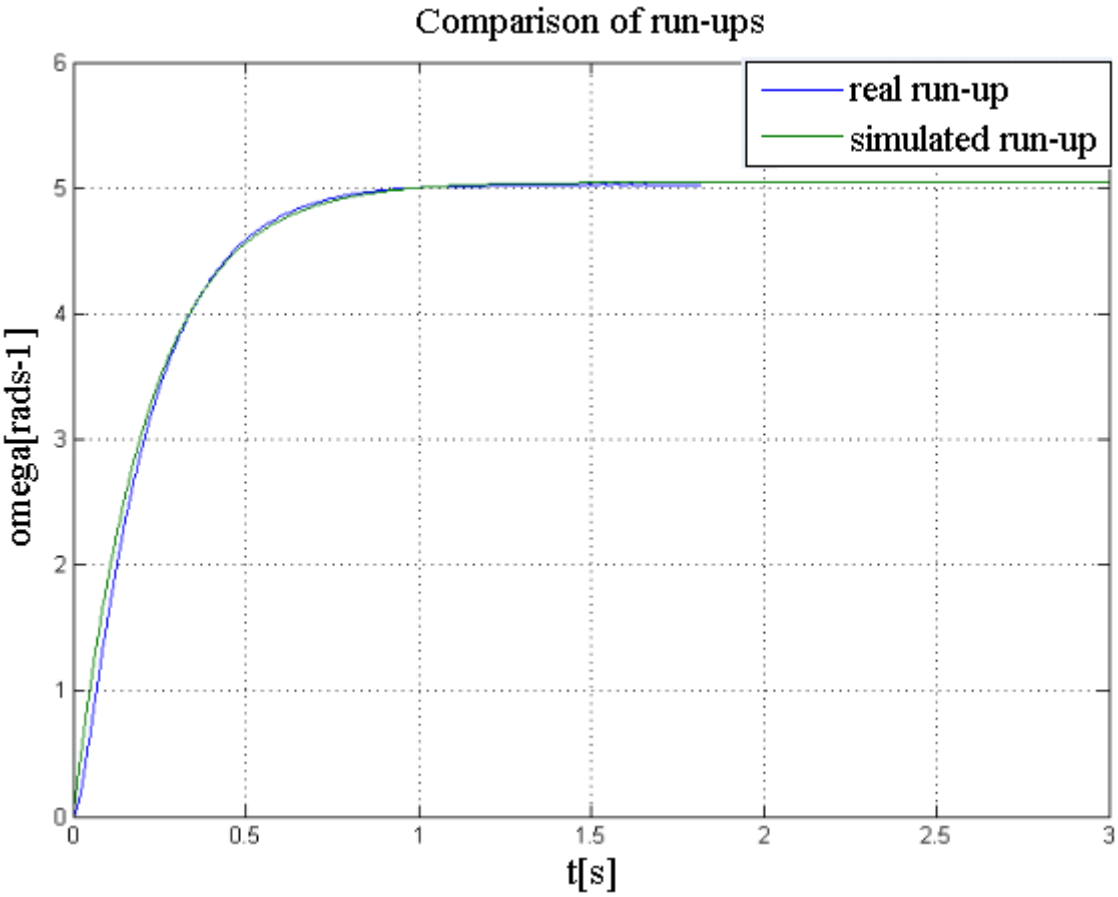



**Figure 20, Comparison, simulated and real drive run-ups of P3**

Parameters of P3 drive and comparison of measured angular velocities and simulated angular velocities can be seen in table 11.

| P3 parameters | |
|---|---|
| kfi [Nm/A] | 1,78 |
| b [Nm/rads-1] | 0,018 |
| T [Nm] | 0,028 |
| cfi [V/rads-1] | 1,473 |
| I [kgm2] | 0,05 |

| | Speed of Rotation [rad/s] | |
|---|---|---|
| Voltage [V] | meassured | simulated |
| 10 | 5,0 | 5,0 |
| 14 | 7,6 | 7,6 |
| 16 | 8,7 | 8,8 |
| 18 | 9,9 | 10,0 |
| 20 | 11,0 | 11,3 |
| 22 | 12,2 | 12,5 |

**Table 11, Parameters of P3 drive on the left and comparison of measured and simulated speed of rotation on the right**

## 5.3. Inverse kinematics model

In equations (3.13) in this case $\vartheta_1 = q_1$, $\vartheta_2 = q_2$ and $\vartheta_3 = q_3$. To be able to solve equations (3.13) it is necessary to define another two angles α and β, meaning of which is clear from Fig. 21. Calculation of $q_2$ and $q_3$ is more complicated. Firstly parameter $c$ needs to be evaluated.

$$\begin{aligned} p &= \sqrt{y^2 + x^2} \\ c &= \sqrt{(L_1 - z)^2 + p^2} \end{aligned} \tag{5.14}$$

In equation (5.14) $p$ stands for the projection of $L_2$ and $L_3$ arms into $xy$ plane. Dimensions for arms were obtained from drawings of manipulator and are shown in table 12. After $p$ is evaluated, it is possible to calculate angles α and β. Calculation of β angle is done by cosine theorem.

$$\begin{aligned} \alpha &= arctg\,\frac{p}{L_1 - z} \\ \beta &= arc\cos\left(\frac{{L_2}^2 + c^2 - {L_3}^2}{2L_2 c}\right) \end{aligned} \tag{5.15}$$

When angles $\alpha$ and $\beta$ are known second and third steering angles can be calculated.

$$\begin{aligned} \vartheta_2 &= \alpha + \beta \\ \vartheta_3 &= -\left(\pi - \arccos\left(\frac{{L_2}^2 + {L_3}^2 - c^2}{2L_2 L_3}\right)\right) \end{aligned} \tag{5.16}$$

Calculation of first steering angle is easier because of known Cartesian coordinates x and y. $\vartheta_1$ can be simply calculated using inverse tangent function.

$$\vartheta_1 = arctg\,\frac{x}{y} \tag{5.17}$$

When using inverse kinematics model some restrictions are important. [1] It must be secured that if condition described in (5.18) is true, then second and third steering angles are assigned values as shown in (5.19).

$$c = L_2 + L_3 \tag{5.18}$$

$$\begin{aligned} \vartheta_3 &= 0 \\ \vartheta_2 &= \alpha \end{aligned} \tag{5.19}$$

Condition $c = L_2 + L_3$ should be satisfied as a range.

$$c \le \left( (L_2 + L_3) + \lim \right) \wedge c \ge \left( (L_2 + L_3) - \lim \right) \tag{5.20}$$

In (5.20) *lim* is a small number. This can help to avoid noise disturbances. Also when robot moves from zero position, some rules should be obeyed in order to avoid rapid changes in required position. From zero position movement should be suggested in the way, that it moves only a little either in $x$ coordinate or in $y$ coordinate. After this, further movement is arbitrary.

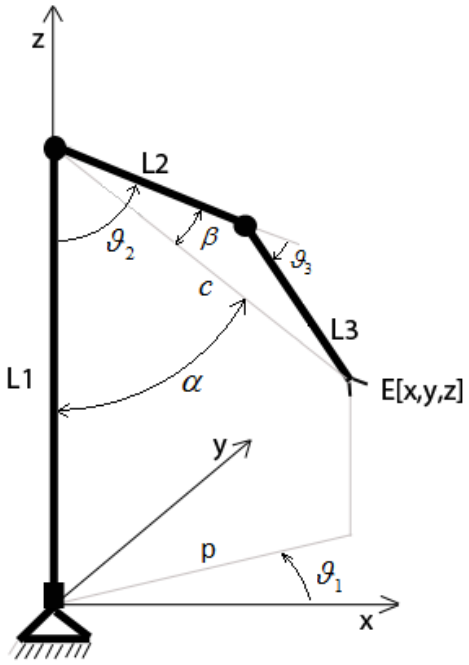



**Figure 21, Inverse kinematics - model for calculations**

| Arm | Length [mm] |
|---|---|
| $L_1$ | 485 |
| $L_2$ | 180 |
| $L_3$ | 180 |

**Table 12, Lengths of arms for inverse kinematics calculation**

## 5.4. Inverse dynamics model

To make an inverse dynamics model of the system certain parameters are important to be known. These parameters are rotational matrix for each arm, position of center of gravity of each arm in arm's CS, position of each arm's CS origin in CS0 and friction coefficients. These parameters can be seen in tables below. In order to be sure, that modeled system is modeled correctly, three different methods are used. All the results are supposed to match each other. Some of the parameters correspond to another axis in another method.

### 5.4.1. Model of 3DOF manipulator using Simmechanics

Links coordinate systems defined for inverse dynamics model calculations for Simmechanics and LR2 method are shown in Fig. 22. $CS0$ is fixed coordinate system. This system does not move at all. $CS1$ is coordinate system attached to $arm1$. $CS2$ is coordinate system attached to $arm2$. And respectively $CS3$ is coordinate system attached to $arm3$.

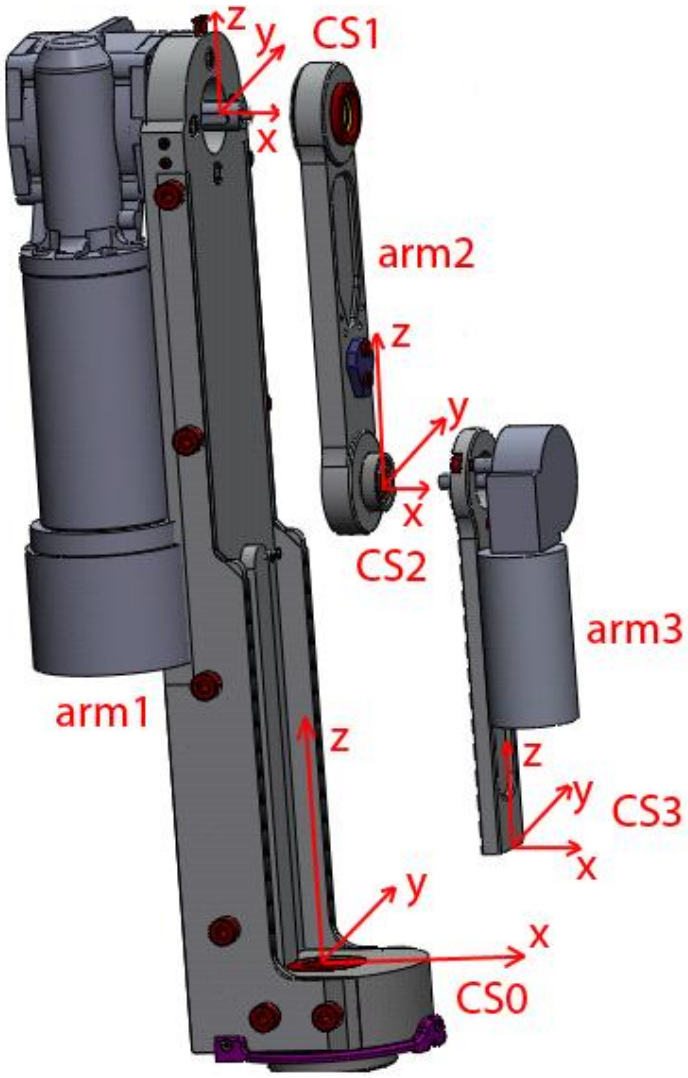



**Figure 22, Coordinate systems for LR2 and Simmechanics methods**

Inverse dynamics model block diagram made in Simmechanics is shown in Fig. 23.

