ISSN 0280-5316
ISRN LUTFD2/TFRT--5834--SE

Modeling and control of a Delta-3 robot

André Olsson

Department of Automatic Control
Lund University
February 2009

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
February 2009

Document Number
ISRN LUTFD2/TFRT--5834--SE

Author(s)
André Olsson

Supervisor
Dr. Wolfgang Reinelt ELAU GmbH
Prof. Anders Robertsson Automatic Control, Lund
Prof. Rolf Johansson Automatic Control Lund (Examiner)

Sponsoring organization

Title and subtitle
Modeling and control of a Delta-3 robot. (Modellering och kontroll utav en Delta-3 robot)

Abstract
This Master Thesis describes the mathematical modeling of a Delta-3 robot actuated by motors and drive units developed by ELAU GmbH. A given model of the ELAU GmbH drive unit and motor is used when building the Delta-3 robot model including three drive units, one for each motor to be able to actuate the three upper arms.
The Delta-3 robot model is divided into kinematics and dynamics parts. The kinematics is used to calculate the trajectories for the three robot arms (joint space) and the corresponding motion of the robot travelling plate (Cartesian space). The Thesis also looks into the robot dynamics, so the coupling effect between the three arms is taken into account in the Simulink model. Different trajectories created with ELAU GmbHs own software are imported to Matlab workspace and simulated with the Simulink model. The results from the Simulink model are compared with the results from a real Delta-3 robot driven by ELAU GmbH hardware and software.
The Jacobian matrix for the Delta-3 robot is also calculated to be used in the equations for the coupling effect between the three arms. The Jacobian matrix is also implemented in ELAU GmbH software to be able to calculate the joint velocity or joint acceleration when the TCP velocity or TCP acceleration is known and vice versa. These results can then be used in equations for calculating the torque which is needed for each of the three motors to actuate the upper arms along with the desired TCP trajectory. The torque calculations can be done offline so the real robot don't have to be running, which gives the opportunity to see how much torque each motor needs so the robot is able to follow the desired trajectory for the robots travelling plate.
Experiments with comparison between the Simulink model and the real robot are done and the results of the measured values are shown in the Thesis. Also the experiments for the implementation of the Jacobian matrix in ELAU GmbH software are shown in the Thesis.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
71

Recipient's notes

Security classification

Acknowledgements

This Master thesis was carried out between the time August 2008 and February 2009 at ELAU GmbH in Marktheidenfeld, Germany. I would like to take the opportunity to thank all the colleagues at ELAU GmbH who gladly helped out with questions I had. Thanks to Jens, Christian, Mario, Carsten and Markus who helped me with my project.
I would like to thank Prof. Karl-Erik Årzén who made the contact with Dr. Wolfgang Reinelt at ELAU GmbH possible, Prof. Rolf Johansson my examinator at Lund University, Prof. Anders Robertsson my supervisor at Lund University. I would like to send my special thanks to my supervisor at the company Dr. Wolfgang Reinelt who helped me with my project. He also helped me to find an accommodation. Special thanks to Felix Böttcher who helped me with the measurements from the real Delta-3 robot. Special thanks to Reiner Volk who helped me with my project and introduced me to his choir. In the choir I got lots of new experience.
Finally I also would like to send my special thanks to my parents, brothers, relatives and friends back home in Sweden who kept the contact with me during my stay in Germany. Thanks mom, dad, Hans and Camilla for your visit in Marktheidenfeld, it meant a lot to me.

Marktheidenfeld, February 2009

TABLE OF CONTENTS

Acknowledgments iii

1 Introduction 3
1.1 Motivation 3
1.2 Software and Hardware used in Thesis 3
1.3 Notations 3
1.3.1 Acronyms 3
1.3.2 Variables and parameters 3

2 Background 6
2.1 Robotics 6
2.2 ELAU GmbH 7
2.2.1 PacDrive automation system – ELAU GmbH 7
2.2.2 Software – ELAU GmbH 8
2.2.3 SERCOS bus 8
2.2.4 Delta-3 Robot – ELAU GmbH 9

3 Kinematics – Serial Robot 10
3.1 Forward Kinematics 10
3.2 Inverse Kinematics 12
3.3 Velocity Kinematics 13

4 Kinematics & Kinetics – Delta-3 Robot 14
4.1 Model assumptions 14
4.2 Geometric Parameters 15
4.3 Forward Kinematics 16
4.4 Inverse Kinematics 17
4.5 Velocity Kinematics 19
4.6 Acceleration Kinematics 21
4.7 Actuator Dynamics 22
4.8 Inverse Dynamic model 22
4.8.1 Dynamic Parameters 23
4.8.2 Virtual work principle 24
4.8.3 Calculation of dynamic model based on virtual work principle 25
4.9 System Dynamics 26

5 Trajectory 28
5.1 Trajectory with desired start and end velocity 28

6 Trajectory – ELAU GmbH 30
6.1 Linear interpolation 31
6.2 Circular interpolation 32
6.3 Cubic spline 32

7 Control approach – ELAU GmbH 33
7.1 MC-4 Drive 34
7.2 Motor 34
7.3 Tracking Error 34
8 Simulink model 35
8.1 MC-4 36
8.2 Motor 36
8.3 Manipulator 37
9 Parameter estimation 38
10 Experiments 40
11 Results 41
11.1 Simulink model compared with real robot 41
11.1.1 Simulink Experiment 1 41
11.1.2 Simulink Experiment 2 46
11.1.3 Simulink Experiment 3 51
11.1.4 Discussion 54
11.2 Jacobian matrix Results in EPAS 55
11.2.1 Jacobian Experiment 1 55
11.2.2 Jacobian Experiment 2 58
11.2.3 Jacobian Experiment 3 62
11.2.4 Discussion 64
12 Conclusions 66
13 References 67

1 INTRODUCTION

1.1 MOTIVATION

The company ELAU GmbH which is the global Packaging Solution Center for Schneider Electric has developed an own Delta-3 robot controlled with ELAU GmbH own motors, drive units and controller unit.
This Master Thesis treats the modeling of the Delta-3 robot actuated with ELAU GmbH own motors and drive units. Also the kinematics for a Delta-3 robot is implemented to be able to see where the traveling plate has for position for different arm angle configurations.

1.2 SOFTWARE AND HARDWARE USED IN THESIS

To simulate the Delta-3 robot actuated with ELAU GmbH own controller, drive units and motors Matlab Simulink was used. To create trajectory motions for the Delta-3 robot to use in Simulink ELAU GmbH own programming software EPAS-4 was used. These trajectories was first simulated in EPAS-4 then transferred into Matlab Simulink so the same reference was used in both the Simulink model when simulating the robot and in EPAS-4 when controlling the real robot. When creating trajectories for only one motor simulation ELAU GMBHSs own motion toolkit ECAM-4 was used. The computer software Mathematica was used to solve the equations for kinematic transformation and the dynamics equations for the Delta-3 robot.

1.3 NOTATIONS

The following notations are used in the thesis.

1.3.1 ACRONYMS

The following acronyms are used:

TCP	Tool center point
DOF	Degree of freedom

1.3.2 VARIABLES AND PARAMETERS

The following variables and parameters are used in the thesis.

$\{R\}$	reference frame at the base plate
$\{R_i\}$	reference frame for arm i
${}^{R}_{i}R_{z}$	transformation matrix between frames $\{R_i\}$ and $\{R\}$
θ_1	angle of the first upper arm attached to motor 1
θ_2	angle of the second upper arm attached to motor 2
θ_3	angle of the third upper arm attached to motor 3

α_1	frame rotation angle for arm attached to motor *1*
α_2	frame rotation angle for arm attached to motor *2*
α_3	frame rotation angle for arm attached to motor *3*
l_A	length of upper arm
l_B	length of forearm
R_A	radius from base plate centre to one of the three arms
R_B	radius from traveling plate centre to one of the three forearms
J	Jacobian matrix for a Delta-3 robot
X_n	column vector, $(x,y,z)^T$ of TCP coordinates expressed in {R}
P_{ai}	point at the centre of the actuator axis
P_{bi}	point between the upper arm and the forearm
P_{ci}	point at the forearms connection with the traveling plate
P	vector *(x,y,z),* the forearms lower connection in simplified model
m_b	mass of upper arm
m_{fb}	mass of forearm
m_n	mass of traveling plate
m_{nt}	total contribution mass of traveling plate
m_c	mass of elbow between upper arm and forearm
r_{Gb}	mass centre point of upper arm
τ	column vector, $(\tau_1, \tau_2, \tau_3)^T$ of torques
k_r	gear Ratio for transmission between motor shaft and robot axis.

MC-4 drive unit

K_{Ppos}	P gain in the position controller
K_{Pvel}	P gain in the velocity controller
K_{Ivel}	I part in the velocity controller
K_{Pcurr}	P gain in the current controller
K_{Icurr}	I part in the current controller
FF_{curr}	current feed forward
F_{vel}	time constant for velocity filter
$F_{currRef}$	time constant for current ref filter
F_{curr}	time constant for current filter
CO	controller option, on or off
CT	cycle time at the SERCOS bus

Motor

i_{peak}	motors maximum current
i_{nom}	motors nominal current
RPM_{max}	maximum shaft rotation
Km	torque Constant
I_m	J_Internal
I_{Brake}	J_Brake
R	W_Resistance
L	W_Inductivity
z_p	pol pairs
F_v	viscous friction in motor

F_s	static friction in motor
GR_{in}	gear in
GR_{out}	gear out
I_{Load}	moment of inertia of load
τ_{load}	load force
CE	*c*urrent equipment

2 BACKGROUND

2.1 ROBOTICS

The word robot was introduced in 1921 by the Czech playwright Karel Capek in his satirical play R. U. R. (Rossums's Universal Robots), where he depicted robots as machines which resembled people but worked tirelessly (1).

The structure of a robot is usually mostly mechanical and can be called a kinematic chain. The system structures of a robot various depending on the task it will perform. A typically industrial robot is the serial robot which is built up by an open kinematic chain. This chain can be described as links connected to each other with revolute joints driven by actuators. This type of structure is often seen in the car industry.

The history of industrial automation is characterized by periods of rapid change in popular methods. Either as a cause or, perhaps, an effect, such periods of change in automation techniques seem closely tied to world economics. Along with computer aided design (CAD) systems, and computer aided manufacturing (CAM) systems the industrial robot became identifiable as a unique device, in the 1960's (2). The robots are growing in complexity and their use in industry is becoming more widespread. This leads to new structures with different advantages and disadvantages.

Parallel structures as the Delta-3 robot possess a number of advantages when compared with serial manipulators. They offer generally a much higher rigidity and smaller mobile mass than their serial counterparts. Thus they can move a much heavier payload relative to the components of body mass compared to a serial manipulator. The system structure of a Delta-3 robot also makes it more stiff perpendicular to the travelling plate (e.g. resistive against vibration perpendicular to the travelling plate). The major drawback of the parallel robots is their limited range of motion compared to a serial robot, but with a system structure as the Delta-3 robot this limitation has partially been resolved.

2.2 ELAU GMBH

2.2.1 PACDRIVE AUTOMATION SYSTEM – ELAU GMBH

The PacDrive automation system by ELAU GmbH is used to control different applications with the same software. The applications is programmed with help by the program EPAS-4 and then transferred into a PacDrive controller which controls the motions of the different drives in the system. Figure 2.1 shows the structure of the PacDrive system. The PacDrive controller is the brain of the system where the program code is stored, which in turn synchronize and sends information over a SERCOS bus to the different servo drives (MC-4 motor controllers) connected to the system. Each servo drive then controls the motion of the motor shaft using the information from the PacDrive controller.

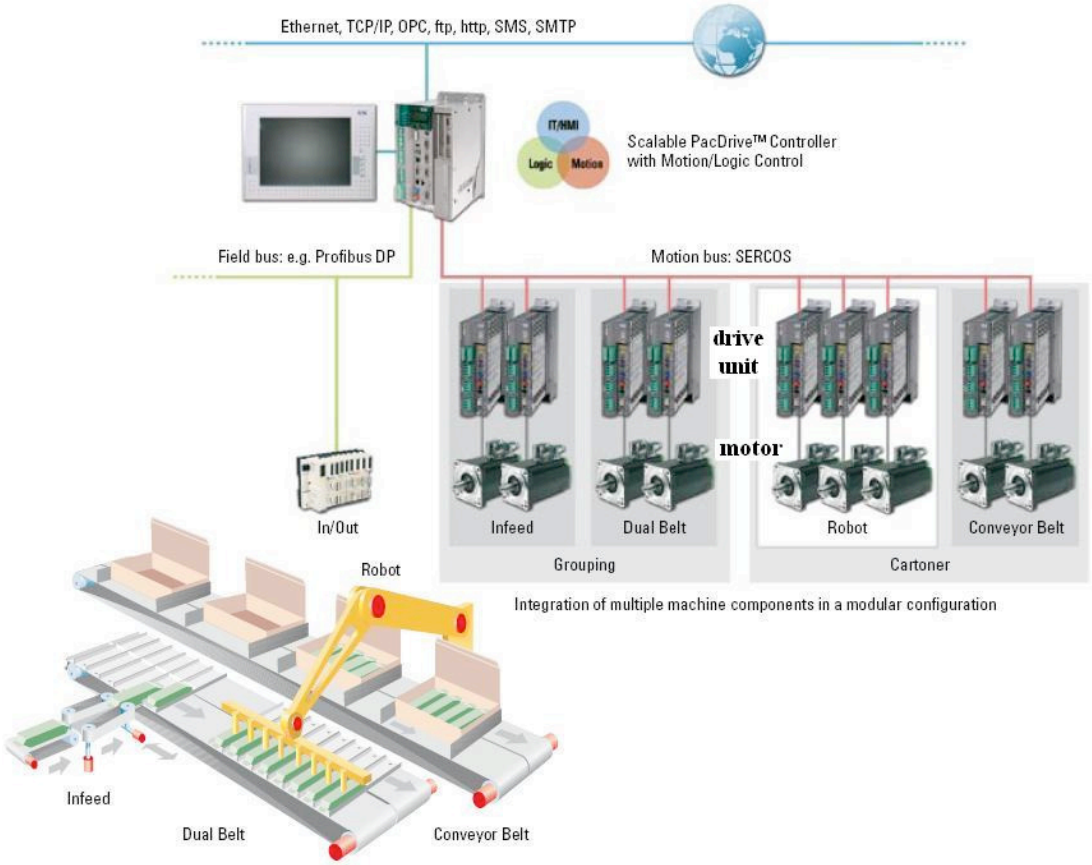


Figure 2.1 (3), Shows the architecture of a ELAU GmbH packaging system. The applications are transferred over Ethernet to a PacDrive controller which in turn can be monitored and controlled by a touch screen. The reference motion signal which is created in the PacDrive controller is transferred over a SERCOS bus to a drive unit which controls the motor based on information from the motors real shaft position.

2.2.2 SOFTWARE – ELAU GMBH

EPAS-4 is the software which includes different libraries for programming a PacDrive controller (e.g. C600 controller which is used for the Delta-3 robot). There are different libraries for the different products one want to control. In the robotic library there are function for the transformation of a path one want the TCP to follow to the actually angle coordinates for each a joint actuator.
After implementing the code in EPAS-4 software it has to be compiled and transferred into a PacDrive controller. From the PacDrive controller the information is send through the SERCOS bus to respectively system.