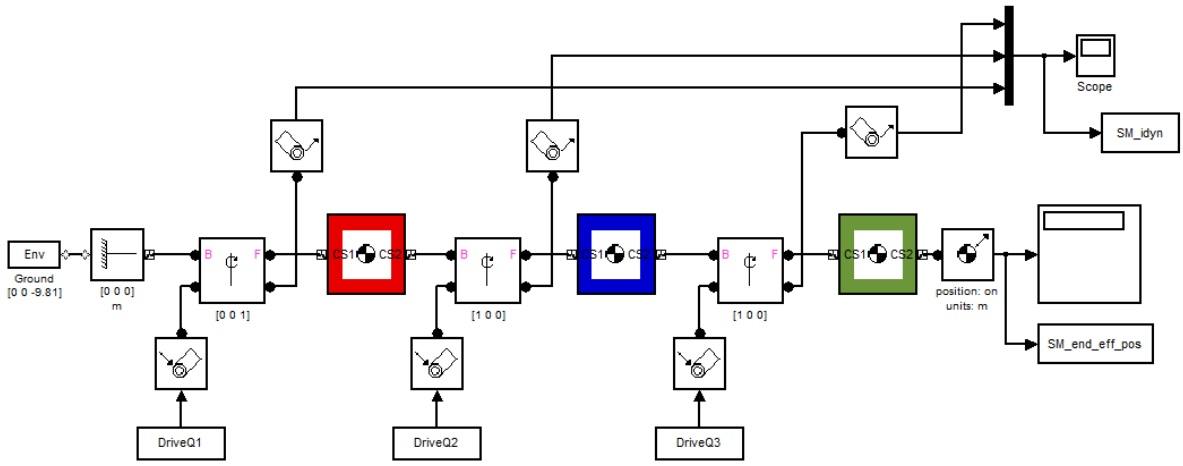



**Figure 23, Model for inverse dynamics calculation in Simmechanics**

In this block diagram DriveQ1, DriveQ2, DriveQ3 are required positions, velocities and accelerations for each joint. Output of this model is SM_idyn where required torques are stored. Color blocks are bodies of the system. These represent arms. In table 13 defined inertia matrices gained from Solidworks model are shown.

| Inertia matrix for Simmechanics and LR2 method | | | | | |
|---|---|---|---|---|---|
| Arm 1 [kgm^2] | | Arm 2 [kgm^2] | | Arm3 [kgm^2] | |
| Ixx | 0,2324 | Ixx | 0,0025 | Ixx | 0,0021 |
| Ixy | 0 | Ixy | 0 | Ixy | 0 |
| Ixz | 0 | Ixz | 0 | Ixz | 0 |
| Ixy | 0 | Ixy | 0 | Ixy | 0 |
| Iyy | 0,236 | Iyy | 0,0024 | Iyy | 0,0021 |
| Iyz | 0 | Iyz | 0 | Iyz | 0 |
| Ixz | 0 | Ixz | 0 | Ixz | 0 |
| Iyz | 0 | Iyz | 0 | Iyz | 0 |
| Izz | 0,0232 | Izz | 0,00001 | Izz | 0,00001 |

**Table 13, Inertia matrices of arms for Simmechanics and LR2 method**

Only moments of inertia are taken into consideration. All products of inertia are ignored, because they are much smaller then moments of inertia. Positions of centers of gravity in coordinate systems of arms are shown in table 14. Positions of origins of coordinate systems in fixed coordinate system $CS0$ are set in Simmechanics body block as described further. Translated from origin tag is set as *adjoining* and the distance between input and output coordinate system of the arm is set. Position with respect to $CS0$ is then calculated automatically. Used setting is show in table 15.

| Position of COG with respect to link coordinate system | | | | | |
|---|---|---|---|---|---|
| Simmech method [m] | | LR2 method [m] | | DH method [m] | |
| X_arm1 | -0,0101 | X_arm1 | -0,0101 | X_arm1 | -0,0125 |
| Y_arm1 | -0,0125 | Y_arm1 | -0,0125 | Y_arm1 | -0,2383 |
| Z_arm1 | -0,2383 | Z_arm1 | -0,2383 | Z_arm1 | -0,0101 |
| X_arm2 | -0,0069 | X_arm2 | -0,0069 | X_arm2 | -0,0987 |
| Y_arm2 | 0 | Y_arm2 | 0 | Y_arm2 | 0 |
| Z_arm2 | 0,0987 | Z_arm2 | 0,0987 | Z_arm2 | 0 |
| X_arm3 | 0,0242 | X_arm3 | 0,0242 | X_arm3 | -0,1367 |
| Y_arm3 | -0,0136 | Y_arm3 | -0,0136 | Y_arm3 | -0,0136 |
| Z_arm3 | 0,1367 | Z_arm3 | 0,1367 | Z_arm3 | 0,0242 |

**Table 14, Positions of arms COGs for all methods**

| | Position of arms CS origin in CS0 for Simmechanics | |
|---|---|---|
| Arm 1 | x position | 0 |
| | y position | 0 |
| | z position | L1 |
| Arm 2 | x position | 0 |
| | y position | 0 |
| | z position | -L2 |
| Arm 3 | x position | 0 |
| | y position | 0 |
| | z position | -L3 |

**Table 15, Position of arms' CS origins with respect to CS0 in Simulink**

### 5.4.2. Model of 3DOF manipulator using DH parameters

To calculate equations (3.14) in Matlab robotic toolbox using DH parameters convention, recursive Newton-Euler algorithm is used. Defined parameters are shown in table 16 and illustrated in Fig. 24.

| | α [rad] | a [m] | θ [rad] | r [m] |
|---|---|---|---|---|
| Arm 1 | $-\pi/2$ | 0 | $\pi/2$ | 0,485 |
| Arm 2 | 0 | 0,18 | $-\pi/2$ | 0 |
| Arm 3 | 0 | 0,185 | 0 | 0 |

**Table 16, DH parameters**

DH parameters are *α, a, θ, r*. In Fig. 24. parameters *a* and *r* are illustrated. CS0, CS1, CS2, CS3 are coordinate systems oriented for DH calculation. Axes of these coordinate systems are turned with respect to each other. Angles of these rotations are parameters $\alpha$ and $\theta$. These are not show in the figure because of lucidity. Parameters $r$ and $\theta$ are also parameters which represent rotation angles or translation movements of joints. For more detail about DH parameters and information how to define DH parameters see [3].

Inertia matrices for all arms and positions of centers of gravity have changed when compared to those ones used in Simmechanics method. Positions of centers of gravity are

shown in table 14. Inertia matrices are shown in table 17. Origins of coordinate systems of arms are derived automatically from DH parameters.

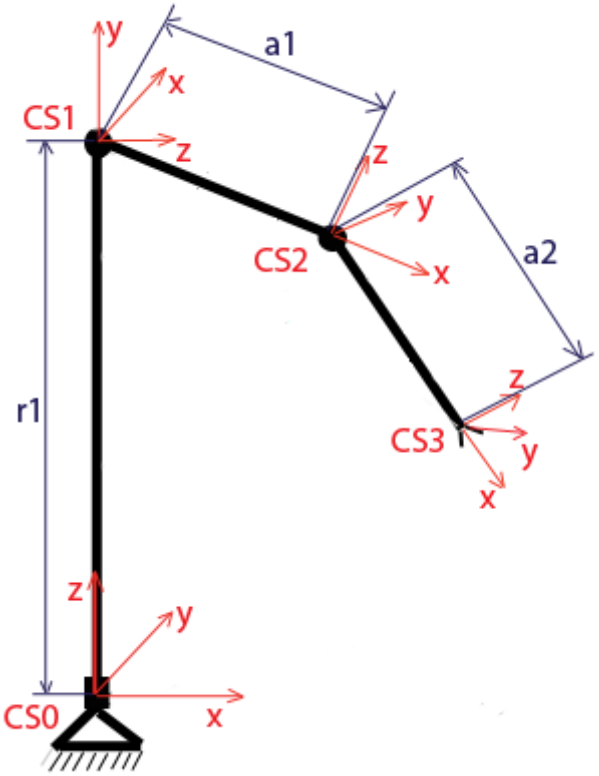



**Figure 24, DH parameters**

| Inertia matrix for DH parameters method | | | | | |
|---|---|---|---|---|---|
| Arm 1 [kgm^2] | | Arm 2 [kgm^2] | | Arm3 [kgm^2] | |
| Ixx | 0,236 | Ixx | -0,00001 | Ixx | -0,00001 |
| Ixy | 0 | Ixy | 0 | Ixy | 0 |
| Ixz | 0 | Ixz | 0 | Ixz | 0 |
| Ixy | 0 | Ixy | 0 | Ixy | 0 |
| Iyy | 0,0232 | Iyy | 0,0024 | Iyy | 0,0021 |
| Iyz | 0 | Iyz | 0 | Iyz | 0 |
| Ixz | 0 | Ixz | 0 | Ixz | 0 |
| Iyz | 0 | Iyz | 0 | Iyz | 0 |
| Izz | 0,2324 | Izz | 0,0025 | Izz | 0,0021 |

**Table 17, Inertia matrices of arms for DH parameters method**

### 5.4.3. Model 3DOF manipulator using LR2 – automated calculation method

Calculation of motion equations is made according to equations in *calculation of matrices* section and is made in Maple and Matlab – Symbolic Toolbox as well. In Maple all results are much more complicated than those ones calculated in Matlab – Symbolic Toolbox. Calculated results are too long to be shown but can be found in Appendixes. Only equation that describes movement of P3 drive calculated in Matlab – Symbolic Toolbox takes approximately half of a page.

### 5.4.4. Comparison of calculated inverse dynamics models

To compare quality of models torques calculated with all three methods are compared. Movement for the comparison is planned in joint coordinates and starts from zero position. This position is labeled $q_0$. Then it continues to positions $q_1$, $q_2$ and $q_3$. Values of positions are shown below. Note that these models do not take friction torques into consideration. Simulated torques, which are needed to achieve required positions are composed only from centripetal torque, coriolis torque and gravity load.

$$q_0 = [0, 0, 0] \quad \rightarrow \quad q_1 = [-1.9, 0.6, -0.43]$$
$$q_1 = [-1.9, 0.6, -0.43] \quad \rightarrow \quad q_2 = [-1.9, 0.4, 0.2]$$
$$q_2 = [-1.9, 0.4, 0.2] \quad \rightarrow \quad q_3 = [1, -0.1, -0.3]$$

In Fig. 25 simulated torques are shown. Graph labeled Torque 1 corresponds to torque on drive P1 and respectively Torque 2 and Torque 3 corresponds to torques on drives P2 and P3.