2.2.3 SERCOS BUS

SERCOS bus stands for SErial Real-time COmmunication System. The SERCOS interface is the only globally standardized (IEC 61491 and EN 61491) digital interface for communication between controllers and drives (4). It is a master-slave system.
The cycle time for the SERCOS bus can be chosen between 1ms, 2ms or 4ms.
The communication with the SERCOS is divided into two main options (4):

Cyclical communication
Non-cyclical communication

The two options can be used simultaneously.

Cyclical communication
Cyclical communication is executed once in every communication cycle and is used to exchange real-time data (e.g. reference values and actual values for the actuators of the Delta-3 robot). Certain specified data is transferred from the controller to all drives and from all drives to the controller in every cycle (4).
SERCOS enables the processing cycle of the controller to be synchronized with data exchange and the control cycle in the drives. Because of this there is no interference between these individual cycles and the control loops have a minimum, constant dead time in each drive. This also means that the new reference value goes into effect in all drives at the same time and that all bus slaves record the measure values that they forward to the PacDrive controller as actual values at the same time.

Non-cyclical communication
Non-cyclical communication is used to exchange data such as parameters for configuring communication (cycle time), the drive parameters, status etc. where time is not a critical factor. Non-cyclical communication is slower than cyclical communication since the data is transferred in portions. Typically, 10 ms (for 1 ms CycleTime) are required to read a variable (4).

2.2.4 DELTA-3 ROBOT – ELAU GMBH

ELAU GmbH has a Pick and Place Robot, PacDrive Robot P3, which transports products with high precision from the acceptance location to the target location. It can move products in a three dimensional Cartesian coordinate system. The combination of the constrained motion of the three arms connecting the traveling plate to the base plate ensues in a resulting 3 translatory degrees of freedom (DOF). As an option, with a rotating axis at the Tool Center Point (TCP), 4 DOF are possible.

The Robot consists of, consider Figure 2.2:

1. Actuators (3 motors)
2. Base plate
3. Upper robot arm
4. Lower robot arm (Forearm)
5. Rotation arm (optional, 4-DOF)
6. Travelling plate, TCP

The upper robot arms are mounted direct to the actuators to guarantee high stability. And the three actuators are rigidly mounted on the base plate with 120° in between. Each of the three lower robot arms consists of two parallel bars, which connects the upper arm with the travelling plate via ceramic ball joints. Lower frictional forces result from this. The wear reduces respectively as a result. To measure each motor shaft angle a SinCos encoder is used.

A fourth bar, rotational axes, is available for the robot mechanics as an option. The actuator for this axis is then mounted on the upper side of the robot base plate. The bar is connected directly to the tool and ensures for an additional rotation motion.

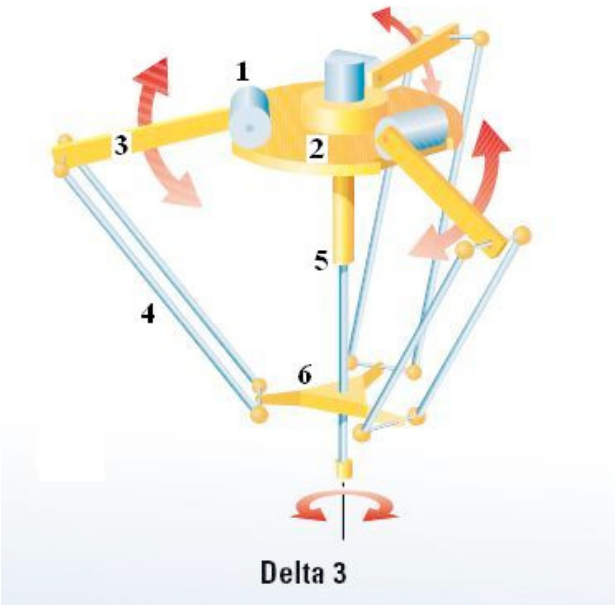


Figure 2.2 (3), ELAU GmbH Pick and Place robot, where the numbers in the figure are described in the text above.

3 KINEMATICS – SERIAL ROBOT

In the book *Introduction to Robotics Mechanics & Control*, kinematics is described as follows: "Kinematics is the science of motion which treats motion without regard to the forces which cause it. Within the science of kinematics one studies the position, velocity, acceleration and all higher order derivatives of the position variables, with respect to time or any other variable(s)" (2).

The kinematics for a parallel manipulator is too complicated to be present at this point. Therefore we will start with a two-link robot in two-dimension as in Spong and Vidyasagar (5) to give the reader more background how the kinematics works for an industrial robot. The kinematic for a Delta-3 Robot in three dimensional is calculated in chapter 4.
The kinematics for an industrial robot can be distributed in three different problem formulations, Forward Kinematics, Inverse Kinematics and the Velocity Kinematics. Consider Figure 3.1 as a starting-point for the introduced kinematics for a two-link serial industrial robot.

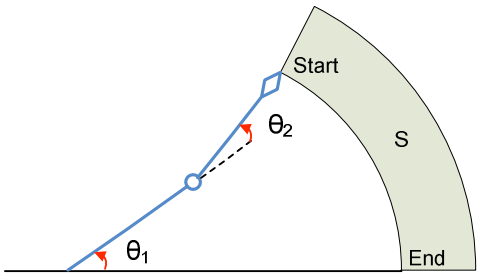


Figure 3.1, two-link serial robot with end-effector at starting point.

3.1 FORWARD KINEMATICS

The first problem is to describe both the position of the tool and the start and end location (also the surface between these two locations) with respect to a base frame, reference coordinate system.
Most commonly is to establish a fixed coordinate system, base frame, to which all objects including the manipulator are referenced. Then each new coordinate system for each joint can be expressed in the base frame with help by transformation matrixes (rotation matrixes).

Usually, the manipulator will be able to sense its own joint angles using some type of resolvers or encoders for each angle. With help of these angles the end-effector or tool position can be determined by the Forward Kinematics, which is to determine the end-effector or tool position in terms of the joint variables. In this case of a two-linked robot manipulator the base coordinate frame $o_0x_0y_0$ is established at the base of the robot as showed in Figure 3.2. Then the coordinates (x, y) of the tool can be expressed in the base frame with the two equations shown in Eq. 3.1 below.

$$\begin{cases} x = a\cos(\theta_1) + b\cos(\theta_2) \\ y = a\sin(\theta_1) + b\sin(\theta_2) \end{cases}$$

Eq. 3.1

Where a is the length of the lower arm, b is the length of the upper arm and θ_1, θ_2 are the angles for lower and upper arm respectively as shown in Figure 3.2.

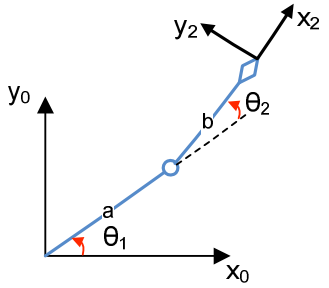


Figure 3.2, two-link serial robot.

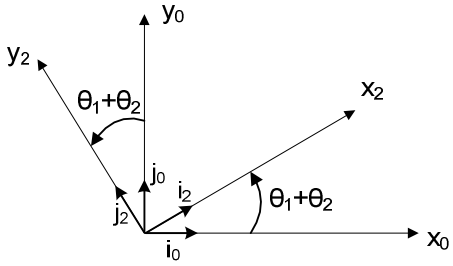


Figure 3.3, frame transformation between lower arm and tool frame.

The orientation of the joint frame between the lower and the upper arm, and the tool frame can be described with help of a transformation matrix relative the orientation of the base frame. Consider Figure 3.3 where $\boldsymbol{i_0}$ and $\boldsymbol{j_0}$ are orthonormal unit vectors in the base frame, and $\boldsymbol{i_2}$ and $\boldsymbol{j_2}$ are orthonormal unit vectors in the tool frame. Then the rotation of the tool frame can be described by an orientation matrix as in Eq. 3.2.

$$\begin{bmatrix} \vec{i}_0 \cdot \vec{i}_2 & \vec{j}_0 \cdot \vec{i}_2 \\ \vec{i}_0 \cdot \vec{j}_2 & \vec{j}_0 \cdot \vec{j}_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Eq. 3.2

These equations Eq. 3.1- Eq. 3.2 are called the forward kinematics equations in (5 p. 21) . But for a 3-DOF parallel manipulator these equations can be calculated so that only one angle is needed for each arm structure instead of two angles as for the two linked robot. This makes the equations a bit more complex and cannot be written as easily as for a two-linked manipulator.

3.2 INVERSE KINEMATICS

When the forward kinematics is solved we got the actually end-effector position for the angles, *(θ_1, θ_2)*. Then when the robot shall move from a point A to a point B the reference angles, *(θ_1, θ_2)* that moves the end-effector to the start point A position can be calculated with the forward kinematics. What we now need is which reference angles, *(θ_1, θ_2)* that the robot shall follow so that the end-effector moves to point B. This calculation is called the inverse kinematics. Take for example the two-linked planar robot, and a given *(x,y)* position that are within the end-effectors reach. Then there can be two solutions of the problem shown in Figure 3.4, elbow up and the elbow down, or if the manipulator has to be fully extended to reach the end point there will be only one solution. If the given position *(x,y)* are outside the end-effector reach there will be no solution to the inverse kinematics. There can also be infinite number of solution to the inverse kinematics, which occurs when the robot says be in a singular configuration. This is more described in velocity kinematics.

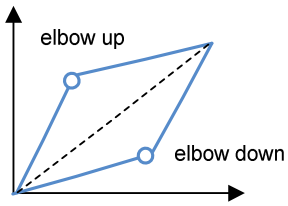


Figure 3.4, shows elbow up and elbow down for a two-link robot.

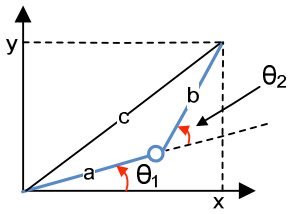


Figure 3.5, angles and arm lengths for a two-link robot.

To take into account both the elbow up and the elbow down position, the Law of cosine together with the trigonometry one can be used to calculate the angle θ_2. Consider Figure 3.5 for the following calculations:

$$\sin^2(\theta_2)+\cos^2(\theta_2)=1$$

Eq. 3.3

$$\left.\begin{aligned}\cos(\theta_2)&=\frac{x^2+y^2-a^2-b^2}{2ab}=D\\ \sin(\theta_2)&=\pm\sqrt{1-D^2}\end{aligned}\right\}\Rightarrow\theta_2=\tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D}\right)$$

Eq. 3.4

This approach gives two solutions for the θ_2 which covers both the elbow up and the elbow down solutions. Then the solution for the angle θ_1 can be calculated with help of the two solutions of θ_2 as follows:

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{b\sin(\theta_2)}{a + b\cos(\theta_2)}\right)$$

Eq. 3.5

where the coordinates *x* and *y* are the end-effectors position. Notice that θ_1 is depending on θ_2 and therefore will also have two different solutions.

3.3 VELOCITY KINEMATICS

The velocity kinematics describes the relationship between the end-effector and the joint velocities. This is useful when one want to follow a trajectory with the end-effector at constant velocity. In the example for a two linked robot the end-effector or the tool velocity can be calculated from the *x* and *y* coordinates that was calculated in the forward kinematics, Eq. 3.1.

$$\dot{x} = -a\sin(\theta_1)\cdot\dot{\theta}_1 - b\sin(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2)$$
$$\dot{y} = -a\cos(\theta_1)\cdot\dot{\theta}_1 - b\cos(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2)$$

Eq. 3.6

Expressed in vector form this gives

$$\dot{\mathbf{x}} = \begin{bmatrix} -a\sin(\theta_1)\cdot\dot{\theta}_1 - b\sin(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \\ -a\cos(\theta_1)\cdot\dot{\theta}_1 - b\cos(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \end{bmatrix}\dot{\boldsymbol{\theta}} = J\dot{\boldsymbol{\theta}}$$

Eq. 3.7

where the matrix *J* is the Jacobian matrix of the two-linked robot that maps joint velocity to end-effector velocity. To map the end-effector velocity to the joint velocity the inverse of the Jacobian can be used. There is a problem with this inverse mapping when the manipulator is in a singular configuration, which means that for the two-linked manipulator $\theta_2 = 0$, the two arms is in a straight line. This can be shown with the inverse of the Jacobian for the two-linked manipulator:

$$\dot{\boldsymbol{\theta}} = \frac{1}{ab\sin(\theta_2)}\begin{bmatrix} b\cos(\theta_1+\theta_2) & b\sin(\theta_1+\theta_2) \\ -a\cos(\theta_1)-b\cos(\theta_1+\theta_2) & -a\sin(\theta_1)-b\sin(\theta_1+\theta_2) \end{bmatrix}\dot{\mathbf{x}} = J^{-1}\dot{\mathbf{x}}$$

The determinant of the Jacobian shows that for $\theta_2 = 0$ or $\theta_2 = \pi$, there will not exist an inverse to the Jacobian matrix. Therefore a singular position of the robot has to be avoided.

4 KINEMATICS & KINETICS – DELTA-3 ROBOT

Unlike the more common serial-geometry mechanisms, all three of the primary actuators of this parallel mechanism work together to produce the three Cartesian translation degrees of freedom. For the fourth optional rotary axis on the TCP a fourth conventional rotary motor is added at the base plate to control the fourth, θ_r, degree-of-freedom.

4.1 MODEL ASSUMPTIONS

Because the Delta-3 robot is a closed loop manipulator as in Figure 4.1 it is more difficult to calculate the kinematics. To simplify the model and to reduce the number of parameters, the following assumptions are made (6):

The Travel plate always stays parallel to the base plate and its orientation around the axis perpendicular to the base plate is constantly zero. Thus, the parallelogram type joints (forearm) can be substituted by simple rods without changing the robot kinematic behaviour.

The revolute joints (between the base plate and the upper arms and between the forearms and the travelling plate) are identically placed on a circle, see Figure 4.1. Thus, the travelling plate can be replaced by a point *P* which the three forearms are connected to, see Figure 4.2.

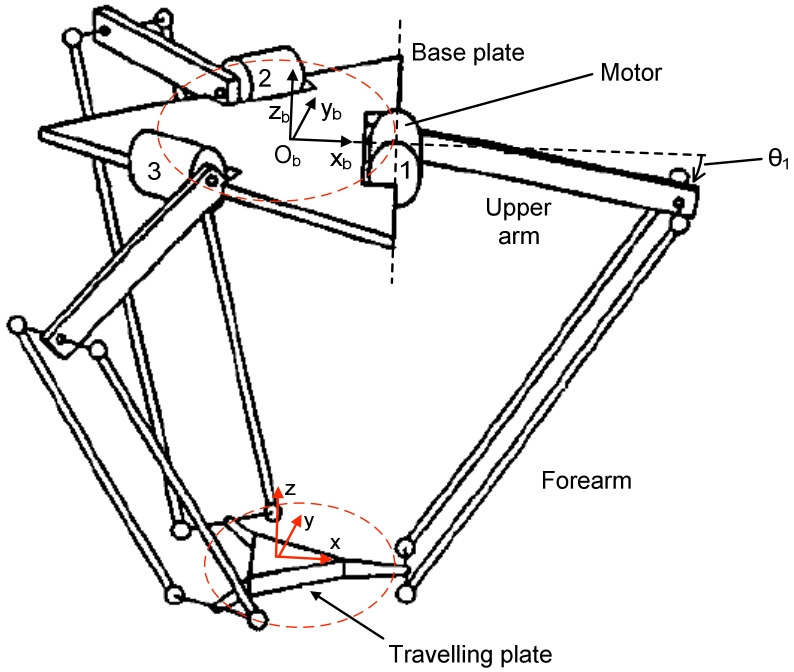


Figure 4.1 (6), describes a Delta-3 robot where the travelling plate always is parallel to the base plate. The frame at the base plate is the world frame.