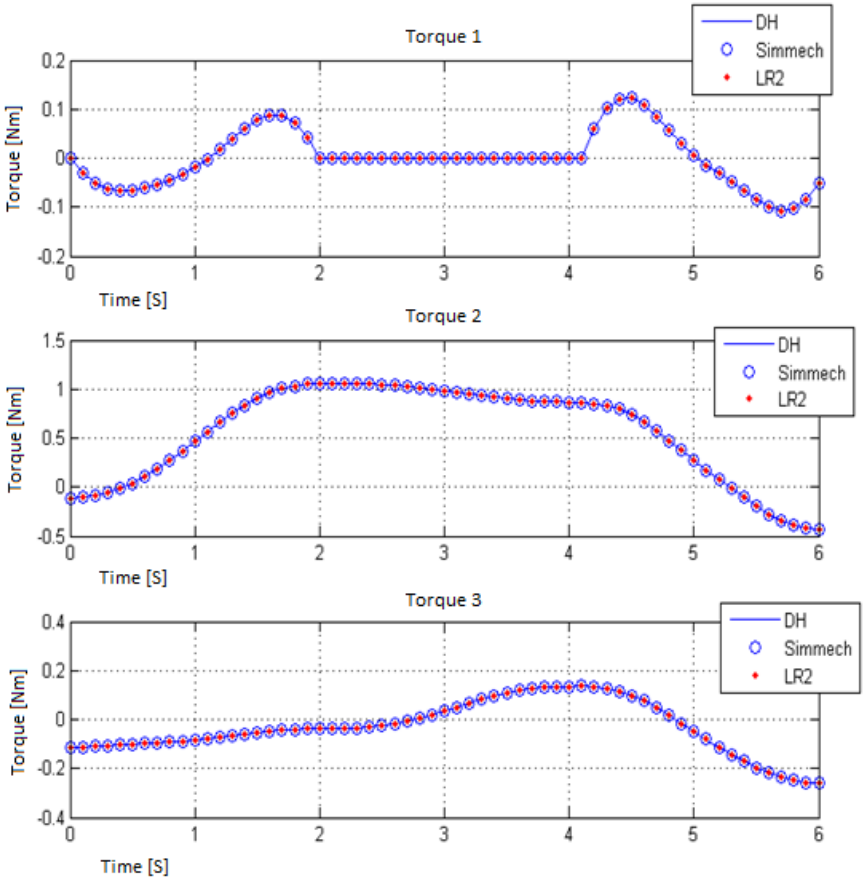



**Figure 25, Comparison of torques calculated by different methods**

### 5.4.5. Adding friction

Firstly only viscous friction is taken into consideration. A real manipulator is supposed to track some planned movement driven only by torques calculated in model. This method cannot harm the manipulator, because, action torque with no friction involved is smaller than with friction.

Viscous friction coefficients are tuned experimentally. Firstly the robot is driven by computed action torque, which ignores friction load. Afterwards the $B_i$ coefficients are added one by one to the model according to formula (3.17) up to the real movement had satisfactory correspondence with required movement. Tuned $B_i$ coefficients are shown in table 18. on the left and movement based on these coefficients shown in Fig. 26.

After only viscous friction coefficients are found, than Coulomb friction coefficients can be found. Firstly viscous coefficients are divided by 2 and Coulomb friction coefficients are progressively increased until movement in the slow region is fast enough. Then $B_i$ coefficients can be retuned a little. To be able to fit the required movement more precisely a simpler movement is chosen in comparison with movement used, when only viscous friction coefficients are guessed.

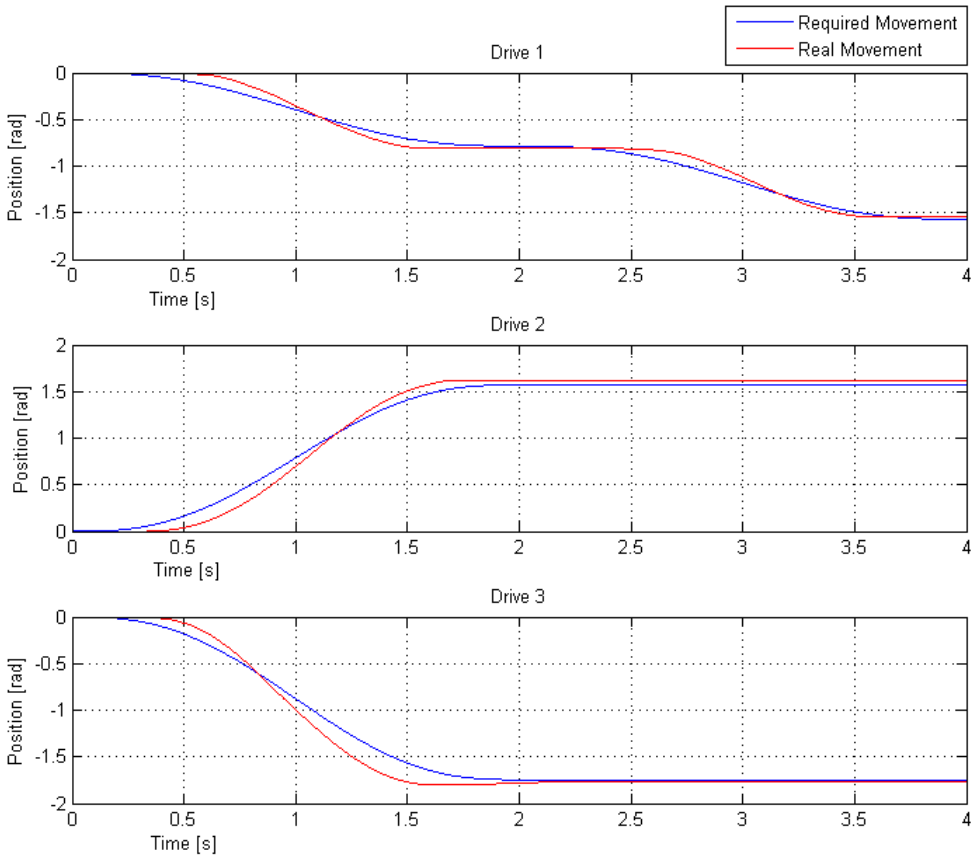



**Figure 26, Movement modeled with viscous friction**

As can be seen in Fig. 26 drive 3 correspond worse to required movement, than other two motors. This is because the structure of motors is not known exactly. Structure of motors P1 and P2 does not affect the overall movement as much as structure of motor P3.

| Arm | $B_i\,[Ns]$ |
|---|---|
| Arm 1 | 5,1 |
| Arm 2 | 1,45 |
| Arm 3 | 6,4 |

| Arm | $B_i\,[Ns]$ | $Q_c\,[Nm]$ |
|---|---|---|
| Arm 1 | 2,6 | 1,48 |
| Arm 2 | 0,77 | 0,86 |
| Arm 3 | 4,16 | 2,3 |

**Table 18, Only viscous friction coefficients on the left and viscous and coulomb friction coefficients on the right**

Movement modeled only with viscous friction compensation is easier to model and precise enough to perform feedforward control strategy. On the other hand action torque, which is modeled with compensation of both frictions, follows the movement of the real system better.

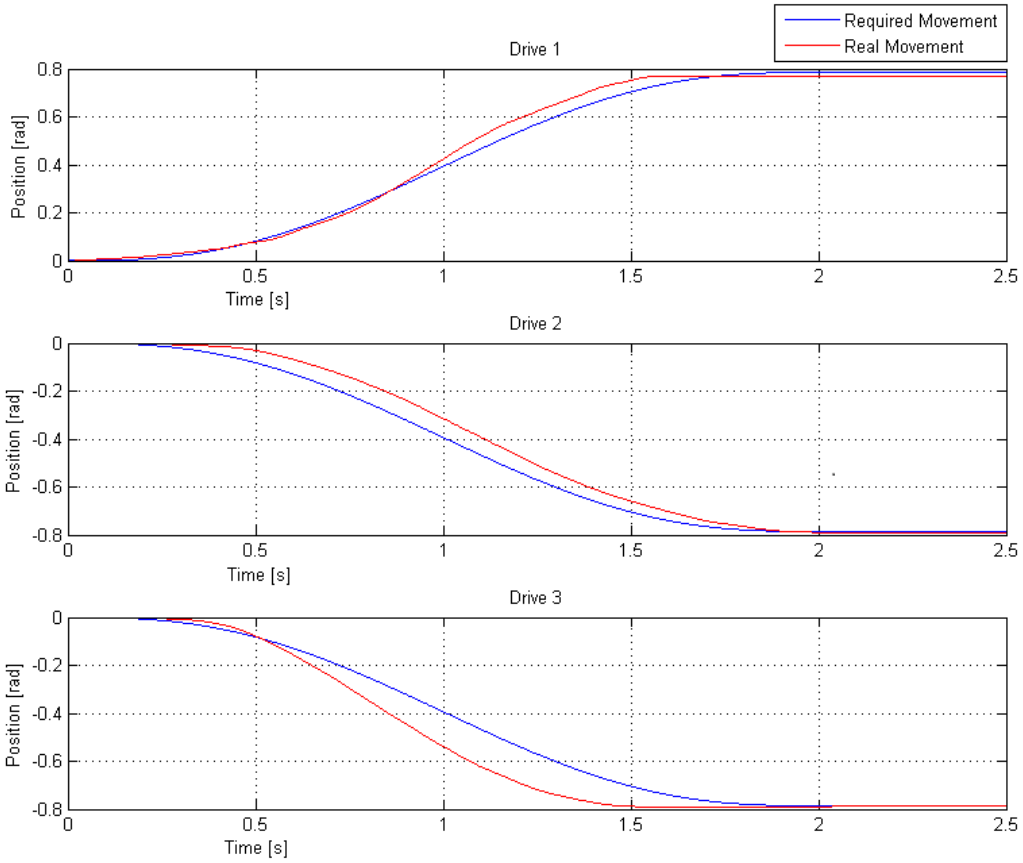



**Figure 27, Movement modeled with viscous and Coulomb friction**

# 6. Control of manipulator

## 6.1. Control with PID controllers

### 6.1.1. Trajectory planning

In order to the planned movement in Cartesian coordinates was smooth, it was planned offline with fifth order polynomial using jtraj function from Robotic Toolbox. Then it was online transformed into joint coordinates using inverse kinematic model. Trajectory planning data flow and Simulink model for control can be seen in Figures 28 and 29.

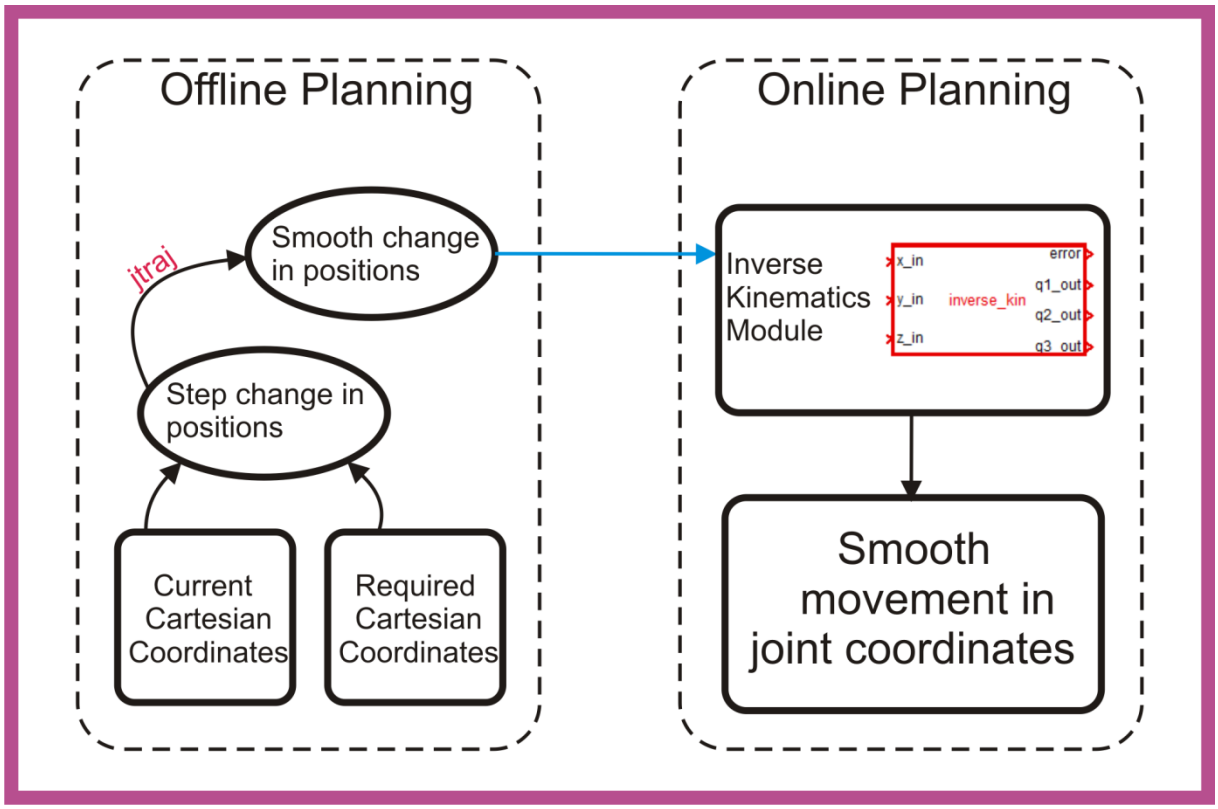



**Figure 28, Trajectory planning data flow for 3xPID control**

### 6.1.2. Tuning PID regulators

Three different PID regulators, each for one of three motors are separately tuned. Disturbance caused by movement of other arms is not taken into consideration. This means, that for example if motor P1 movement is disturbed by movements of other motors, this disturbance is considered as external error and must be handled by the regulator action. Each regulator has to be tuned properly to be able to handle this error.

P I D parameters for each regulator are tuned experimentally and final gains are shown in table 19. When tuning regulators, first step is to find P gain which can regulate without overshoot and with the smallest steady state error possible. Then approximately 10 times smaller I gain and 100 times smaller D gain are set. Regulator for each motor is tuned with

other two motors blocked. When all regulators are tuned, mutual movement can be tested. Results from mutual movement can be seen in figures 30, 31 and 32.

| | Motor 1 | Motor 2 | Motor 3 |
|---|---|---|---|
| P | 25 | 10,5 | 18,3 |
| I | 0,9 | 0,6 | 0,11 |
| D | 0,03 | 0,03 | 0,01 |

**Table 19, PID constants for control**

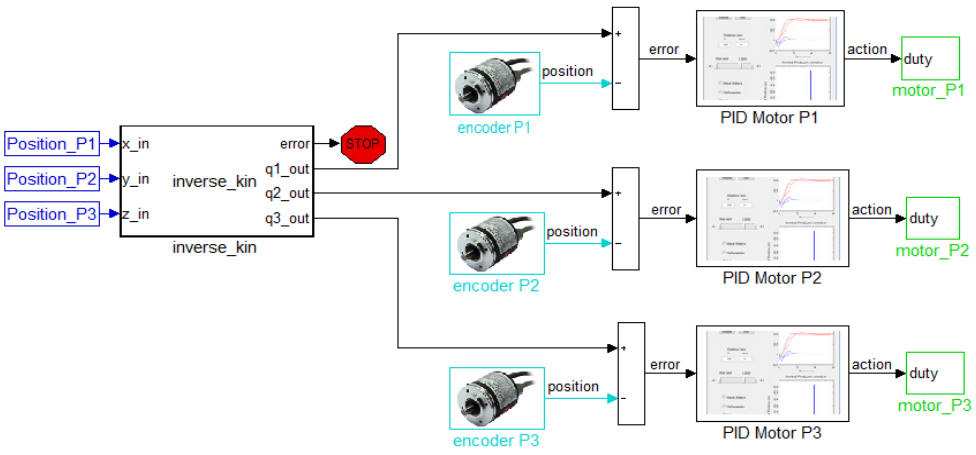



**Figure 29, Simulink model for 3xPID control**

### 6.1.3. Results of 3xPID control

All three motors tracking required trajectory are shown in this section. Results show joint coordinates of each motor. As can be seen in Fig. 31, the second drive requires quite high actions in comparison with action required by feedforward control.

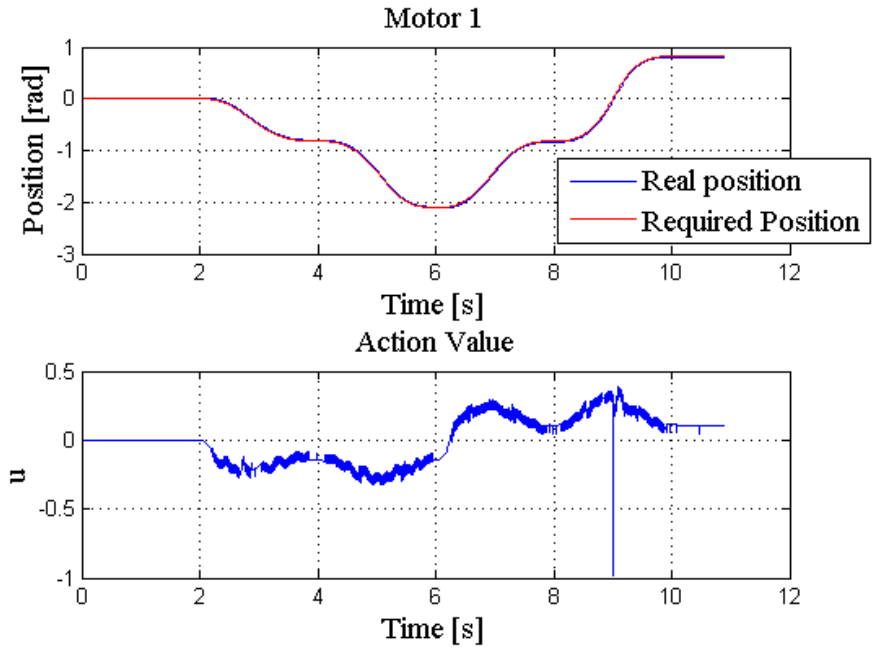



**Figure 30, Motor 1 position tracking and action value, 3xPID control**

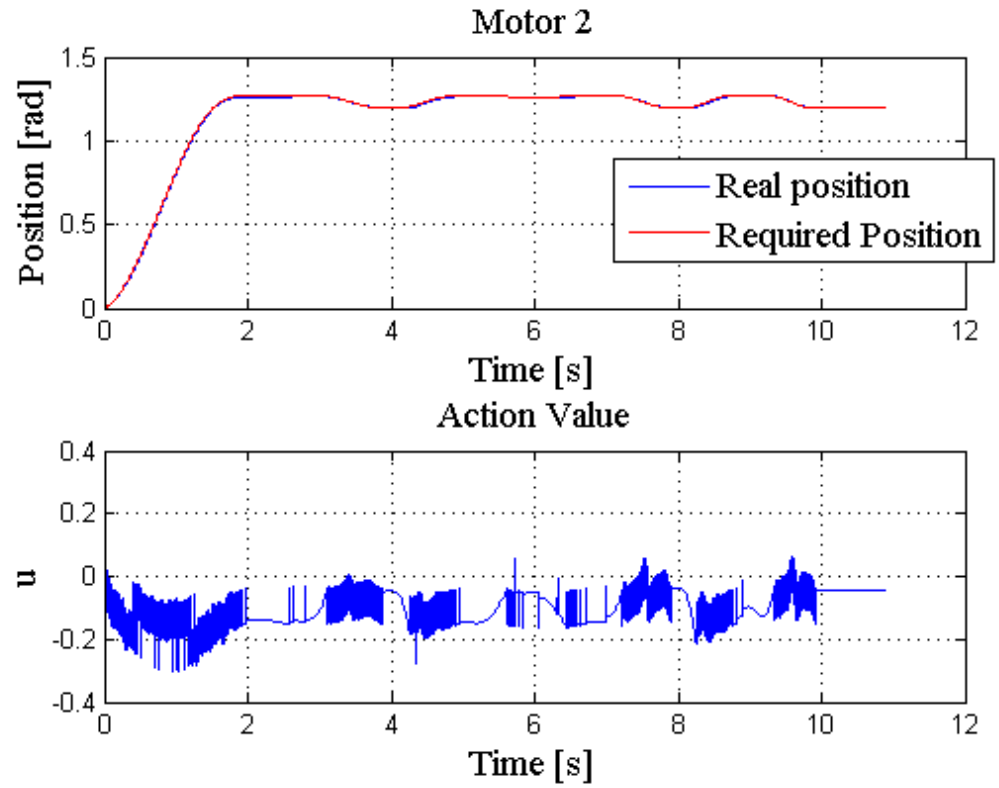



**Figure 31, Motor 2 position tracking and action value, 3xPID control**

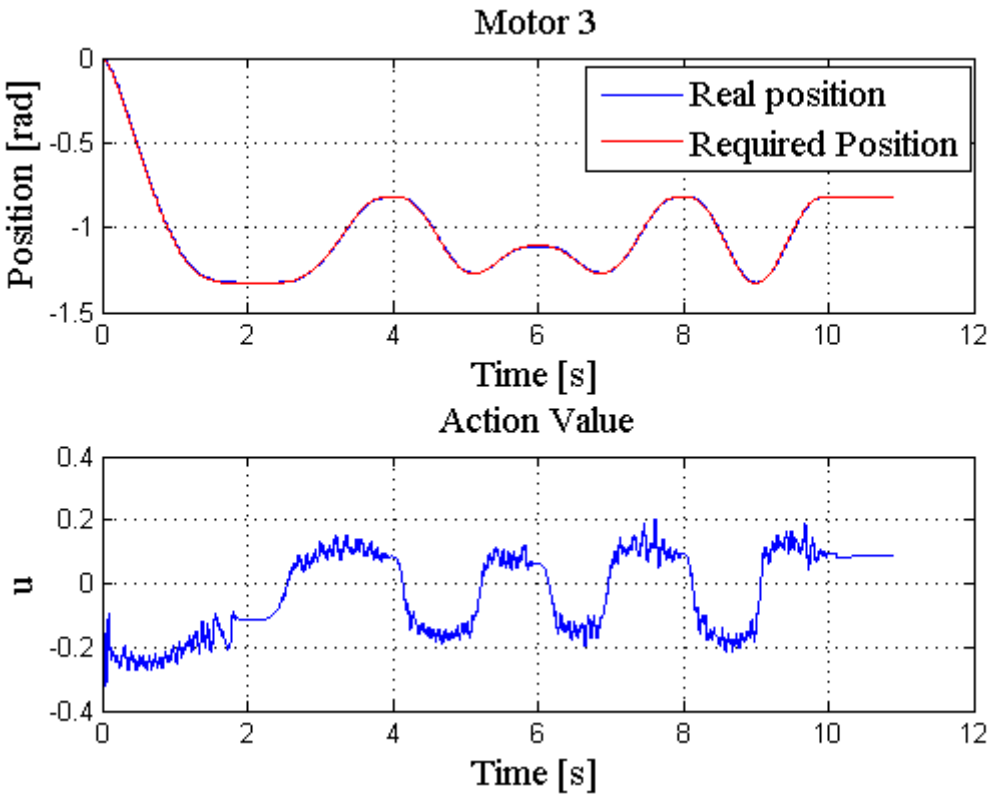



**Figure 32, Motor 3 position tracking and action value, 3xPID control**

## 6.2. Control with PID regulators with feedforward compensation

### 6.2.1. Trajectory planning

Trajectories, velocities and accelerations have to be planned between each two desired points of movement. After these kinematic properties of the movement are planned compensation torques can be calculated. This is done by using offline strategy to calculate trajectories, velocities and accelerations between selected points and online strategy to calculate compensation torques from these trajectories, velocities and accelerations. Trajectory planning data flow and Simulink model for control can be seen in Figures 33 and 34. Compensation torque needs to be divided by gear ratio of corresponding drive.