4.2 GEOMETRIC PARAMETERS

The reference frame {R} is chosen as shown in Figure 4.1 at the centre of the circle with z pointing upwards and x perpendicular to the axis of motor 1. Because of the Delta-3 robots symmetry of the arms, each arm can be treated separately (7). The new model for one of the arms is shown in Figure 4.2. The index *i* (*i=1,2,3*) is used to identify the three arms. Each arm is separated by an angle of 120°. For each arm, a corresponding frame is chosen, located at the same place as the frame {R} but rotated $\alpha_1 = 0°$, $\alpha_2 = 120°$ and $\alpha_3 = 240°$ for the three arms respectively. The different frames $\{R_i\}$ can be described by a rotation matrix around z axis of the reference frame {R}. The rotation matrix ${}^{R}_{i}R_z$ is given by

$$ {}^{R}_{i}R_z = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. $$

The assumption above that the three forearms can be connected in one point, allows us to consider the distance from the reference frame {R} to one of the motors as $R = R_A - R_B$. Thus the forearms are connected in one point *P*. Each angle of each of the three upper arms has its initial value, 0°, parallel with the x axis of the frame $\{R_i\}$. The angle θ_i value is then increasing when the upper arm is moving downwards and decreasing when the upper arm is moving upwards. The parameter l_A represents the length of the upper arm and the parameter l_B represents the length of the forearm. With the information above, a direct- and an inverse-geometric model of the Delta-3 Robot can be established. These two models are also called the forward- and the inverse-kinematics.

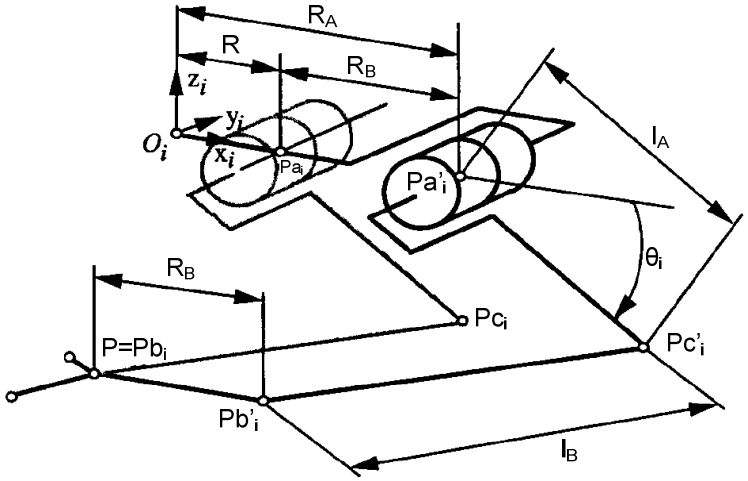


Figure 4.2, shows model simplification of the Delta-3 robot (7).

4.3 Forward Kinematics

The forward kinematics also called the direct kinematics of a parallel manipulator determines the *(x,y,z)* position of the travel plate in base-frame, given the configuration of each angle θ_i of the actuated revolute joints, see Figure 4.3.

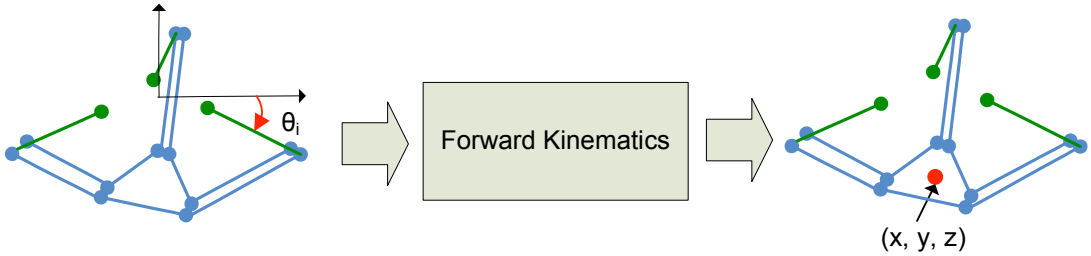


Figure 4.3, forward kinematics transforms the three arm angle positions θ_1, θ_2, θ_3 into the x,y and z components of the TCP position.

Consider three spheres each with the centre at the elbow (Pc_i) of each robot arm chain, and with the forearms lengths (l_B) as radius. The forward kinematic model for a Delta-3 Robot can then be calculated with help of the intersection between these three spheres. When visualizing these three spheres they will intersect at two places. One intersection point where z is positive and one intersection point where z coordinate is negative. Based on the base frame {R} where z-axis is positive upwards the TCP will be the intersection point when z is negative. Figure 4.4 shows the intersection between three spheres. Where two spheres intersects in a circle and then the third sphere intersects this circle at two places.

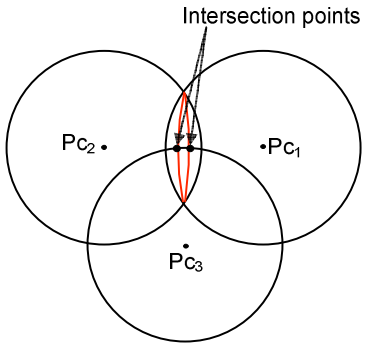


Figure 4.4, two spheres intersect in a circle and a third sphere intersect the circle at two places.

Based on the model assumptions made in chapter 4.1, one can take advantage that $R = R_A - R_B$ and the vector P_{C_i} that describes the elbow coordinates for each of the three arms as

$$P_{C_i} = \left[R + l_A \cos(\theta_i) \quad 0 \quad l_A \sin(\theta_i)\right]$$

To achieve a matrix that describes all of the three points P_{C_i} in the base frame {R} one has to multiply P_{C_i} with the rotational matrix ${}^{\mathrm{R}}_{\mathrm{i}}\mathrm{R}_{\mathrm{z}}$

$$ {}^{\mathrm{R}}_{\mathrm{i}}\mathrm{R}_{\mathrm{z}} = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

The result is the matrix P_C

$$ P_C = \mathrm{P}_{C_i} \cdot {}^{\mathrm{R}}_{\mathrm{i}}\mathrm{R}_{\mathrm{z}} = \begin{bmatrix} \cos(\alpha_1)(l_A\cos(\theta_1)+R) & -(R+l_A\cos(\theta_1))\sin(\alpha_1) & -l_A\sin(\theta_1) \\ \cos(\alpha_2)(l_A\cos(\theta_2)+R) & -(R+l_A\cos(\theta_2))\sin(\alpha_2) & -l_A\sin(\theta_2) \\ \cos(\alpha_3)(l_A\cos(\theta_3)+R) & -(R+l_A\cos(\theta_3))\sin(\alpha_3) & -l_A\sin(\theta_3) \end{bmatrix} $$

Then can three spheres be created with the forearms lengths l_B as radius, and their centre in P_{C_i} respectively.
The equation for a sphere is $(x-x_0)^2+(y-y_0)^2+(z-z_0)^2=r^2$ which gives the three equations

$$ \begin{cases} (x-[\cos(\alpha_1)(l_A\cos(\theta_1)+R)])^2+(y-[-(R+l_A\cos(\theta_1))\sin(\alpha_1)])^2+(z-[-l_A\sin(\theta_1)])^2 = {l_B}^2 \\ (x-[\cos(\alpha_2)(l_A\cos(\theta_2)+R)])^2+(y-[-(R+l_A\cos(\theta_2))\sin(\alpha_2)])^2+(z-[-l_A\sin(\theta_2)])^2 = {l_B}^2 \\ (x-[\cos(\alpha_2)(l_A\cos(\theta_3)+R)])^2+(y-[-(R+l_A\cos(\theta_3))\sin(\alpha_2)])^2+(z-[-l_A\sin(\theta_3)])^2 = {l_B}^2 \end{cases} $$

Eq. 4.1

With help from computer this equation system can be solved. There will be two solutions that describe the two intersection points of the three spheres. Then the solution that is within the robots working area must be chosen. With the base frame {R} in this case it will lead to the solution with negative *z* coordinate.

4.4 INVERSE KINEMATICS

The inverse kinematics of a parallel manipulator determines the θ_i angle of each actuated revolute joint given the *(x,y,z)* position of the travel plate in base-frame {R}, see Figure 4.5.

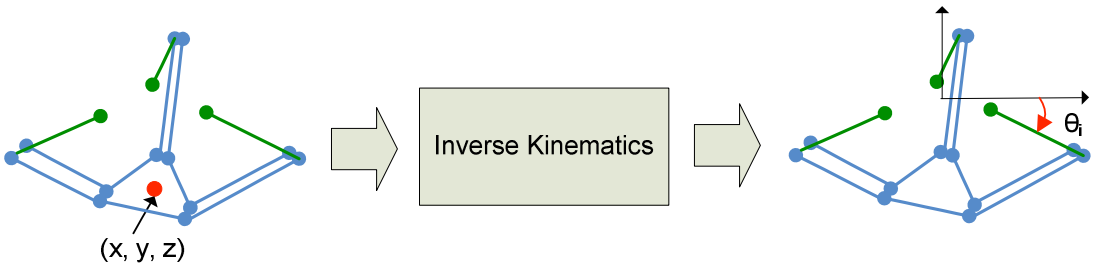


Figure 4.5, inverse kinematics transforms the robots TCP position (x,y and z component) into the three upper arm angle positions θ_1, θ_2, θ_3.

The inverse kinematics gives multiple solutions of the $\boldsymbol{\theta}$ vector with the three revolute joint angles that all satisfies the specific travel plate position. This may cause problems because the system has to be able to choose one of them. The criteria upon which to base a decision vary, but a very reasonable choice would be the closest solution (2), which is the solution where the manipulator moves the links as little as possible.
For the 3-DOF parallel manipulator with the system structure as in Figure 4.1, each arm can satisfies the same TCP with to different approaches which where called elbow up and elbow down for the serial robot. Together the three arms of the 3-DOF parallel manipulator result in eight different combinations of the $\boldsymbol{\theta}$ vector for a single goal, see Figure 4.6.

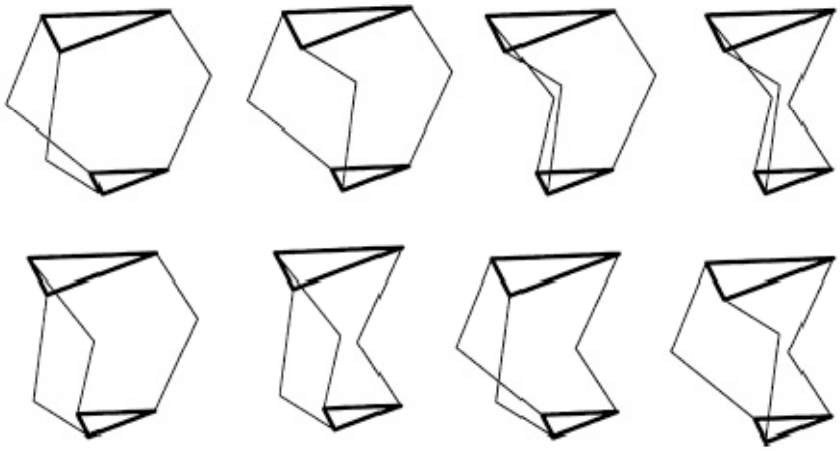


Figure 4.6, The eight solutions of the inverse kinematic for a Delta-3 robot (8).

The inverse kinematic model is obtained by using the three closure equations, constraints, of the kinematic chains:

$$\left\| P_{c_i} P_{b_i} \right\|_2^2 - l_B^2 = 0 \quad i = 1,2,3$$

Eq. 4.2

Then for each arm the angle is chosen so it is inside the angle constraints for the robot. This will convey to the first configuration, upper left corner in Figure 4.6.

4.5 VELOCITY KINEMATICS

The Jacobian matrix specifies a mapping from velocities in joint space to velocities in Cartesian space. The Jacobian determines a linearized matrix of the first derivatives.
To calculate the Jacobian matrix for a Delta-3 robot one can use a set of constraint equations linking the Cartesian space variables to the joint space variables (7). The three constraints equations for the Delta-3 robot can be chosen as

$$\left\| P_{c_i} P_{b_i} \right\|_2^2 - l_B^2 = 0 \quad i = 1,2,3$$

Eq. 4.3

assuming that the length of the forearms is constant.
Let s_i denote the vector $\mathrm{P}_{C_i}\mathrm{P}_{b_i}$ then can the Euclidean norm be written as $\mathrm{s_i^T s}_i$. Consider Figure 4.2 for the following calculations. The vector s_i can be written as

$$\mathrm{s}_i = O_i P_{b_i} - (O_i P_{A_i} + P_{A_i} P_{C_i}) = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} - {}^{\mathrm{R}}_{\mathrm{i}}\mathrm{R}_{\mathrm{z}} \left(\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_A \cos(\theta_i) \\ 0 \\ l_A \sin(\theta_i) \end{bmatrix} \right) \quad i = 1,2,3$$

Eq. 4.4

To introduce the first time derivative in joint space and in Cartesian space which the Jacobian matrix maps between one can take the time derivative of

$$\mathrm{s_i^T s}_i - l_B^2 = 0 \quad i = 1,2,3$$

Eq. 4.5

which gives

$$\mathrm{s_i^T \dot{s}}_i + \mathrm{\dot{s}_i^T s}_i = 0 \quad i = 1,2,3$$

Eq. 4.6

With help of commutativity property of the product the Eq. 4.6 can be written as

$$\mathrm{s_i^T \dot{s}}_i = 0 \quad i = 1,2,3$$

Eq. 4.7

where the first time derivative out of s_i is given by

$$\dot{s}_i = \begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{z}_n \end{bmatrix} - {}^{R}_{i}R_z \begin{bmatrix} -l_A \sin(\theta_i) \\ 0 \\ l_A \cos(\theta_i) \end{bmatrix} \dot{\theta}_i = \dot{X}_n - b_i \dot{\theta}_i \quad i = 1,2,3$$

Eq. 4.8

where

$$b_i = {}^{R}_{i}R_z \begin{bmatrix} -l_A \sin(\theta_i) \\ 0 \\ l_A \cos(\theta_i) \end{bmatrix} \quad i = 1,2,3$$

Eq. 4.9

For one robot arm the Eq. 4.7 can be written as

$$s_i^T \begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{z}_n \end{bmatrix} - s_i^T b_i \dot{\theta}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad i = 1,2,3$$

Eq. 4.10

which can be expressed in matrix form for all of the three robot arms as

$$\begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix} \dot{X}_n - \begin{bmatrix} s_1^T b_1 & 0 & 0 \\ 0 & s_2^T b_2 & 0 \\ 0 & 0 & s_3^T b_3 \end{bmatrix} \dot{\theta} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Eq. 4.11

From Eq. 4.11 the Jacobian matrix for a Delta-3 robot is obtained by considering

$$\dot{X}_n = J\dot{\theta}$$

Eq. 4.12

where

$$J = \begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix}^{-1} \begin{bmatrix} s_1^T b_1 & 0 & 0 \\ 0 & s_2^T b_2 & 0 \\ 0 & 0 & s_3^T b_3 \end{bmatrix}$$

Eq. 4.13

The Jacobian matrix J is not only depending of θ, as is usually the case for serial robots, but also a function of the TCP position X_n, which can be calculated with the forward kinematic model of the Delta-3 robot.

4.6 ACCELERATION KINEMATICS

The acceleration kinematics specifies a mapping from acceleration in joint space to acceleration in Cartesian space. To calculate this mapping one can use the result from the previous section 4.5 and derive Eq. 4.11 one more time to obtain the acceleration relationship. It is worth noting here that the time derivative of Eq. 4.11 is the derivative of a product.

Deriving Eq. 4.11 gives

$$\begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix} \ddot{X}_n + \begin{bmatrix} \dot{s}_1^T \\ \dot{s}_2^T \\ \dot{s}_3^T \end{bmatrix} \dot{X}_n - \left(\begin{bmatrix} s_1^T b_1 & 0 & 0 \\ 0 & s_2^T b_2 & 0 \\ 0 & 0 & s_3^T b_3 \end{bmatrix} \ddot{\theta} + \begin{bmatrix} \dot{s}_1^T b_1 + s_1^T \dot{b}_1 & 0 & 0 \\ 0 & \dot{s}_2^T b_2 + s_2^T \dot{b}_2 & 0 \\ 0 & 0 & \dot{s}_3^T b_3 + s_3^T \dot{b}_3 \end{bmatrix} \dot{\theta} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Eq. 4.14

Using the definition of Eq. 4.12 and Eq. 4.13 and rearranging Eq. 4.14, the following is obtained

$$\ddot{X}_n = \begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix}^{-1} \left(- \begin{bmatrix} \dot{s}_1^T \\ \dot{s}_2^T \\ \dot{s}_3^T \end{bmatrix} J + \begin{bmatrix} \dot{s}_1^T b_1 + s_1^T \dot{b}_1 & 0 & 0 \\ 0 & \dot{s}_2^T b_2 + s_2^T \dot{b}_2 & 0 \\ 0 & 0 & \dot{s}_3^T b_3 + s_3^T \dot{b}_3 \end{bmatrix} \right) \dot{\theta} + J\ddot{\theta}$$

Eq. 4.15

In Eq. 4.15, the time derivative of the Jacobian matrix $\dot{J}$ can be identified as the term multiplying $\dot{\theta}$. And finally, the relationship between the Cartesian acceleration and the acceleration in joint space can be expressed as

$$\ddot{X}_n = \dot{J}\dot{\theta} + J\ddot{\theta}$$

Eq. 4.16

4.7 ACTUATOR DYNAMICS

In most existing industrial robots, a transmission is introduced between each actuator and the corresponding articulated body of the manipulator. Such an actuated joint for one robot arm is illustrated in Figure 4.7.

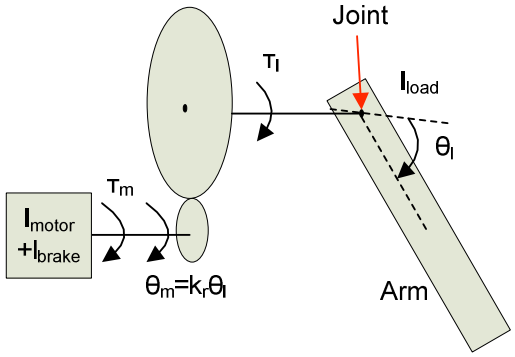


Figure 4.7, actuated joint with transmission between the joint and the motor.

On the assumption of a rigid transmission and with no backlash the relationship between the input forces (velocities) and the output forces (velocities) are purely proportional (9). This gives,

$$\theta_m = k_r \theta_l$$

Eq. 4.17

where θ_m is the motor shaft displacement, θ the robot joint displacement and the constant k_r is a parameter which describes the gear reduction ratio. I_{load} is the body moment of inertia about the rotation axis of an actuated link. τ_l is the load torque at the robot axis and τ_m is the torque produced by the actuator at the shaft axis. In view of Eq. 4.17 one can write

$$\tau_m = \frac{\tau_l}{k_r}$$

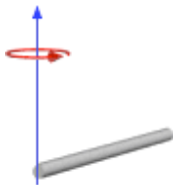
Eq. 4.18

4.8 INVERSE DYNAMIC MODEL

The inverse dynamic model of a parallel robot can be calculated with different methods (e.g. Lagrange or Newton-Euler).
A way to calculate the inverse dynamic model is to cut out the closed chain mechanism at passive joints and consider firstly the dynamics of the tree-structure robot thus created. The closure condition can then be relaxed by the application d'Alembert virtual work principle by mean of some Jacobian matrix.
For the Delta-3 robot, the complexity of the dynamic model arises mainly due to the movement of the forearms. This problem can be simplified if their rotational inertias are neglected. Thus

the force between the travelling plate and the arm is in a direction given by the orientation of the forearm. The model for the Delta-3 robot is calculated with Newton-Euler method with the following simplifying hypotheses (7):

> The rotational inertias of the forearms are neglected.
> For analytical purpose the masses of the forearms are optionally separated into two portions and placed at their two extremities. The fact that the inertia of a rigid rod of length *L* and mass *m* about one of the extremities is given by *I=1/3mL^2* shows that one can chose 1/3 of the forearms mass at its lower extremity (travelling plate) and 2/3 at its upper extremity (elbow), when consider the inertia at the elbow axis (10).



> Friction and elasticity effects are neglected.

4.8.1 DYNAMIC PARAMETERS

In the following parameters are used to calculate the dynamic model of the Delta-3 robot. They are listed here for convenience.
At the travelling plate only the total mass that is acting on it is used. This is the sum of the travelling plate mass m_n, mass of the payload $m_{payload}$, and the 3 reported masses contributed each of the 3 forearms $3(1\text{-}r)m_{fb}$.

$$m_{nt} = m_n + m_{payload} + 3(1-r)m_{fb}$$

r is the ratio of the mass of the forearms that is located at their upper extremities. As explained above in the simplifying hypotheses r is chosen to be equal to 2/3.
The position of the centre of mass for each of the upper arms is calculated using the centre of mass equation. Which is defined as the average of a systems particles positions r_i, weighted by their masses, m_i.

$$r_{Gb} = l_A \frac{\frac{1}{2}m_b + m_c + rm_{fb}}{m_{br}}$$

with

$$m_{br} = m_b + m_c + rm_{fb}$$

where m_b is the mass of the upper arm, m_c the mass of the elbow, m_{fb} the mass of the forearm, and r=2/3, the portion of m_{fb} that is placed at the elbow.

The inertia contribution of each upper arm is the sum of the motor inertia I_m, the motor brake inertia I_{brake} and the arm inertia I_{bc}. The motor inertia and the motor brake inertia can be expressed at joint level by multiply with the quadratic of the gear transmission constant. The Inertia at the joint level is then given by:

$$I_{bi} = I_m kr^2 + I_{brake} kr^2 + I_{bc}$$

where I_{bc} is the sum of the inertia created from the upper arm mass and the inertia created of the end point mass of the upper arm, which gives

$$I_{bc} = \frac{m_b}{3} l_A^2 + l_A^2 (m_c + rm_{fb}) = l_A^2 (\frac{m_b}{3} + m_c + rm_{fb})$$

4.8.2 VIRTUAL WORK PRINCIPLE

Virtual work on a system is the work resulting from either virtual forces acting through a real displacement or real forces acting through a virtual displacement. The term displacement may refer to a translation or to a rotation and the term force to a force or a moment. The principle does not refer to any geometric or inertial parameters and may therefore be considered as a fundamental connection between dynamics and kinematics.

Since any generalized forces system can be used, the equality of the virtual works associated to the two coordinates systems of interest in robotics yields:

$$\tau^T \cdot \delta\theta = \tau_n^T \cdot \delta X_n$$

Eq. 4.19

where τ is the force/torque vector corresponding to joint space virtual displacement $\delta\theta$ and τ_n is the force/torque acting on the traveling plate corresponding to the virtual displacement δX_n in Cartesian space.

When introducing the relationship between joint velocity and Cartesian velocity

$$\dot{X}_n = J\dot{\theta}$$

into Eq. 4.19 one can see that the Jacobian matrix can be used to transform the force/torques acting in Cartesian space to joint space as $\tau^T = \tau_n^T \cdot J$ which is equal to $\tau = J^T \cdot \tau_n$.
According to the hypothesis above one can reduce the robot to 4 bodies: the travelling plate and the three upper arms. Then the calculation of the contribution of torque/forces acting on traveling plate can be transferred to joint space with help of the Jacobian according to virtual work principle as shown above.

4.8.3 CALCULATION OF DYNAMIC MODEL BASED ON VIRTUAL WORK PRINCIPLE

Two kinds of forces are acting on the traveling plate. The gravity force G_n and the inertial force (d'Alembert) F_n. These two is given by:

$$G_n = m_{nt}\begin{pmatrix}0 & 0 & -g\end{pmatrix}^T \qquad F_n = m_{nt}\ddot{X}_n$$

The contribution of these two forces in joint space, can then be calculated with the transpose of the Jacobian matrix as described in section 4.8.2.

$$\tau_n = J^T F_n = J^T m_{nt}\ddot{X}_n \qquad \textbf{Eq. 4.20}$$

$$\tau_{Gn} = J^T G_n = J^T m_{nt}\begin{pmatrix}0 & 0 & -g\end{pmatrix}^T \qquad \textbf{Eq. 4.21}$$

Two kinds of torques are acting on the actuated joints from each arm. The torque τ_{Gb} produced by the gravitational force of each upper arm and the torque τ_b produced from the inertial (d'Alembert) force acting on each upper arm.
The gravitational contribution can be calculated as the force that is acting perpendicular to the upper arm through the mass centre point as shown in Figure 4.8.

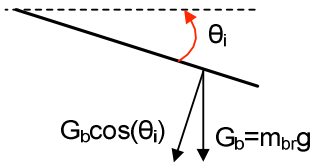


Figure 4.8, gravitational force acting on the upper arm of a Delta-3 robot.

This gives

$$\tau_{Gb} = r_{Gb}G_b\begin{pmatrix}\cos\theta_1 & \cos\theta_2 & \cos\theta_3\end{pmatrix}^T \qquad \textbf{Eq. 4.22}$$

where G_b is the gravitational force acting on the mass centre point of each upper arm. The torque contribution from each upper arm to the actuated joints can be expressed by:

$$\tau_b = I_b\ddot{\theta} \qquad \textbf{Eq. 4.23}$$

where I_b is the inertia matrix of the arms in joint space and is given by:

$$I_b = \begin{bmatrix} I_{b1} & 0 & 0 \\ 0 & I_{b2} & 0 \\ 0 & 0 & I_{b3} \end{bmatrix}$$

According to d'Alembert's principle the contribution of all the inertial forces must equal the contribution of all the non-inertial forces (11), this applied at the joint level leads to:

$$\tau + \tau_{Gn} + \tau_{Gb} = \tau_b + \tau_n$$

Eq. 4.24

where τ is the vector of torques that have to be applied to the three actuated joints. τ_n is containing the term $\ddot{X}_n$ which can be expressed in joint space by Eq. 4.16. This gives

$$\tau = (I_b + m_{nt} J^T J)\ddot{\theta} + (J^T m_{nt} \dot{J})\dot{\theta} - (\tau_{Gn} + \tau_{Gb})$$

Eq. 4.25

which can be written as:

$$\tau = A(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta)$$

Eq. 4.26

where *A* is the inertia matrix, *C* describes the accounting of the centrifugal and Coriolis forces and *G* contains the gravity forces acting on the manipulator.

4.9 SYSTEM DYNAMICS

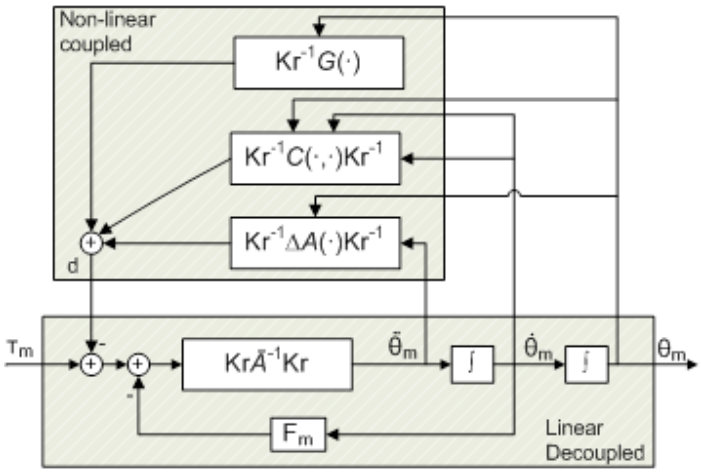


Figure 4.9, block scheme of manipulator with linear decoupled and non-linear coupled part.

As shown in section 4.8.3 the motion of a manipulator can be described by

$$\tau = A(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + F_m\dot{\theta} + G(\theta)$$

Eq. 4.27

where F_m is added and describes the friction coefficients of the manipulator.
By observing that the diagonal elements of $A(\theta)$ are formed by constant terms and functions of sine and cosine for revolute joints, one can set

$$A(\theta) = \overline{A} + \Delta A(\theta)$$

Eq. 4.28

where $\overline{A}$ is the diagonal matrix whose constant elements representing the average inertia at each joint (9). Substituting Eq. 4.17 - Eq. 4.18 into Eq. 4.27 yields

$$\tau_m = Kr^{-1}\overline{A}Kr^{-1}\ddot{\theta}_m + F_m\dot{\theta}_m + d$$

Eq. 4.29

where

$$F_m = Kr^{-1}(F_v + F_s)Kr^{-1}$$

Eq. 4.30

describes the matrix of viscous friction and static friction about the motor axis, and

$$d = Kr^{-1}\Delta A(\theta)Kr^{-1}\ddot{\theta}_m + Kr^{-1}C(\theta,\dot{\theta})Kr^{-1}\dot{\theta}_m + Kr^{-1}G(\theta)$$

Eq. 4.31

is the vector which represents the coupling effect between the three arms. θ_m is the angular variable of the motor and θ is the angular variable of the robot arm position.
As illustrated by Figure 4.9 the system of the manipulator with coupling effects can be constituted by two subsystems, one that has τ_m as input and θ_m as output and the other with θ_m, $\dot{\theta}_m$, $\ddot{\theta}_m$ as input and d as output. The first subsystem is a linear and decoupled system, since each component of τ_m only influences the corresponding component of θ_m. The other subsystem corresponds for all those non-linear and coupling terms between the three arms.

5 TRAJECTORY

A trajectory is a path that an object follows through space. In robotics the object could be the end-effector for a serial robot or the TCP for a Delta-3 robot. A trajectory can be described mathematically either by the geometry of the path, or as the position of the object over time. There are some parameters that have to be decided when a trajectory will be calculated. For example if the motion should be linear, quadratic or cubic, also if the motion should be done with some predefined start and end velocity, constant velocity or for example with minimum time. Then with this information a trajectory is calculated with help of interpolation. This trajectory contains then information about the speed, acceleration and position.