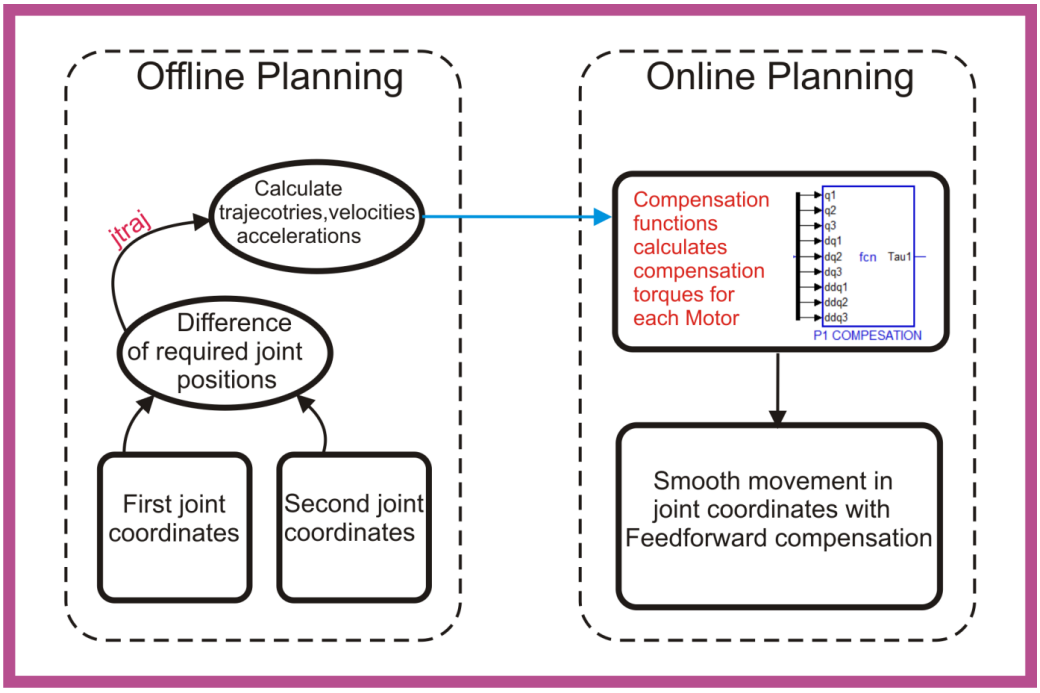



**Figure 33, Trajectory planning data flow for regulator with feedforward compensation**

### 6.2.2. PID regulators tuning

Same gains for PID constants are used as for the simple PID control.

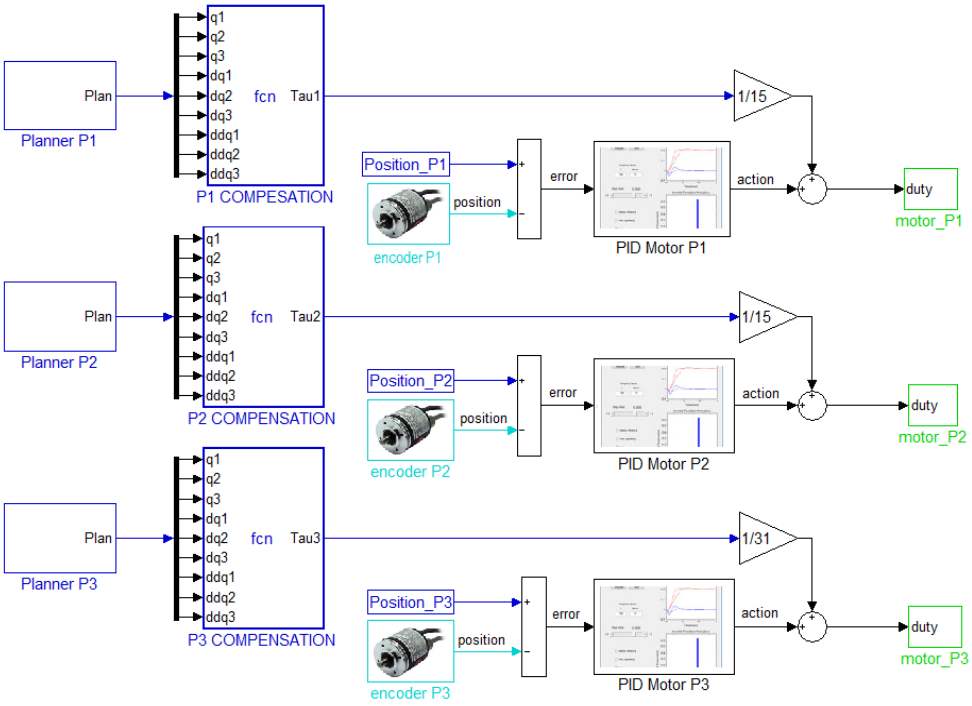



**Figure 34, Simulink model for regulator with feedforward compensation**

### 6.2.3. Results of Feedforward control

Required and real positions and action values for drives P1, P2 and P3 can be seen respectively in Fig. 35. , Fig. 36. and Fig. 37.

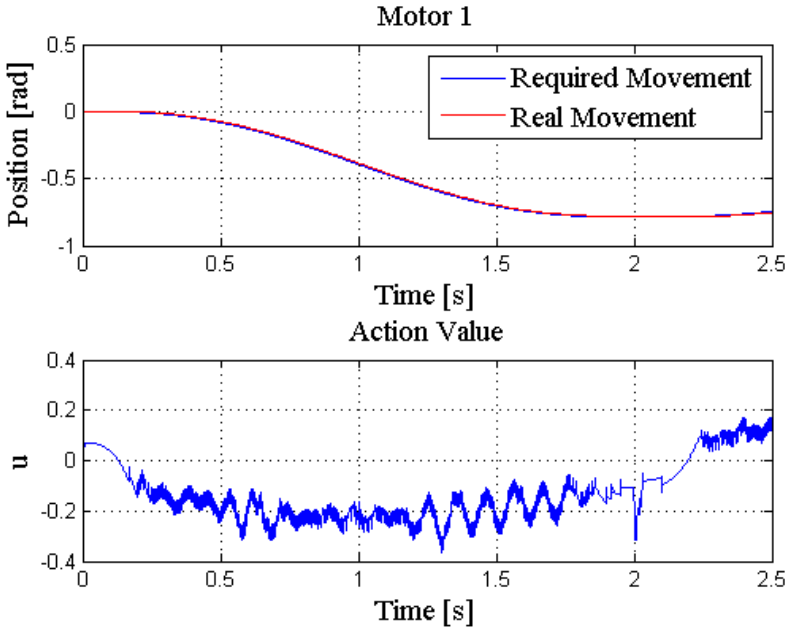



**Figure 35, Motor 1 position and action value, regulator with feedforward compensation**

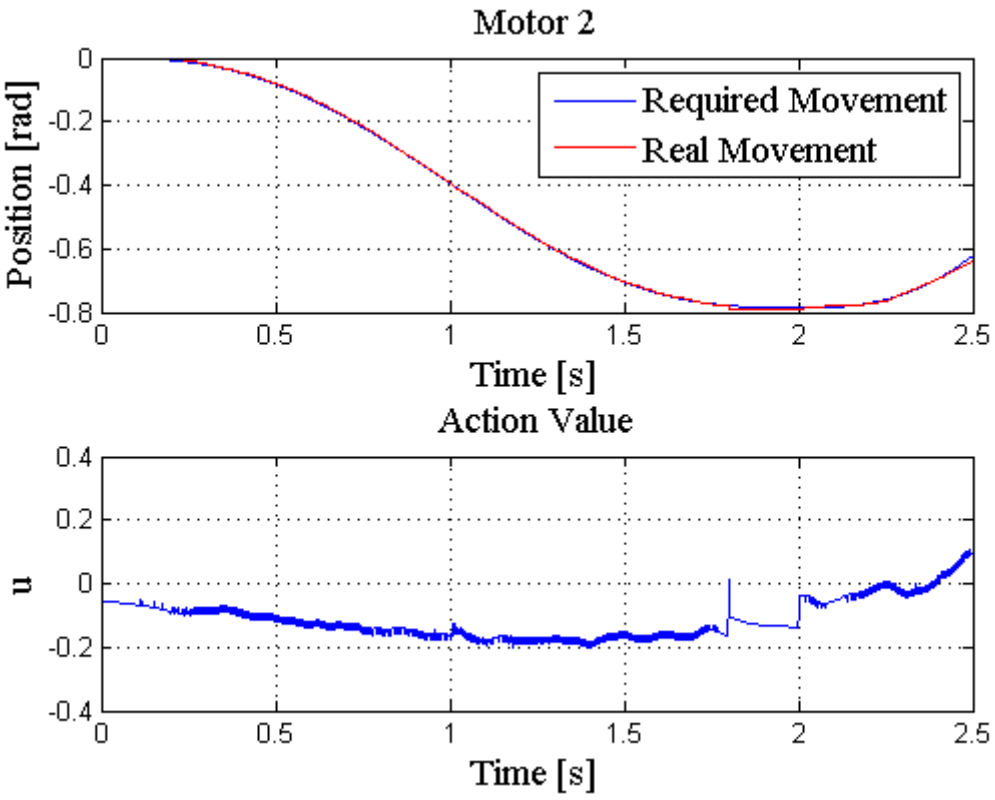



**Figure 36 , Motor 2 position and action value, regulator with feedforward compensation**

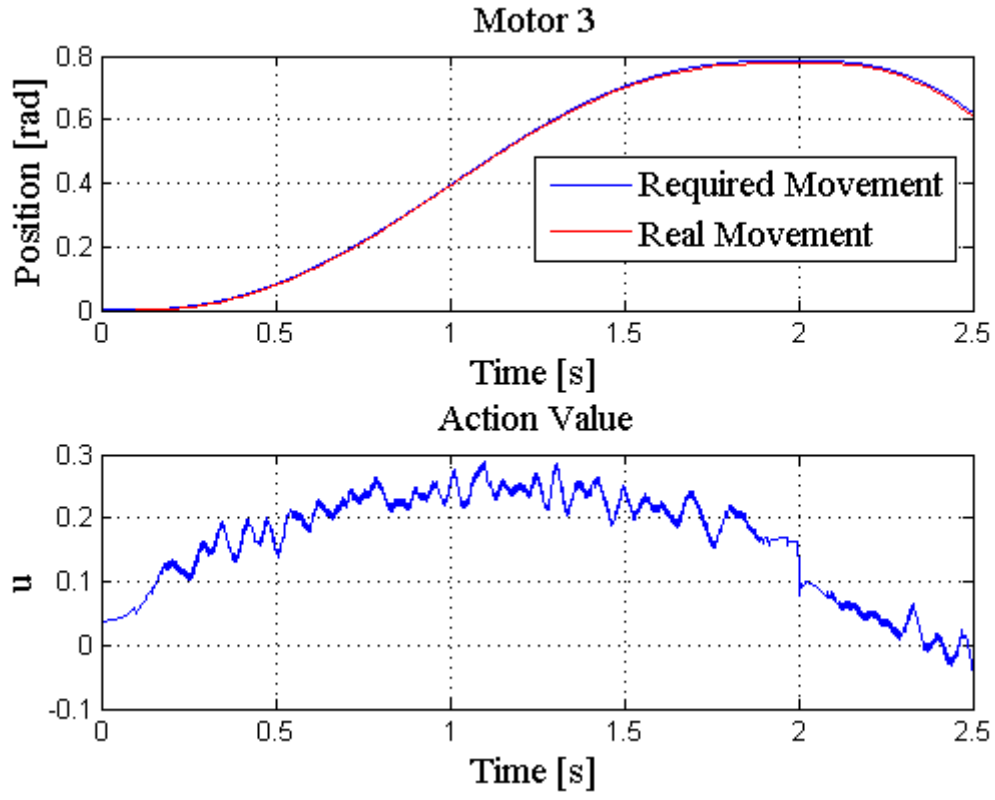



**Figure 37, Motor 3 position and action value, regulator with feedforward compensation**

## 6.3. Comparison of PID and FeedForward Control

### 6.3.1. Comparison Movement

To compare quality of control methods the movement shown in Fig. 38 is chosen. In Fig. 38 required positions as functions of time for each drive are shown.