5.1 TRAJECTORY WITH DESIRED START AND END VELOCITY

One way to generate a smooth trajectory for an object, for example a time history of desired end-effector coordinates is by a polynomial (5). In this case there are some constraints such as the desired start and end velocities. If the variable *X* is a vector that describes the coordinates *(x,y,z)* for the end-effectors position. Then suppose that the object coordinates at time t_0 satisfies

$$X(t_0) = X_0$$
$$\dot{X}(t_0) = \dot{X}_0$$

Eq. 5.1

and the final values for time t_f satisfies

$$X(t_f) = X_f$$
$$\dot{X}(t_f) = \dot{X}_f$$

Eq. 5.2

Because there is four constraints the generated polynomial has to contain four independent coefficients that can be chosen to satisfy these four constraints. This will give a cubic trajectory of the form

$$X_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Eq. 5.3

where $X_d(t)$ is the desired *(x,y,z)* positions at time *t*.

Then the generated velocity that will satisfy the start and end constraints will be given by the derivative of $X_{desired}$ as

$$\dot{X}_d(t) = a_1 + 2a_2 t + 3a_3 t^2$$

Eq. 5.4

and at last the acceleration that makes this smooth trajectory possible will be

$$\ddot{X}_d(t) = 2a_2 + 6a_3 t$$

Eq. 5.5

The four independent coefficients a_0, a_1, a_2, a_3 can be calculated by combining the Eq. 5.3, Eq. 5.4 and the four constraints Eq. 5.1 and Eq. 5.2. This then yields four equations with four unknowns that can be written in matrix form

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} X_0 \\ \dot{X}_0 \\ X_f \\ \dot{X}_f \end{bmatrix}$$

Eq. 5.6

It can be shown that the determinant of the 4x4 coefficient matrix in Eq. 5.6 is $(t_f - t_0)^4$. This shows that the Eq. 5.6 always has a unique solution as long as $t_0 \neq t_f$ is satisfied (5).
Finally for given t_0, t_f and the four constraints as in Eq. 5.1 and Eq. 5.2 a general solution for the trajectory can be calculated as

$$X_d(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3$$

Eq. 5.7

where

$$a_0 = X_0 \qquad a_1 = \dot{X}_0$$

$$a_2 = \frac{3(X_1 - X_0) - (2\dot{X}_0 + \dot{X}_1)(t_f - t_0)}{(t_f - t_0)^4} \quad a_3 = \frac{2(X_0 - X_1) + (\dot{X}_0 + \dot{X}_1)(t_f - t_0)}{(t_f - t_0)^3}$$

Eq. 5.7 can also be used to planning a sequence of movement between different points. For example the end value for the move to the *i*-th point is used as initial value for the move to the *i+1*-th point and so on.

6 TRAJECTORY – ELAU GMBH

C600 is the PacDrive controller that is used to control the Delta-3 robot. The calculations for the trajectory are done in two parts in the C600, see Figure 6.1. The first part (UnitRobot) is done in the non-cyclical communication described in chapter 2.2.3. In this part a virtual motion is calculated for the TCP in *(x,y,z)* direction. The second part (Transformation task), cyclical communication (chapter 2.2.3), then calculates the TCP initial position in the first cycle with help of the Delta-3 robots forward kinematics. After the first cycle the inverse kinematics is used to transform the virtual *(x,y,z)* position into angles positions of the robot arms. These angles are then forwarded to the three MC-4 drives which control the three motors, respectively. The forward kinematics is also calculated in each cycle time to see the Cartesian tracking error.

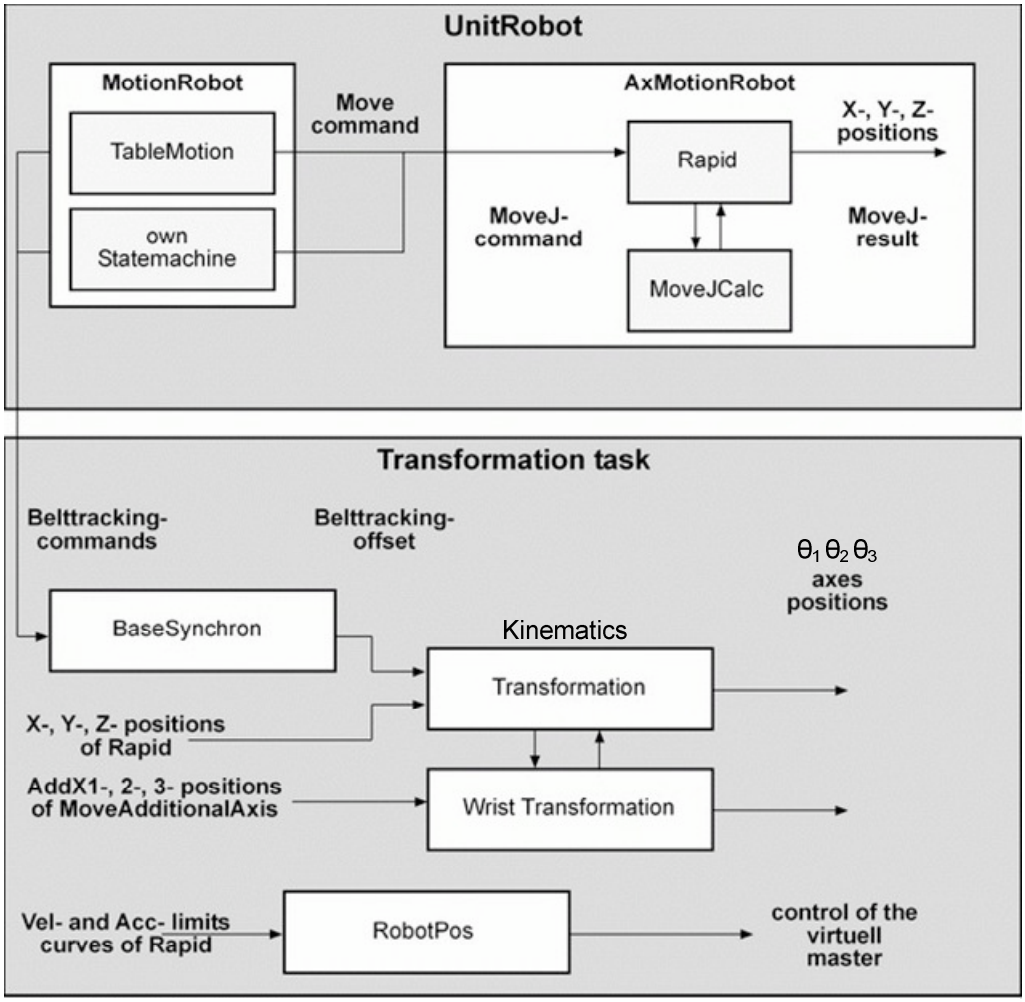


Figure 6.1, describes the calculations in C600 where axes positions are the angles of the robot arms (12).

The virtual motions for *x*, *y* and *z* in the first part can be seen as slaves that follows a virtual master that describes the distance of the TCP motion at time t, see Figure 6.2. Then for each cycle the master decides which value from the slaves that will be used for the inverse kinematics.

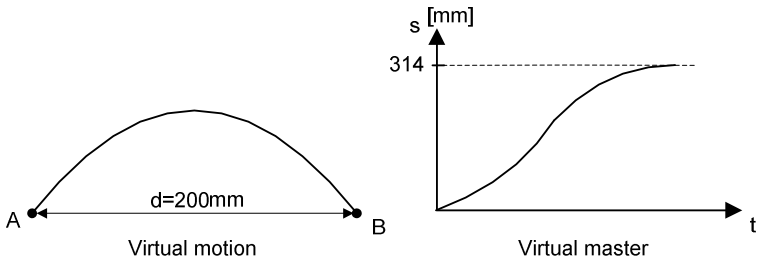


Figure 6.2, shows TCP motion from A to B and the virtual master which describes the distance of this TCP motion.

There exists different predefined blocks in the EPAS-4 package library to create different motions in a *(x,y,z)* coordinate system.

6.1 LINEAR INTERPOLATION

Linear interpolation is used when one want to move from a point A to a point B linearly. In ELAU GmbH EPAS-4 robotic library this function is called MoveL.
Maximum velocity is determined by the user then is velocity- and position path calculated out from this data. The velocity will be ramped up to its maximum value then ramped down to zero where the path ends, see Figure 6.3.

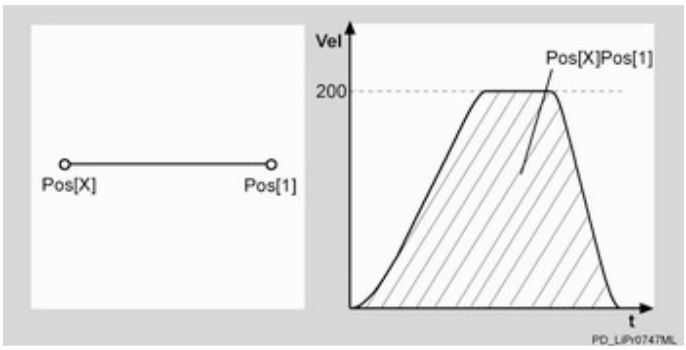


Figure 6.3, Shows position- and velocity path for linear interpolation function MoveL in EPAS (13).

6.2 CIRCULAR INTERPOLATION

In the ELAU GmbH EPAS-4 library there is a function called MoveC. This function needs the following parameters:

Start point, A – Start position for the TCP
End point, B – End position for the TCP
Cirlce point, C – Point between start and end points
Max TCP velocity – The maximum velocity of the TCP when moving along the path

The function moves the TCP on a circular path from the starting position A via the arc point C to the target position B. Between the end and start point the curve is divided into several points, see Figure 6.4. Then the distance between every point at the curve is interpolated using cubic spline.

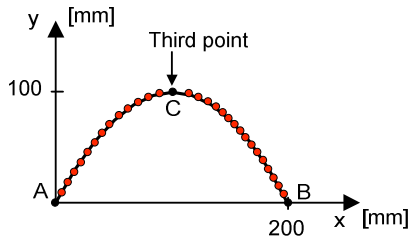


Figure 6.4, shows circle spline with the motion between point A to point B divided into several points which are in turn interpolated with cubic spline.

The virtual master determines the velocity of the TCP for each cycle. To contain this velocity the x, y and z have different velocities so at the end the TCP trajectory has the desired velocity.

6.3 CUBIC SPLINE

Splines represent an alternative approach to data interpolation. In polynomial interpolation, a single formula, given by a polynomial, is used to meet all the data points. The idea of splines is to use several formulas, each a low degree polynomial, to pass trough the data points (14). A cubic spline interpolates between the given data points by degree 3 polynomial. It is used when one want to make a various path with interpolation between many points. In ELAU GmbHs robotic library is this function called MoveSB, and can move the robot in a three-dimensional spline from the start position across a maximum of 100 points to the target position. The cubic spline calculates the smoothes path between each of the data points so at the end a smooth trajectory will be held.

7 CONTROL APPROACH – ELAU GMBH

One idea to controls the Delta-3 robot is independent joint control. This strategy regards the manipulator as formed by three independent systems (the three joints) and controls each joint axis as a single-input/single-output system. This is done by in the PacDrive controller C600 calculate the TCP motion with some of the blocks described in chapter 6 depending on which motion that is sought. Then the inverse kinematics is used to transform the TCP position into angle positions for the three arms that are forwarded from the C600 into respectively MC-4 drive every time cycle. The MC-4 also returns an actual value of the three arm angles every time cycle. These returned angle values are only used to calculate how big the tracking error is. The forward kinematics is also used to calculate the initial position of the TCP so the motion from the initial position to the start position of the motion can be calculated. Then during each time cycle the Cartesian tracking error is calculated with help of the forward kinematics.

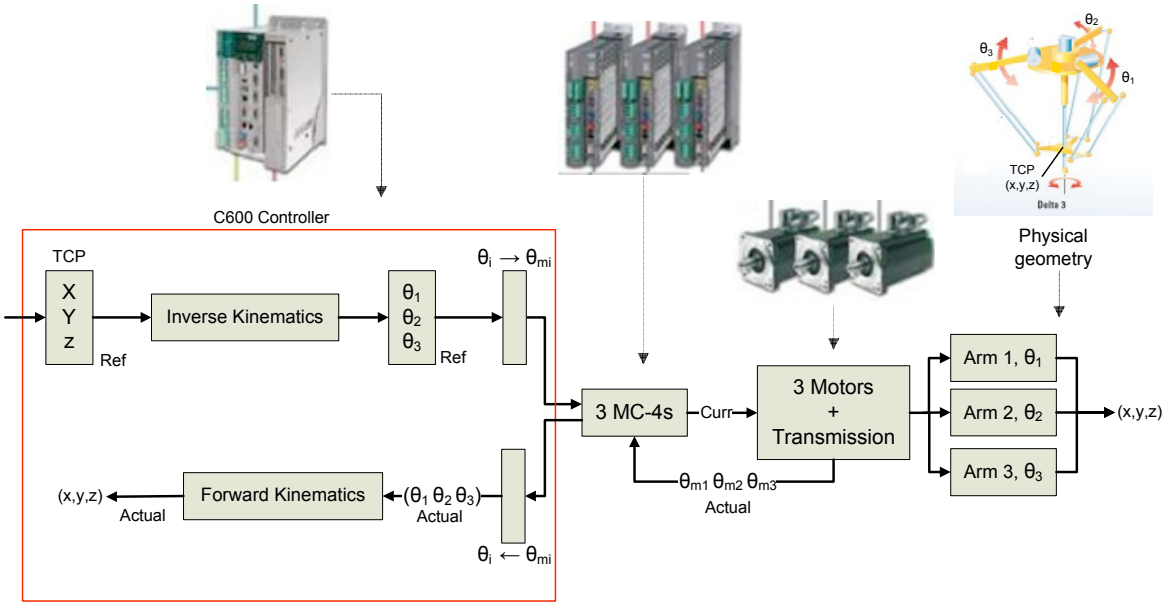


Figure 7.1, shows a C600 PacDrive controller connected to three MC-4s . The three MC-4s controls one motor each to actuate the three arms at the Delta-3 robot.

Figure 7.1 shows an overview of the Delta-3 robot system where each block is associated to a picture from the real EPAS system. The desired TCP motion for the robot acts as input into the C600 controller which with the kinematics for the Delta-3 robot calculates the desired joint positions to fulfill the desired TCP motion. Before the three reference joint positions are send to respectively MC-4 the joint positions are transformed into the three motor angles, respectively. This is shown in the figure with an empty box in the figure with associated text. The last block Arm1, Arm2 and Arm3 in the figure describes the Delta-3 robots real arms so the last x, y and z components of the TCP position are obtained by the real robots physical geometry.

7.1 MC-4 DRIVE

The MC-4 drive is a server amplifier used to control the motors that are used in the PacDrive system. The controller is build up as a cascade controller with feedback loops of the position, velocity and the current. Also feedforward of the position, velocity and the current are used to reduce the tracking error.
The MC-4 can only take a position signal as input and can send back a position signal to the PacDrive controller (e.g. C600 controller). For the Delta-3 robot one can consider Figure 4.9 where the three MC-4 drives controls shaft position i ($i=1, 2,$ 3) corresponding to the single-input/single-output system of the decoupled and linear part. The SERCOS interface is used for communication between the MC-4 and the PacDrive Controller (e.g. C600).

7.2 MOTOR

The motors that are used to actuate the three arms of the Delta-3 robot are high dynamic brushless synchronous AC servomotors. The torque is formed from the stator winding fed with a sine three-phase current system in interaction with the magnetic field energized by the rotor magnet. The three-phase current system is generated dependent of the rotor position in the digital servo amplifier. The rotor position is determined using an integrated measuring system such as SinCos encoder (15).

7.3 TRACKING ERROR

The controller, C600 calculates the tracking error from the slave reference positions and the actual position of the robot. Since the actual position of the robot is not available at the same time as the reference position, the actual robot position is extrapolated linearly with an assumed constant velocity. The tracking error of the reference signal and the actual signal is called following error in EPAS.

8 SIMULINK MODEL

The simulink model structure is divided into different parts such as trajectory generator, robot kinematics, robot dynamics, MC-4 drive unit and motor. Figure 8.1 shows where the different Simulink model part can be found for the real robot controlled with ELAU GmbH equipment. The trajectory which one want the robots TCP to follow is generated with ELAU GmbHs program EPAS. The trajectory is simulated and sampled in EPAS and then imported into Matlabs workspace. A simulink block is created where the trajectory data from Matlabs workspace is imported into the trajectory generation block in the simulink model. The output from this block, the desired arm angles are now recalculated to the motors shaft angle and then send as input to the MC-4 model.
To be able to move the Delta-3 robots TCP after a desired trajectory one have to know respectively upper robot arm angle for each different TCP position. In the simulink model this is achieved with an implemented inverse kinematic block, which takes the three TCP direction components x, y and z as input and gives out the three upper arm angles. These three upper arm angles are recalculated to the respectively motor shaft position (rotation), which is acting as input signal to the three MC-4s. To move each arm of the Delta-3 robot there is a MC-4 that takes a position signal as input and then controls the motor to follow this position.

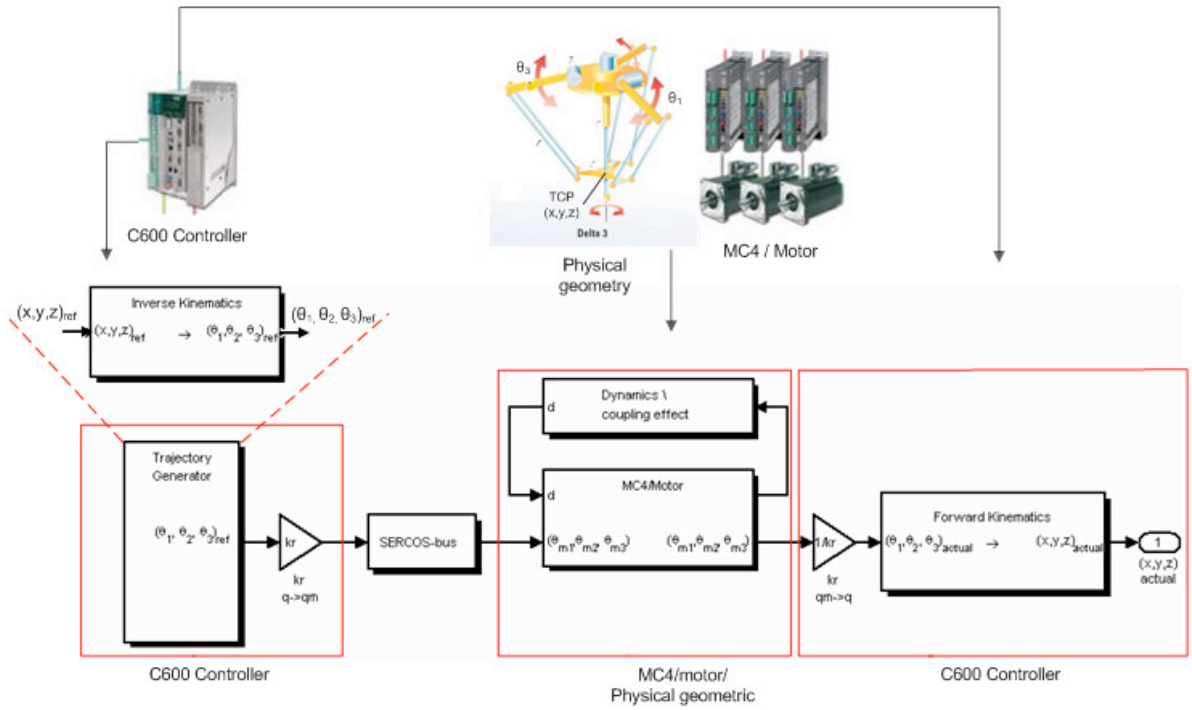


Figure 8.1, describes where the different parts of the simulink model are found for the real robot controlled with ELAU GmbH equipment. To the left in the figure the TCP reference signal is acting as input and to the right the actual TCP value is calculated with the forward kinematics. The middle of the figure shows the three MC4s each connected to a motor and also the dynamic block which simulates the coupling effect between the three arms of the Delta-3 robot.

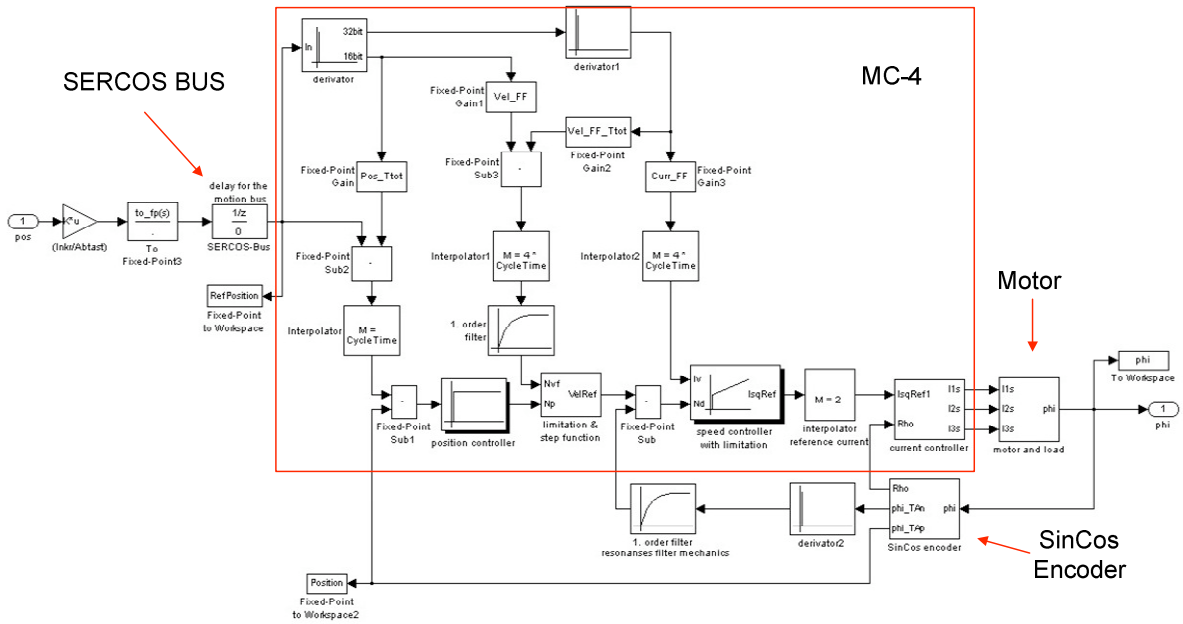


Figure 8.2, Simulink model with motor shaft position in radians as input and gives the actual motor shaft position out in radians. Between the reference and actual value of the motor shaft position the SERCOS bus and one MC-4 connected to one motor is shown. Also the Encoder which measures the motor shaft position is simulated a SinCos Encoder.

8.1 MC-4

The MC-4 is the servo amplifier that is used to control the motor shaft position. For the Delta-3 robot one can consider Figure 4.9 where the three MC-4 drives controls shaft position i ($i=1,2,3$) corresponding to the single-input/single-output system of the decoupled and linear part. The model of one MC-4 and one motor is shown in Figure 8.2. The SERCOS bus is simulated as a delay of the cycle time, which is 1ms for the Delta-3 robot simulation. Before the SERCOS bus there is a gain which increments the input signal with 2^{19}.
As shown in Figure 8.2, a cascade controller is used in the MC-4. The cascade controller is build up by one outer loop which controls the shaft position with a P-controller and two inner loops which controls the velocity and the acceleration, respectively. In the two inner loops PI-controllers are used to reduce the steady state error. Also position, velocity and current feedforward compensation is added to reduce the tracking error.

8.2 MOTOR

The motors that are used to actuate the Delta-3 robot arms are ELAU GmbH motors. The motor type is AC synchronous motors which are modelled in the simulink model as shown in Figure 8.3. To get more information how to derive a motor model in Simulink, see (16 pp. 7-24).

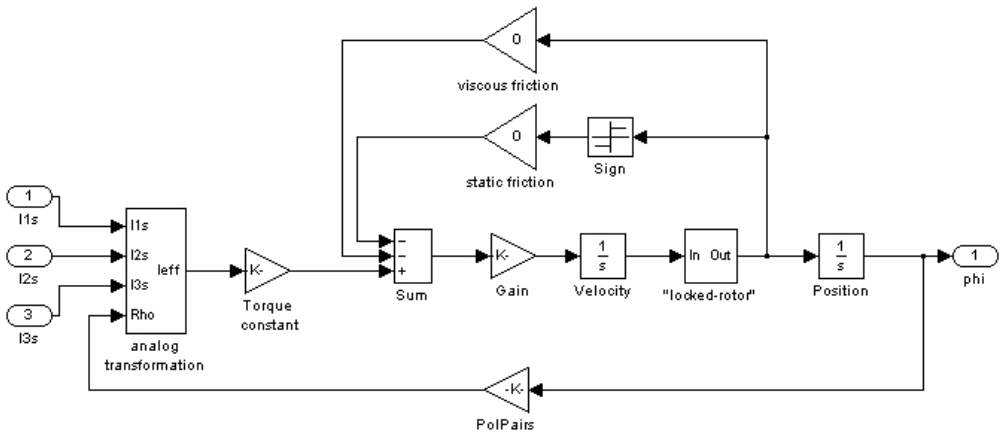


Figure 8.3, AC motor model in Simulink. The three currents *I1s*, *I2s* and *I3s* is output from the PWM stage in the current controller and are transforming to an effective current I_{eff} which is acting as input to the torque constant gain.

8.3 MANIPULATOR

To simulate an independent joint control of the Delta-3 robot, one can use three separate MC-4 motor models (Figure 8.2) in Simulink. The inverse kinematics is used to calculate the robots reference arm angle positions. These signals are recalculated to the corresponding motor shaft rotation and then used as input to the MC4 model. The output shaft rotation position from the motor model is recalculated to its corresponding robot angle position and used in the forward kinematics to become the TCP position of the robot structure. The interaction between the arms is described by component *i* of the vector d in Eq. 4.31. Where the *i*-th component is coupled to the *i*-th motor as shown in Figure 8.4.

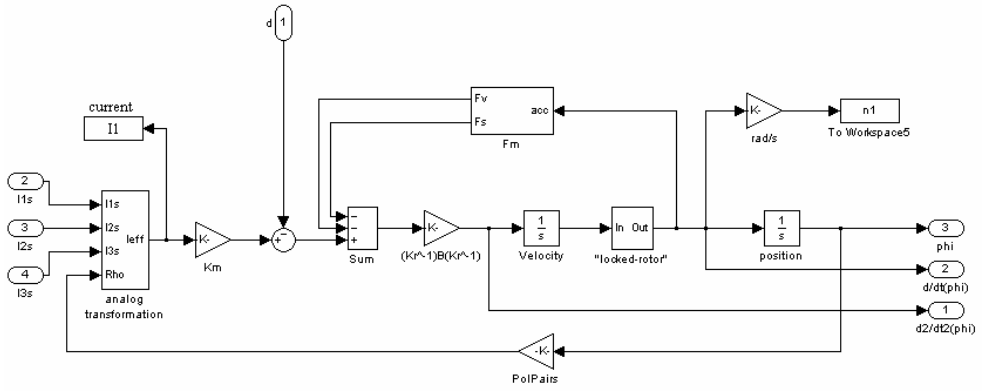


Figure 8.4, AC motor model with arm couple effect as input.

9 PARAMETER ESTIMATION

To simulate a Delta-3 robot in Simulink one needs to choose parameters so the model becomes as close as possible to the real robot. Therefore it is good to divide the parameters into different parts for the different equipment used to control the robot. P1, P2, P3 and P4 contain the parameters used in the different parts shown in Figure 9.1.

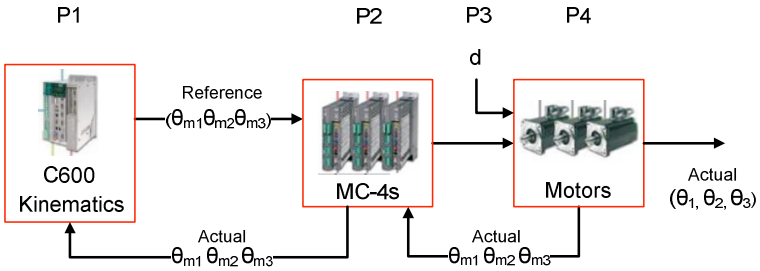


Figure 9.1, P1, P2, P3 and P4 are the parameterization for the different blocks. Where "d" is the robot dynamics.

The parameters in P1 are given from the robot manufacture and are depending on the robots physically structure such as the upper arms, forearms, base plate and the travelling plate. These parameters are needed to be able to calculate the inverse and forward kinematics for the robot. The parameters in P2 are given from the drive unit MC-4 which ELAU GmbH use to control their motors. The configurations of these parameters are the same as the configuration in the real MC-4s which are used at the real robot. P3 contains values given from the real robot structure such as the weight of the different parts of the robot. P3 also contains the parameters given in P1 and the torques given to the motors. The parameters in P3 are used in chapter 4.8.3 to be able to obtain the coupling effect between the arms which is given in Eq. 4.31.
The parameters in P4 are given from the motor model which is used to simulate the motors which ELAU GmbH is using. The parameters are given from the data sheets from the motor which is used on the real robot.
The parameters in P4 are also estimated with the Matlab function *lsqnonlin* to try to obtain better dynamics in the Simulink motor model. *lsqnonlin* is used to solve nonlinear least-squares curve fitting problems. In this case the curve fitting problem is between the Simulink model output and the real robots output. Matlab tries to estimate the parameters which the user has given as initial value so the curve fitting problem becomes minimized. But this function in Matlab can't find any better values for the parameters in P4 to be able to include more of the dynamics from the real motor. The Matlab function only returns the same values as the initial value. This could depend on that the nonlinear least-square curve fitting problem already is in a minimum. And when Matlab tries to change the parameters it only finds values which makes the non-linear curve fitting problem bigger instead of smaller.
Table 9-1 shows an overview of the parameters which are included in the different parameterization parameters P1 (Kinematics), P2 (drive unit MC4), P3 (dynamic for real robot) and P4 (Motor parameters).