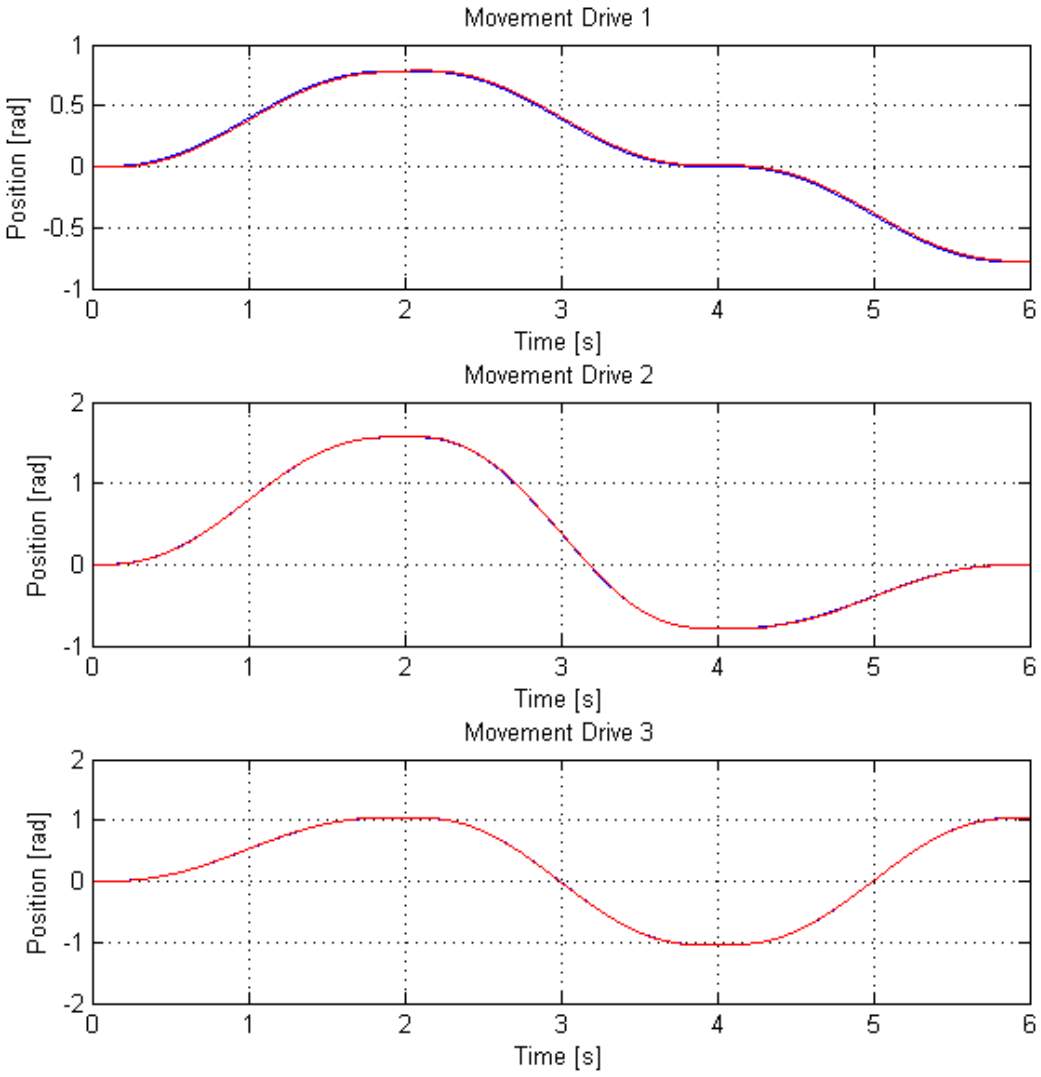



**Figure 38, Movement for comparison of quality of control**

Movement is repeated ten times with PID control and ten times with PID plus feedforward compensation control. Using Matlab *mse* function a mean square error is calculated for each movement and then, the same technique as for model in chapter four is used and the mean of mean square errors is computed. Results can be seen in table 20.

| | Motor 1 | | Motor 2 | | Motor 3 | |
|---|---|---|---|---|---|---|
| | PID | FeedForward | PID | FeedForward | PID | FeedForward |
| mean of MSE | 4,60E-05 | 2,00E-04 | 1,58E-04 | 2,38E-05 | 1,05E-04 | 8,42E-06 |

**Table 20, Comparison of PID and FeedForward control using MSE, MSE is evaluated in [rad$^2$]**

As can be seen from table 20, motor 1 is regulated more precise with only PID control. This can be caused by high toughness of the cable connected to Motor 1. As can be seen from comparison of Fig. 31 and Fig. 36, FFC control decreased action value requirements. However FFC control is strictly dependent on quality of the model, which is not the same in all conditions. PID regulator provides extremely good accuracy as well. Considering the result from the test from chapter four and results from the chapter six it can be seen that for manipulator **PID control is sufficient**. All three motors are powerful, has big gear ratios and are overequipped for the system. Therefore in further development only PID control is going to be used.

# 7. Application

## 7.1. Application description

Because the control with only PID regulators shows very precise results, it was decided, that application will be developed only for PID control strategy. Feedforward controller can be implemented in GUI handler using derivative calculation of desired positions. Robot application is teach and execute application. Using the application the robot can be taught different movements and these movements can be later executed. For better controllability GUI for this application is created. Frame of this GUI can be seen in Fig. 39. As can be seen there are two main control areas. Control area for teaching which is set active when teaching process is supposed to happen and Executing control area. Erasing control area restricts any other action when some movement is about be erased.

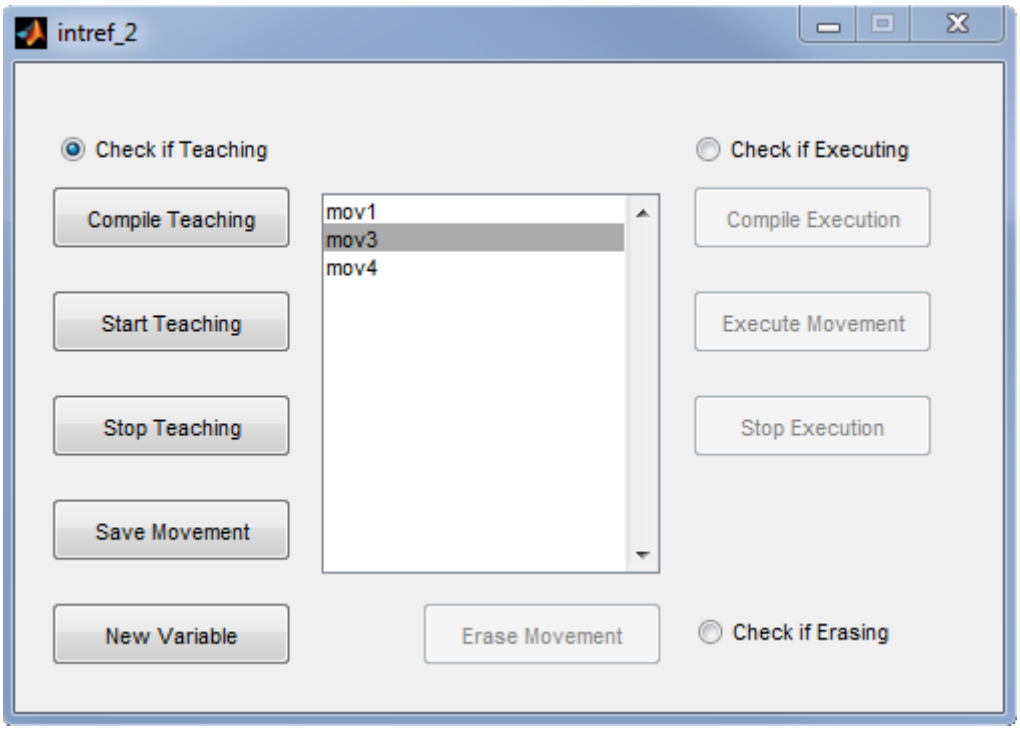



**Figure 39, GUI frame for application**

Each taught movement is stored form Simulink model in variable called Movement in Matlab workspace. Afterwards this variable can be assigned to another name from Data Store Handle of GUI. Variables which are currently active in Data Store Handle can be seen in listbox, shown in Fig. 44.

DS Handle works as variable name Exchange. Other names for variables are created only in DS handle. Every time a new variable is created a window pops up. Pop up window and detailed data flow and operation explanation can be seen in Figures 40 and 41.

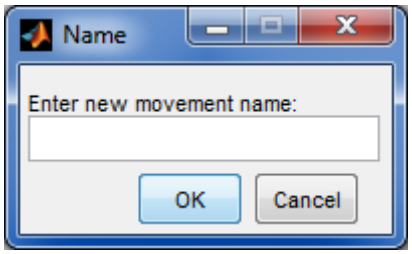



**Figure 40, Popup window for new variable**

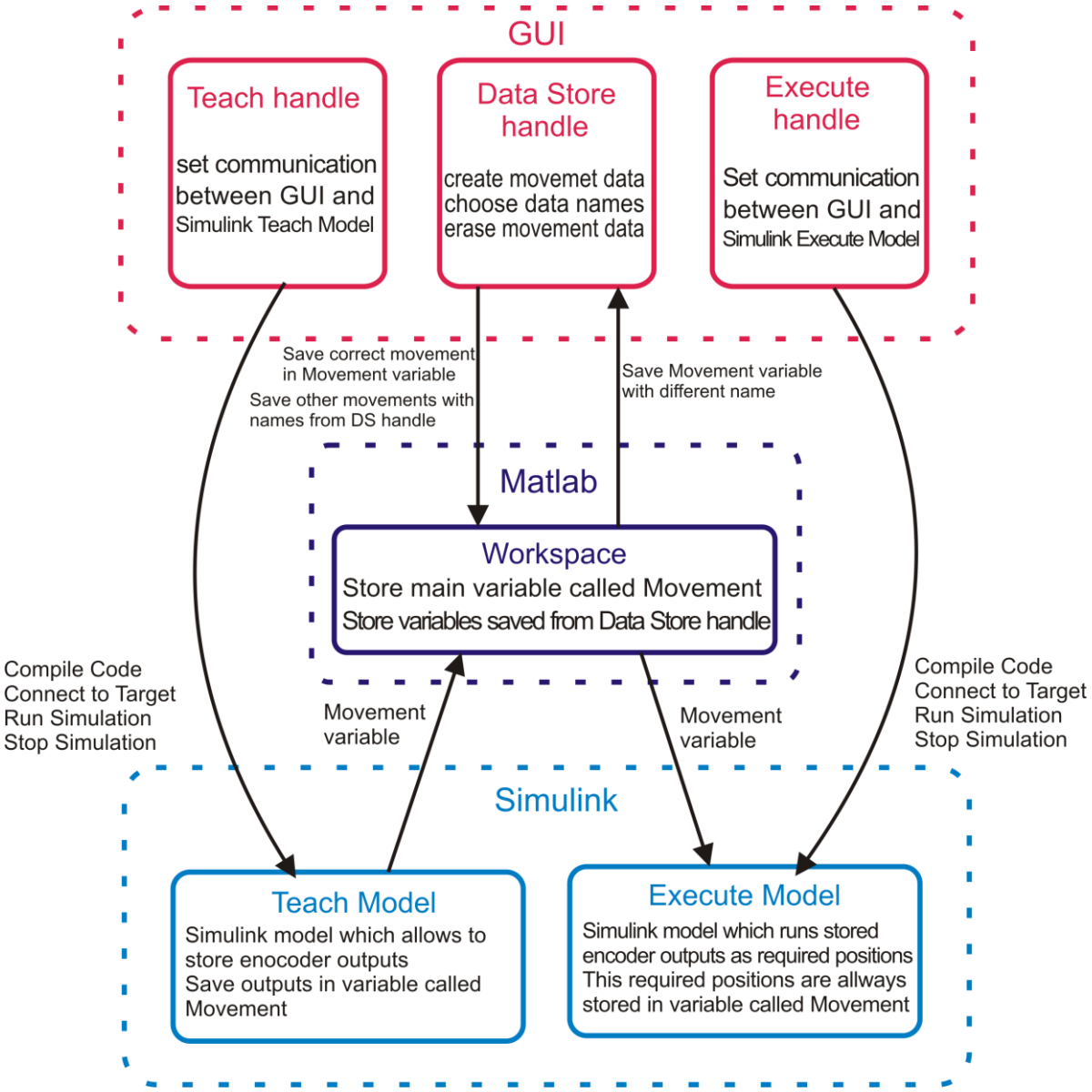



**Figure 41, Description of GUI operation and Data Exchange**

## 7.2. Application programming

The two Simulink models, one for teaching and another one for executing robot movements can be seen in Fig. 42 and Fig. 43. Encoder signals in Teach model need to be filtered. Appropriate filter is found experimentally. This system does not have big demands on signal filtering.

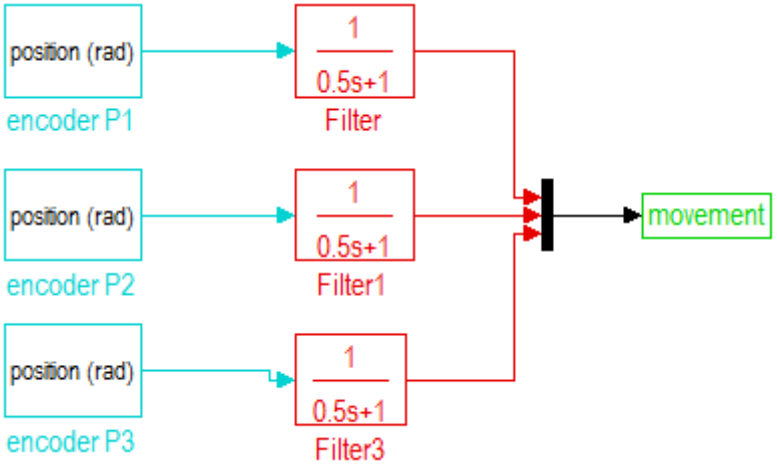



**Figure 42, Simulink model Teach**

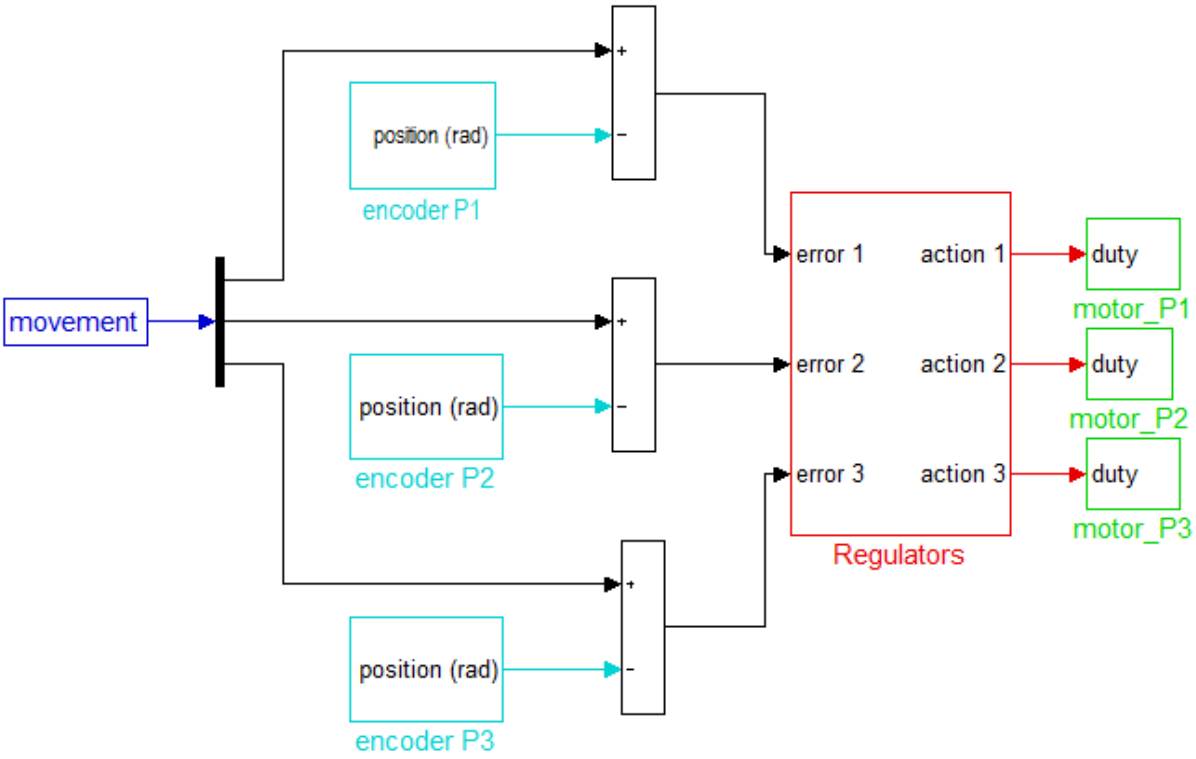
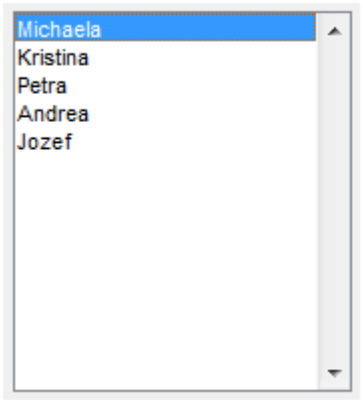



**Figure 43, Simulink model Execute**

Model inputs from Simulink have always blue foreground color. Inputs from physical system have cyan color. Outputs either to workspace or to physical system have green color. Programming GUI required usage of more GUI components and Matlab functions. The most complicated component was Listbox. Listbox is component, which stores string values in rows. Indexes of rows start with one. Each string is aligned a number respectively to row index number. Output of listbox consists of both strings and value. String output contains all strings and value of row index where pointed string is located. Most important used Matlab functions are shown and described in table (21).




```
str =

    'Michaela'
    'Kristina'
    'Petra'
    'Andrea'
    'Jozef'


str_val =

     1
```

**Figure 44, Listbox output.**

**On the left list box with strings and on the right its output when first string is selected**

| | |
|---|---|
| slbuild 'model_name.mdl' | Compile a simulink model run in RTWT with name 'model_name.mdl' |
| get(handles.listbox1,'value') | Return a value of pointed string in Listbox |
| get(handles.listbox1,'string'); | Return all strings in rows from Listbox |
| set_param('model_name.mdl','SimulationCommand','connect') | Connects a simulink model run in RTWT with name 'model_name.mdl' to target |
| set_param('model_name.mdl','SimulationCommand','start') | Starts a simulink model run in RTWT with name 'model_name.mdl' |
| set_param('model_name.mdl','SimulationCommand','stop') | Stops a simulink model run in RTWT with name 'model_name.mdl' |
| evalin('base',new_var_name) | Reads data from WorkSpace and stores them in GUI in *new_var_name* variable |
| assignin('base','movement',data_from_ws); | Assign data stored in GUI variable *data_from_ws* to WorkSpace variable 'movement' |

**Table 21, Main GUI functions used in application**

## 7.3. Application results

Application was tested with several movements and worked well. One of the movements can be seen in figures 45, 46 and 47.