For explanation of the parameters in Table 9-1 see section 1.3.2 Variables and parameters.

Parametrization

P1 (Kinematics)	P2 (MC-4)	P3 (Dynamics)	P4 (Motor)
l_a	K_{Ppos}	l_a	i_{peak}
l_b	K_{Pvel}	l_b	i_{nom}
R_A	K_{Ivel}	R_A	RPM_{max}
R_B	K_{Pcurr}	R_B	Km
α_A	K_{Icurr}	α_A	I_m
α_B	FF_{curr}	α_B	I_{Brake}
α_C	F_{vel}	α_C	R
	$F_{currRef}$	m_b	L
	F_{curr}	m_{fb}	z_p
	CO	m_n	F_v
	CT	$m_{payload}$	F_s
		m_c	GR_{in}
		r_{Gb}	GR_{out}
		τ	I_{Load}
		k_r	τ_{load}
			CE

Table 9-1, the variables P1, P3 and P4 which contains the parameters as shown in the table can be found in Eq. 4.1, Eq. 4.2 , Eq. 4.27, Eq. 4.31. The parameters in P2 can be found in the simulink model.

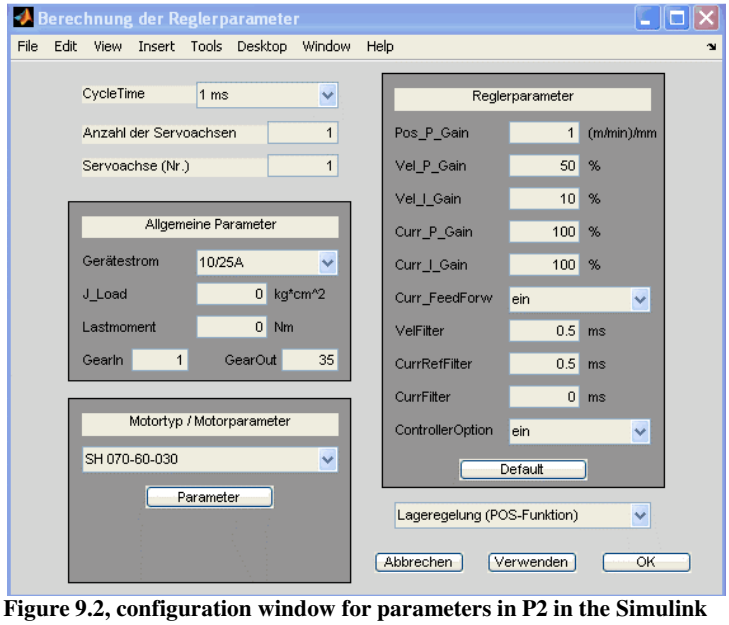


Figure 9.2, configuration window for parameters in P2 in the Simulink model.

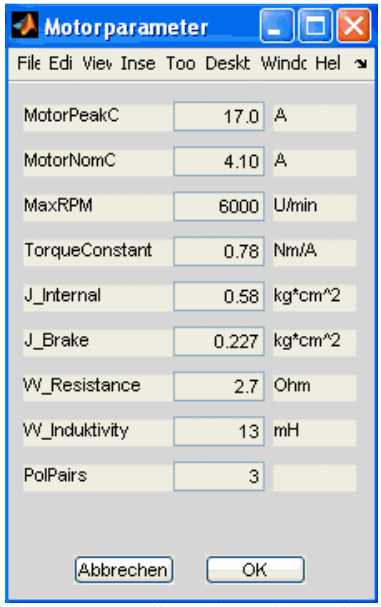


Figure 9.3, configuration window for parameters in P4 in the Simulink model.

10 EXPERIMENTS

For the experiments in the Thesis the Simulink model for the Delta-3 robot is used and compared to different measured values from a real Delta-3 robot. The Simulink model that is used is described in chapter 8. The real robot that is used has the physical geometrics as in the parameters P1 and P3 which are listed in chapter 9. The three motors used to actuate the three upper arms for the Delta-3 robot are ELAU GmbHs own constructed motors with the parameters as in P4. The three motors are each controlled with ELAU GmbHs drive unit MC-4, and these three MC-4 drive units are in turn coupled to ELAU GmbHs PacDrive controller C600. The program EPAS GmbH is used to trace the sampled actual values which are given from the MC-4s to the C600 and the reference values which are calculated in the C600. The TCP position and the three upper arm angles are calculated with help of the forward and inverse kinematics transformations for the Delta-3 robot.
The trajectories which are sampled with EPAS GmbH are imported into Matlabs workspace so they can be used as reference trajectories in the Simulink model.
The signals which are compared between the obtained measured values from the EPAS GmbH program and the measured values in the Simulink model are listed below:

Simulink model	**Real robot**
Actual Arm Angle position	Actual Arm Angle position
Actual Arm Angle velocity	Actual Arm Angle velocity
Actual Arm Angle acceleration	Reference Arm Angle acceleration
Current I_{eff} into the motors	Current I_{eff} into the motors
Actual TCP position	Actual TCP position
Norm of TCP error	Norm of TCP error

Table 10-1, shows which measured signals that are compared between the Simulink model and the real robot.

The measured values such as actual arm angle position, actual arm angle velocity, reference arm angle acceleration, current into the motors and the TCP position, from the real robot are sampled with EPAS trace function. The actual arm angle acceleration from the simulink model is only compared with the reference acceleration given from EPAS while the actual value of these parameters can't be measured in EPAS.
The Simulink model is then simulated with the same reference trajectories as for the real robot and the actual measured values in the Simulink model are compared to the measured values for the real robot.

Three different experiments are taking into account in the Thesis where the different results from the Simulink model and the real robot are compared and discussed in the next chapter, Results. The Jacobian matrix implementation in EPAS is also tested for the same trajectories as used in the three experiments for the Simulink model and the real robot. In all experiment the parameters P1, P2, P3 and P4 has been used.

11 RESULTS

11.1 SIMULINK MODEL COMPARED WITH REAL ROBOT

To be able to compare the simulink model with the real robot one has to import the actual values generated with the real robot into Matlab workspace. This is done by running the real robot with EPAS and then sample the actual data measured from the real robot. After the measured values from the real robot are imported to Matlab workspace the Simulink model is simulated. During the simulation are the corresponding values which have been measured for the real robot measured in the Simulink model. The Matlab command plot is then used to compare the measured values in Simulink and for the real robot.

11.1.1 SIMULINK EXPERIMENT 1

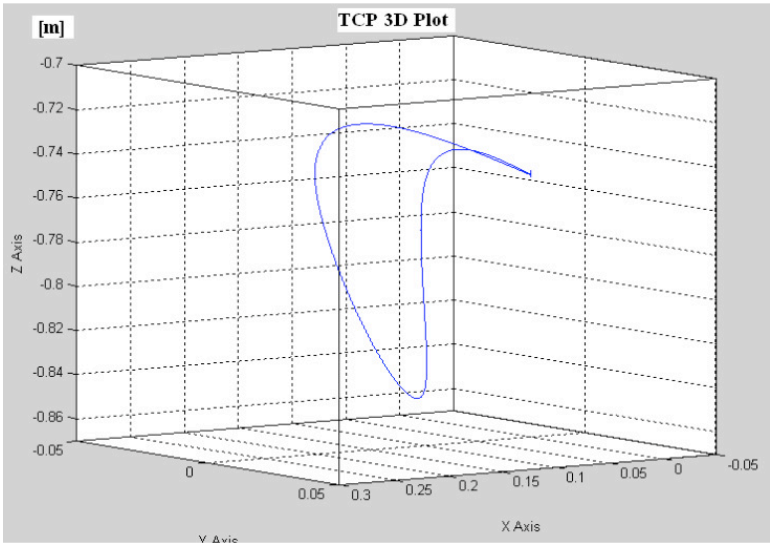


Figure 11.1, describes the motion of the TCP when simulating with the parameters, P1, P2, P3, P4.

The 3D figure describes the TCP motion used at the real robot and in the simulink model to compare the different measured signals such as arm angles, arm velocities, motor currents and the TCP motion calculated from the measured arm angles with the forward kinematics. The TCP motion in experiment 1 starts at (0,0,0.75)*m* and makes then a pick and place motion with the y component for the TCP constant. The z – component is first making a movement up over -0.74*m* and then turning downward to make a pick. The motion ends at the same point as it started in, (0,0,0.75)*m*.

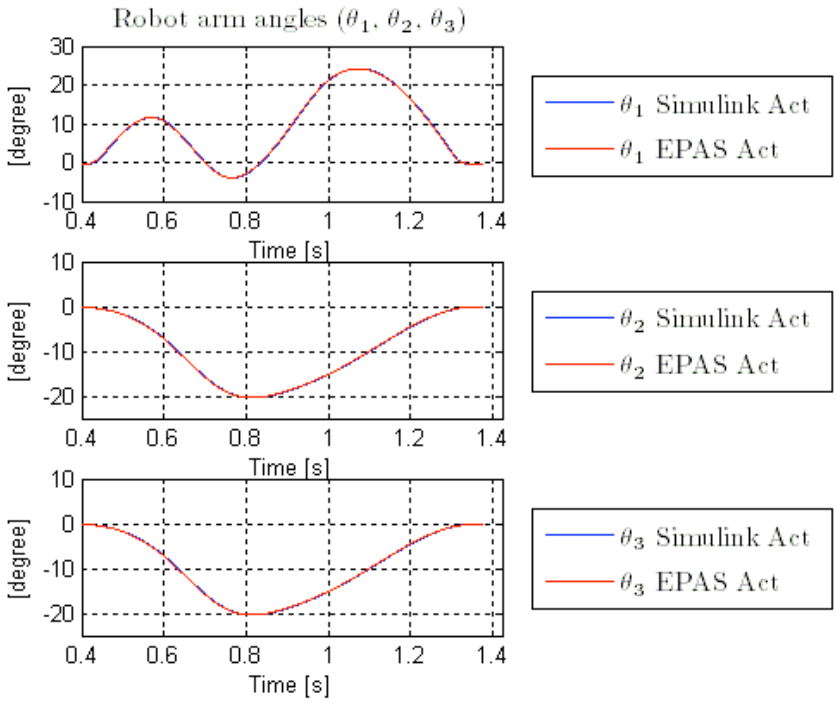


Figure 11.2, shows the motion of the three arm angles for the Delta-3 robot with the TCP motion as in Figure 11.1.

The measured arm angles in the simulink model are compared with the measured arm angles from EPAS for the real robot in Figure 11.2. The units at the y-axis are in degrees and are shown over the time in seconds. The blue line describes the measured actual value from the three upper arms which are connected to the three motors. The red line describes the measured actual value for the real robot sampled in EPAS.

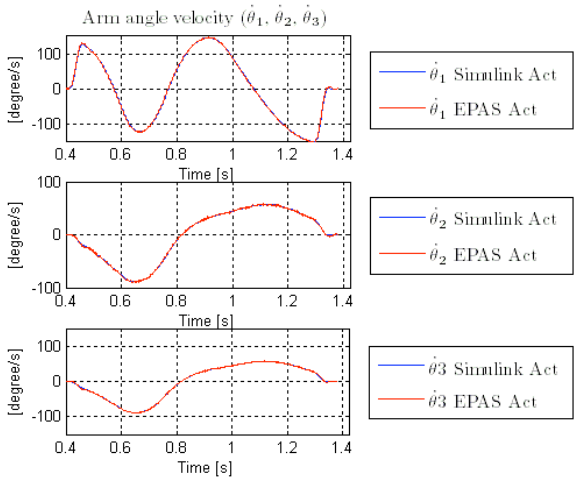


Figure 11.3, shows the velocity for the three arm angles. Compares Simulink model with real robot controlled with EPAS.

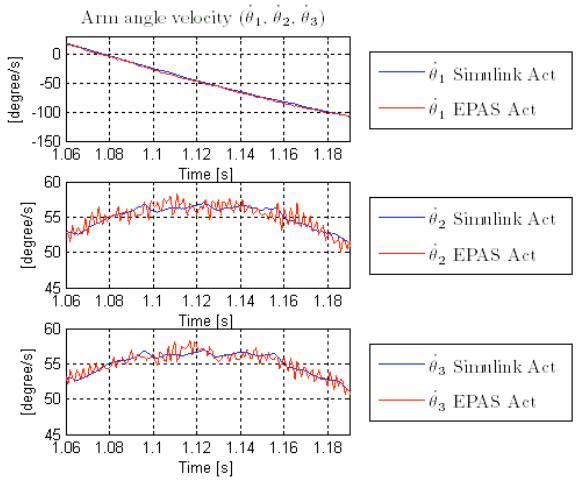


Figure 11.4, shows a zoom of Figure 11.3 of the time interval 0.95s to 1.1s.

The velocity graph above shows the angle velocity in degrees per seconds for the three upper robot arms which are connected to the three motors at the base plate, see Figure 2.2. The blue

line describes the measured actual value in the simulink model and the red line the measured actual value from the real robot. One can see in the right plot that the measured velocity from the simulink model and the real robot match well. But the right plot which shows a zoom from the left plot for the time interval 1.06s to 1.19s, one can see that the simulink model follows the real robots measured value well except from the small amplitude variation which the real robot has.

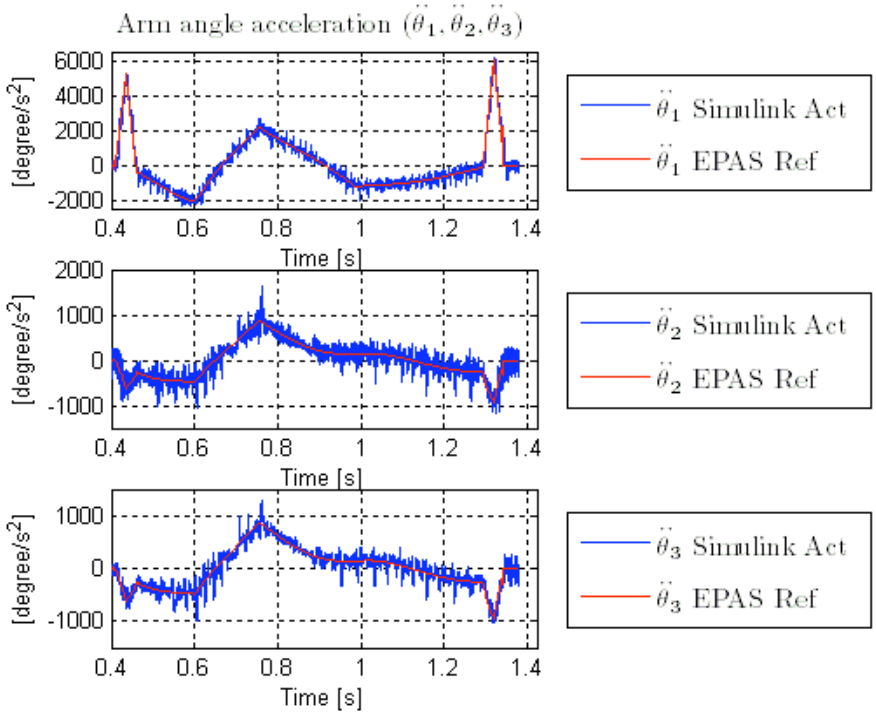


Figure 11.5, shows the acceleration for the three arm angles.