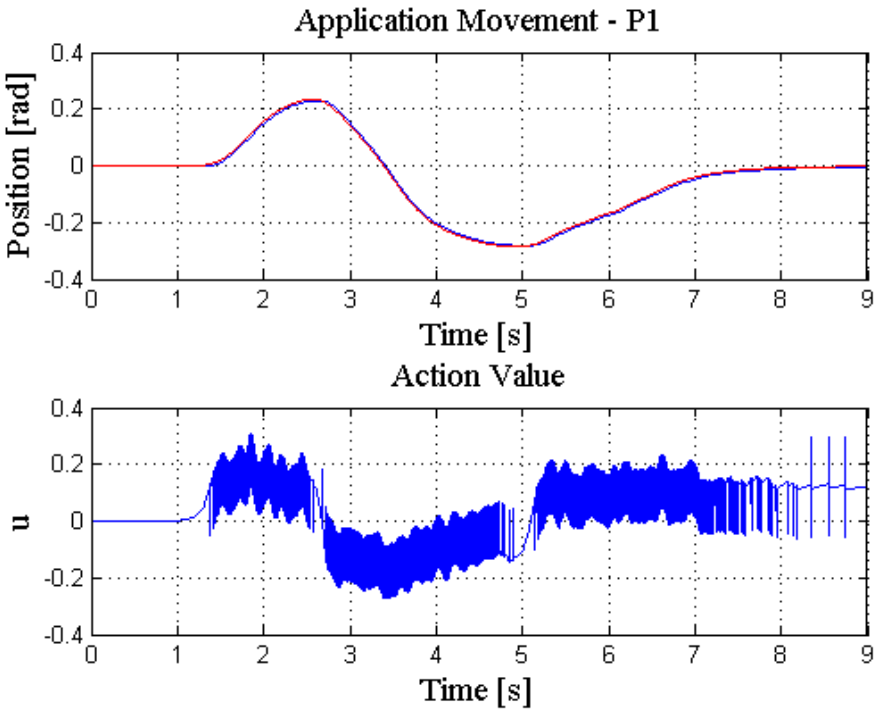



**Figure 45, Application Movement of Motor 1**

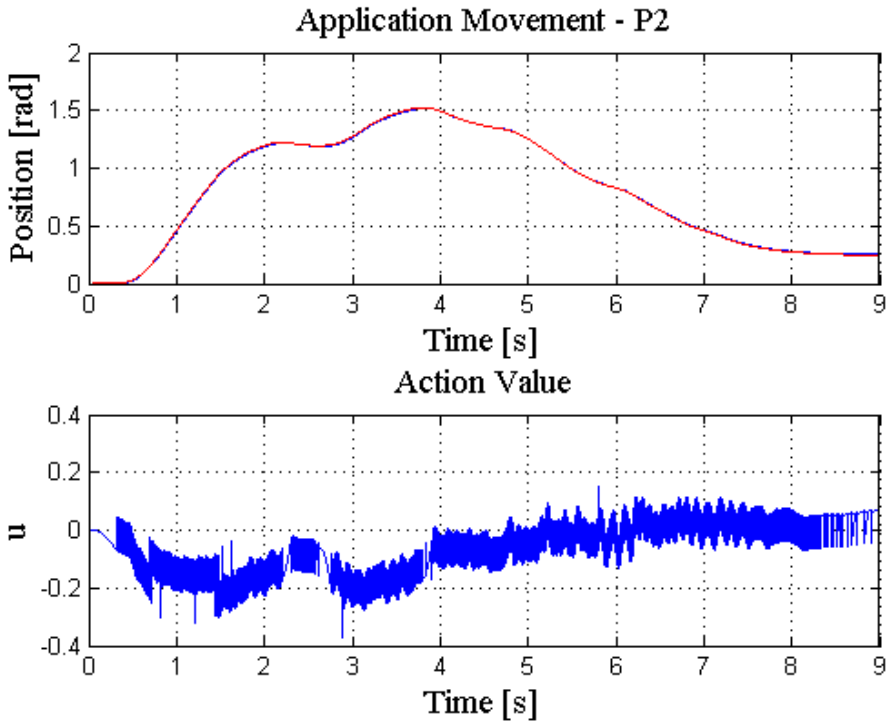



**Figure 46, Application Movement of Motor 2**

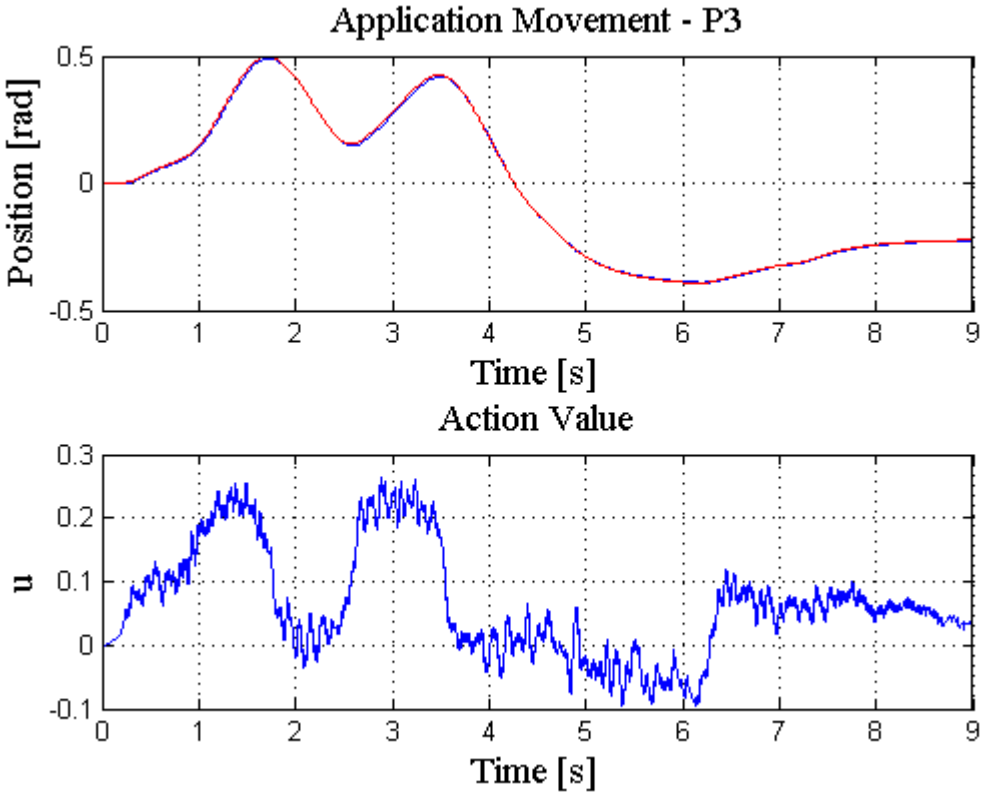



**Figure 47, Application Movement of Motor 3**

# 8. Conclusion

This project was a team project and therefore my thesis processes only a part of the work which is necessary to create a functional 3DoF manipulator.

Main aim of this thesis was develop the model of 3xDoF manipulator and test PID regulator and PID regulation with feedforward compensation. After this is acquired an application was supposed to be developed in order to improve the manipulator.

In the first part of the thesis, the quality of control with PID regulator was compared with quality of control with feedforward compensation on a simple pendulum model. Static and dynamic feedforward compensation was tested. As results of mean square error show, a plain PID regulator had largest value of mse. A slightly better result had PID regulator with static compensation. The best result was achieved with dynamic compensation. MSE values show that feedforward compensation can be used to increase accuracy of the control for systems with small time constants, but quality of the control depends on the quality of the model. However the difference in accuracy is very small and sadly the biggest advantage of FFC could not be tested on the system because it is constructed in such a way, that it is not possible to make it oscillate. In other words, for the system is control with simple PID regulator sufficient.

In the next part models of motors, which were used for manipulator construction were identified. Motor 1 and motor 2 were the same type and motor 3 was different. Therefore two models were developed. Velocities of motor models in steady state were compared with velocities of real motors supplied with certain voltage level. Models of motor reached very accurate results.

Afterwards more complicated system, 3xDoF manipulator was regulated with three independent PID regulators and with dynamic feedforward compensation. It was more complicated to create inverse dynamics model, therefore it was modeled with three different strategies and results from all strategies were compared. Equations of inverse dynamics model were found out with automated calculation method based on Lagrange equations of second kind. Next step was to estimate friction coefficients for this system. It was made experimentally by fitting the graph method. Comparison of regulators shows that for motor 1 there was a more accurate result, when it was regulated only with PID regulator however, for other two motors this was not the case. This was caused by inaccuracy in the inverse dynamics model and by the fact, that the inner structure of motors was unknown and that cables from motor had quite high strength. However the biggest advantage of FFC is that it can increase the stability of a system. But in the end it turned out, that because motors of the system are overequipped and have big gear ratios, it is sufficient to control the manipulator with only 3xPID regulation.

In the last part an application for manipulator was created. Application was developed only for 3xPID control, because it was sufficient control strategy. It was a TEACH-EXECUTE application in which manipulator was able to repeat any taught movement. A graphical user interface was created for this application. Due to the fact that only 3xPID control strategy was taken into consideration, the GUI could be worked out more precisely. The communication between GUI and Simulink was set and Matlab Workspace was used as common memory, where all currently available movements were stored. More possibilities for setting up communication were considered and the simplest and the most suitable was chosen. Two basic functions EVALIN and ASSIGNIN which can pass the data as parameters were used for the communication. Because a Simulink model was programmed in RTWT mode, also *compile* and *connect to target* options were implemented in GUI to achieve better control comfort. In the end manipulator was able to repeat any memorized movement with excellent accuracy and memorize as many movements as needed.

Almost all tasks from assignment were fulfilled. One was approved by thesis supervisor to be omitted and other could not be fulfilled, by reason of the PID control strategy was sufficient. But in the end the 3xDoF manipulator was built, which was able to perform different tasks as for example to repeat taught movements and draw different geometrical pictures. A user friendly human machine interface was created, which allows easy control of manipulator for not trained personnel.

# 9. References


[1] Grepl, R.: Kinematika a dynamika mechatronickým systému. Brno : CERM, 2007, 156 pages. ISBN 978-80-214-3530

[2] Grepl, R.: Modelování mechatronických systému v Matlab/SimMechanics, BEN, 2007

[3] Corke, P.: Robotics, Vision and Control Fundamental Algorithms in Matlab, Springer, 2011, 570 pages, ISBN 978-3-642-20143-1

[4] National Taipei university of technology, Feedforward and Ratio Control [online]. 2011 [cit. 10.4.2013]. Available from: http://www.cc.ntut.edu.tw/~jcjeng/Feedforward%20and%20Ratio%20Control.pdf

[5] Sapienza University of Rome, Trajectory Planning [online]. 2011 [cit. 15.4.2013]. Available from: http://www.dis.uniroma1.it/~deluca/rob1_en/13_TrajectoryPlanningJoints.pdf

[6] Polytechnic Institute of NYU, Modeling, Identification,and Control of a DC-Servomotor [online]. 2011 [cit. 13.4.2013]. Available from: http://mechanical.poly.edu/faculty/slee/ME3411/Exp3.pdf

[7] KLIMEŠ, D.: Hardwarové a softwarové řešení diagnostiky a bezpečnosti provozu robotického manipulátoru, [Diplomová práca] Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 66 s., 2013

[8] Brushed DC motor Dunkermotoren GR/G [online]. 2011 [cit. 19.4.2013]. Available from: http://www.dunkermotoren.de/default.asp?id=9&lang=2

[9] DC motor with worm gear WLD43, Transmotec [online]. 2011 [cit. 10.4.2013]. Available from: http://www.transmotec.com/dc-motors/worm-gear/WLD43-Series.aspx

[9] ŠTĚPÁNEK, J.: Identifikace systému, senzorika a implementace řídicího algoritmu ne-stabilní balancující vozidlo, [Diplomová práce.] Brno: Vysoké učení technické, Fakulta strojního inženýrství, 62s., 2011

[10] University of Twente, Prototype modeling of mechanical systems [online]. 2011 [cit. 10.5.2013]. Available from: http://www.utwente.nl/ctw/wa/software/spacar/2011/protmod/protmod2011.pdf

[11] Cork Institute of Technology, Feed-Forward Control in Closed-Loop systems, [online]. 2011 [cit. 17.4.2013]. Available from: http://www.tommymoriarty.com/academic/Feedrward%20Control%20in%20ClosedLoop%20Systems.pdf

[12] MathWorks, Use Matlab GUIs with Simulink Models [online]. 2011 [cit. 10.4.2013]. Available from: http://blogs.mathworks.com/pick/2012/06/01/use-Matlab-guis-with-Simulink-models/

[13] MathWorks, Simulink-GUI Synchronization Example [online]. 2011 [cit. 11.4.2013]. Available from: http://www.mathworks.com/Matlabcentral/fileexchange/authors/76890

[14] MathWorks, GUI development for Simulink models [online]. 2011 [cit. 10.3.2013]. Available from: http://www.mathworks.com/Matlabcentral/fileexchange/32888-gui-development-for-Simulink-models

[15] Control Tutorials for Matlab and Simulink, Inverted pendulum: System modeling [online]. 2011 [cit. 1.4.2013]. Available from: http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

[16] Motion control systems, Linear and Rotary positioning stages Engineering reference [online]. 2011 [cit. 1.4.2013]. Available from: http://www.parkermotion.com/engineeringcorner/linearmechanics.html

[17] Vejlupek, J., Krejsa, J., Ripel, T.: *Platforma pro výuku modelování, identifikace a řízení dynamických soustav*, [Závěrečná zpráva projektu FRVŠ G1 1984/2012] Brno: VUT, Fakulta strojního inženýrství, 73s., 2013

# 10. List of figures

# 11. List of tables

# 12. List of abbreviations and symbols

FFC – Feedforward control
DOF – Degree of freedom
3xDOF – Three degrees of freedom
DH – Denavit Hartenberg
CoG – Center of gravity
CS – Coordinate system
CS0 – Coordinate system of the base
CS1 – Coordinate system of the arm 1
CS2 – Coordinate system of the arm 2
CS3 – Coordinate system of the arm 3
MSE – Mean square error
DC – Direct current
GUI – Graphical user interface
DS – Data store
RTWT – Real time windows target
P1 – Motor 1 in manipulator
P2 – Motor 2 in manipulator
P3 – Motor 3 in manipulator
$\alpha, a, \theta, r$ – Denavit Hartenberg parameters
LR2 – Lagrange equations of the second kind
L1, L2, L3 – Lengths of manipulator arms
b – Viscous friction constant
T – Coulomb friction constant
J, I – Rotor inertia
R – Resistance
x_T – Position of center of gravity
m_T – Mass of the pendulum
V – Voltage applied across DC motor
$v_s$ – Induced voltage
$\vartheta_1, \vartheta_2, \vartheta_3$ – Steering angles
$\alpha, \beta$ – Auxiliary angles
$c_\phi, k_\phi$ – Motor constants

# 13. Appendixes

Matlab – Simulink
1. PID and feedforward manipulator control models
2. PID and feedforward pendulum control models
3. Application for manipulator - Simulink programs and GUI m-files
4. Inverse kinematics module
5. Simmechanics inverse dynamics model
6. Motor P1, P2 and P3 simulation modules and m-files with constants
7. Equations of inverse dynamics model

Maple
1. Calculation of inverse dynamics model