In the acceleration plot above are the three measured upper arm angles accelerations from the simulink model compared to the reference upper arm angles accelerations value in EPAS. This is because only the reference value for the arm acceleration can be measured in EPAS and not the actual value. The plot shows still that the mean value for three upper arm angle accelerations of the Delta-3 robot in the simulink model follows the measured reference acceleration value in EPAS well. The arm angle accelerations are shown in degrees per square seconds over the time in second.

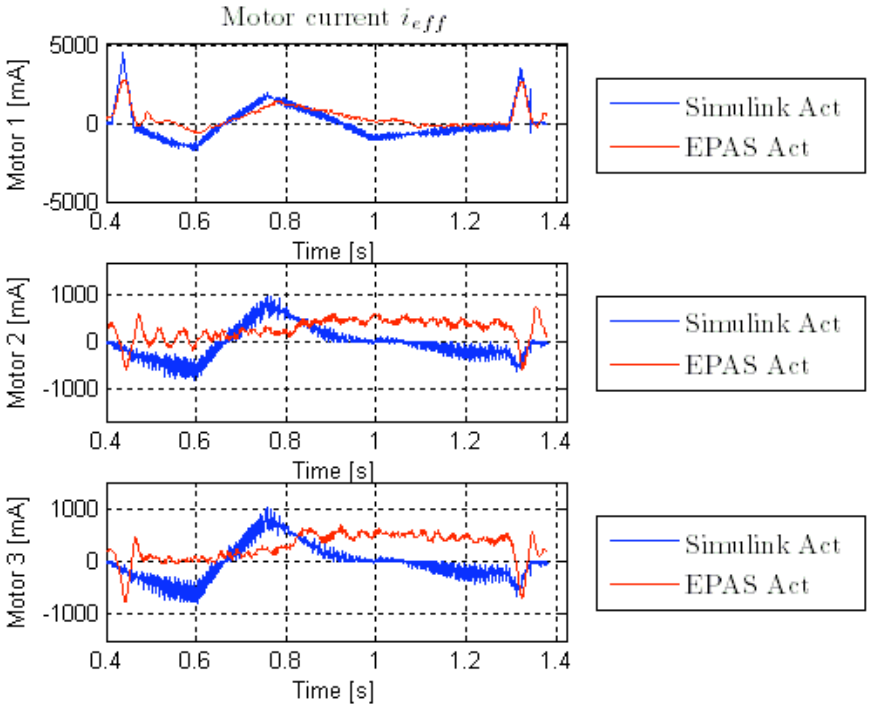


Figure 11.6, shows the current I_{eff} for the three motors.

In Figure 11.6 are the three currents shown which acts as input to each of the three motors. The blue line represents the measured current I_{eff} in the Simulink model, see Figure 8.4 and the red line represents the measured current from each real motor sampled in EPAS.

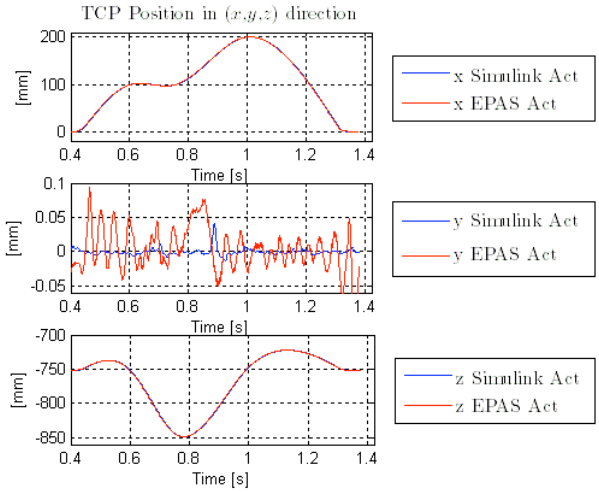


Figure 11.7, shows the TCP motion in x, y and z components of the Delta-3 robots TCP position.

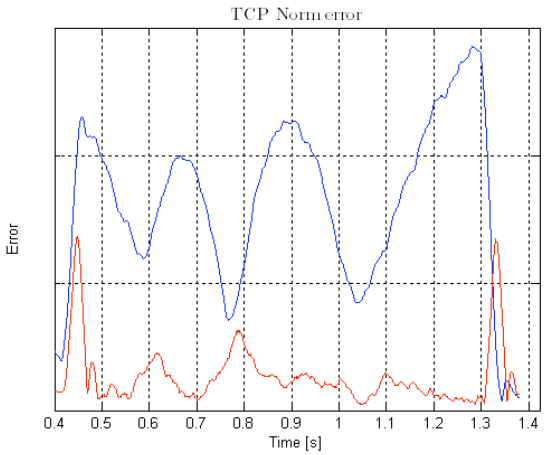


Figure 11.8, shows the norm of the TCP error. blue line is simulink models error and red line is TCP error from real robot controlled with EPAS.

Figure 11.7 above shows the x, y and z components of the TCP position. The blue line represents the measured actual value in the simulink model and the red line represents the

measured actual value from the real robot. The units at the y-axis are *mm* and the x-axis units are in seconds.
Figure 11.8 shows the two norm of the deviation between the measured actual TCP position and the reference TCP position over the time in seconds. The blue line shows the TCP norm error for the simulink model and the red line are for the real robot.
In Figure 11.7 it seems that the real robot will have a larger norm error than the simulink model because the real robot has a larger deviation from the real reference value in y direction which is zero. What makes that the real robot has a smaller norm error than the simulink model is that in x and z direction the simulink model has a larger deviation from its reference value than the real robot has. Therefore when one look at the right plot one can see that the simulink model has a larger norm error than the real robot. The deviation between the real robots norm error and the simulink models norm error can be explained by some dynamics missing in the simulink model.

11.1.2 SIMULINK EXPERIMENT 2

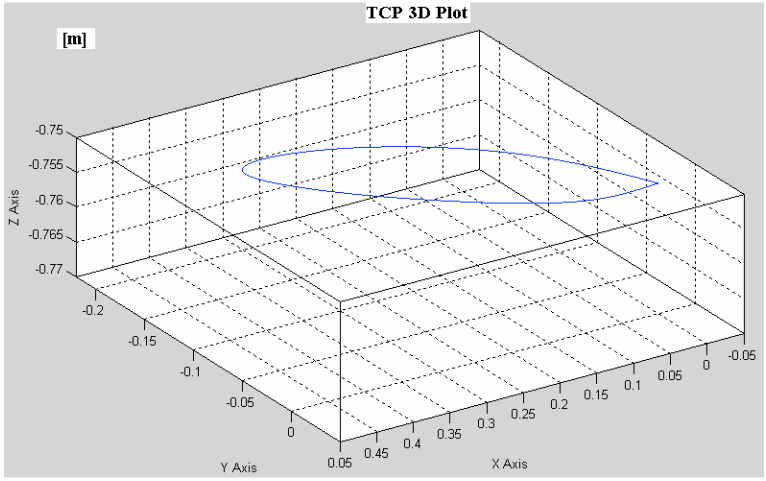


Figure 11.9, describes the motion of the TCP when simulating with the parameters, P1, P2, P3, P4.

The 3D figure describes the TCP motion used at the real robot and in the simulink model to compare the different measured signals such as arm angles, arm velocities, motor currents and the TCP motion calculated from the measured arm angles with the forward kinematics. The TCP motion makes an ellipse movement in the x-, y-plane with same start and end position. The z component makes a small height difference between the start/end position and the middle position of the TCP motion. This can be seen in Figure 11.15.

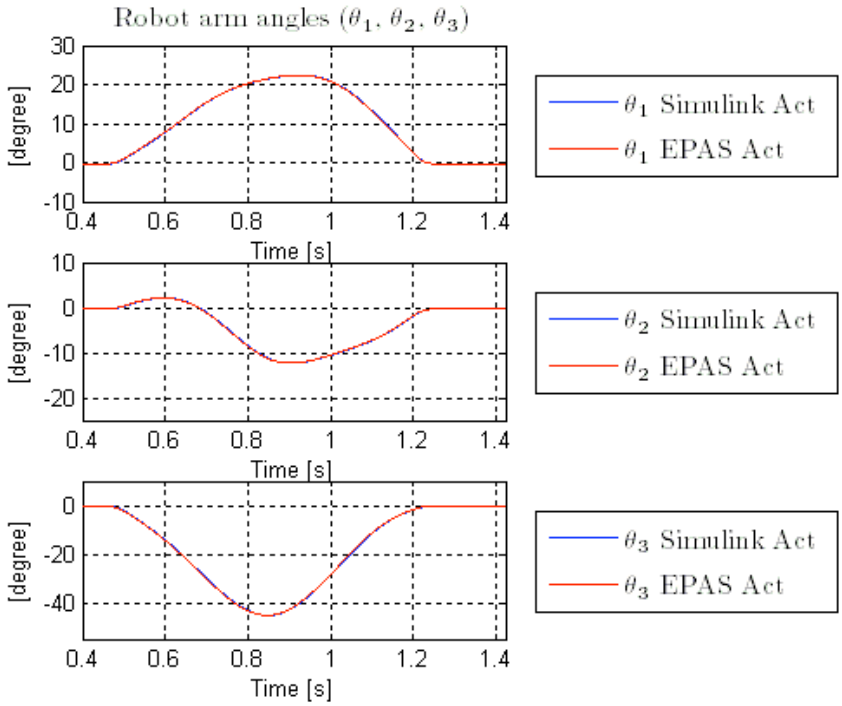


Figure 11.10, shows the motion of the three arm angles for the Delta-3 robot with the TCP motion as in Figure 9.9.

The measured arm angles in the simulink model are compared with the measured arm angles from EPAS for the real robot in Figure 11.10. The units at the y-axis are in degrees and are shown over the time in seconds. The blue line describes the measured actual value from the three upper arms which are connected to the three motors. The red line describes the measured actual value for the real robot sampled in EPAS.

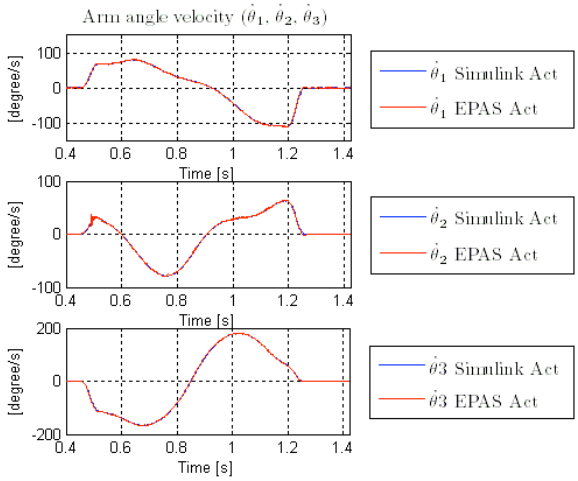


Figure 11.11, shows the velocity for the three arm angles. Compares Simulink model with real robot controlled with EPAS.

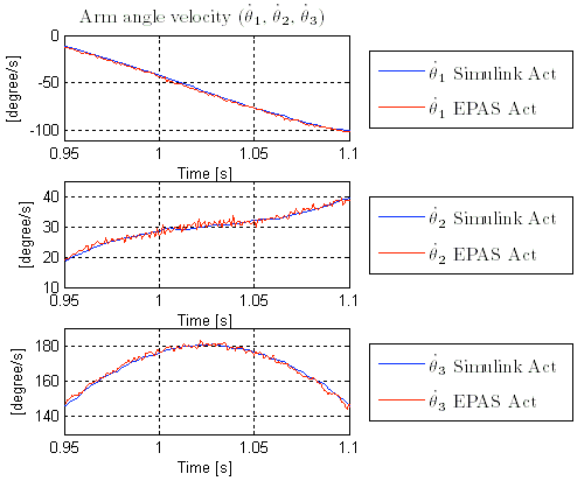


Figure 11.12, shows a zoom of Figure 11.11 of the time interval 0.95s to 1.1s.

The velocity graph above shows the angle velocity in degrees per seconds for the three upper robot arms which are connected to the three motors at the base plate, see Figure 2.2. The blue line describes the measured actual value in the simulink model and the red line the measured actual value from the real robot. One can see in the right plot that the measured velocity from the simulink model and the real robot match well. But as in experiment 1the right plot which shows a zoom from the left plot for the time interval 0.95s to 1.1s, one can see that the simulink model follows the real robots measured value well except from the small amplitude variation which the real robot has.

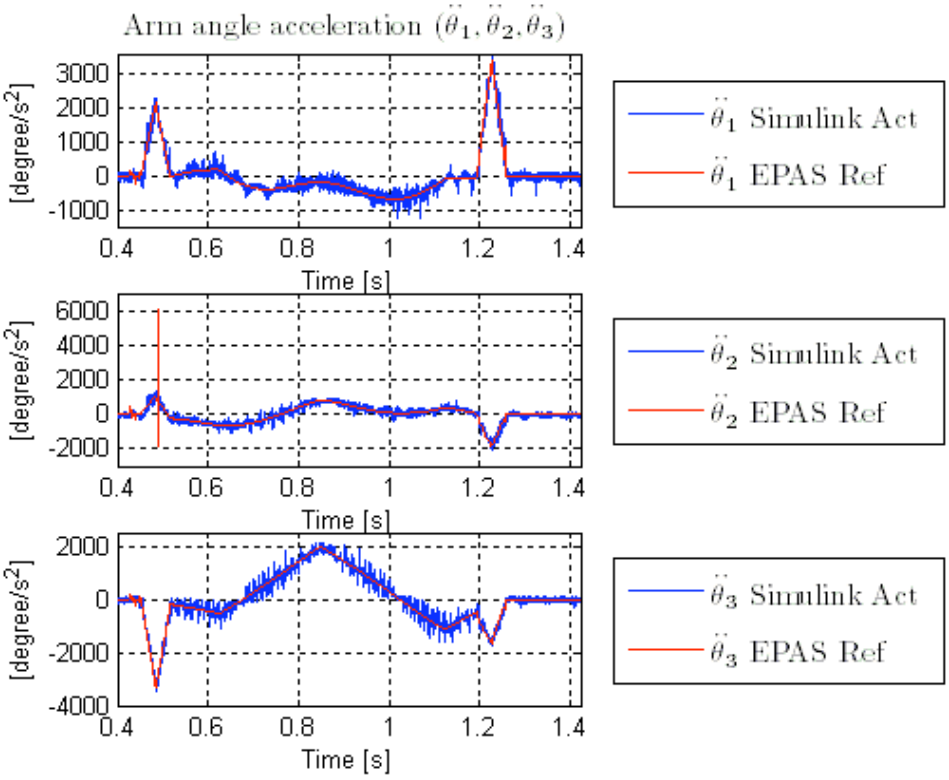


Figure 11.13, shows the acceleration for the three arm angles.

In the acceleration plot above are the three measured upper arm angles accelerations from the simulink model compared to the reference upper arm angles accelerations value in EPAS. This is because only the reference value for the arm acceleration can be measured in EPAS and not the actual value. The plot shows still that the mean value for three upper arm angle accelerations of the Delta-3 robot in the simulink model follows the measured reference acceleration value in EPAS. The arm angle accelerations are shown in degrees per square seconds over the time in second.

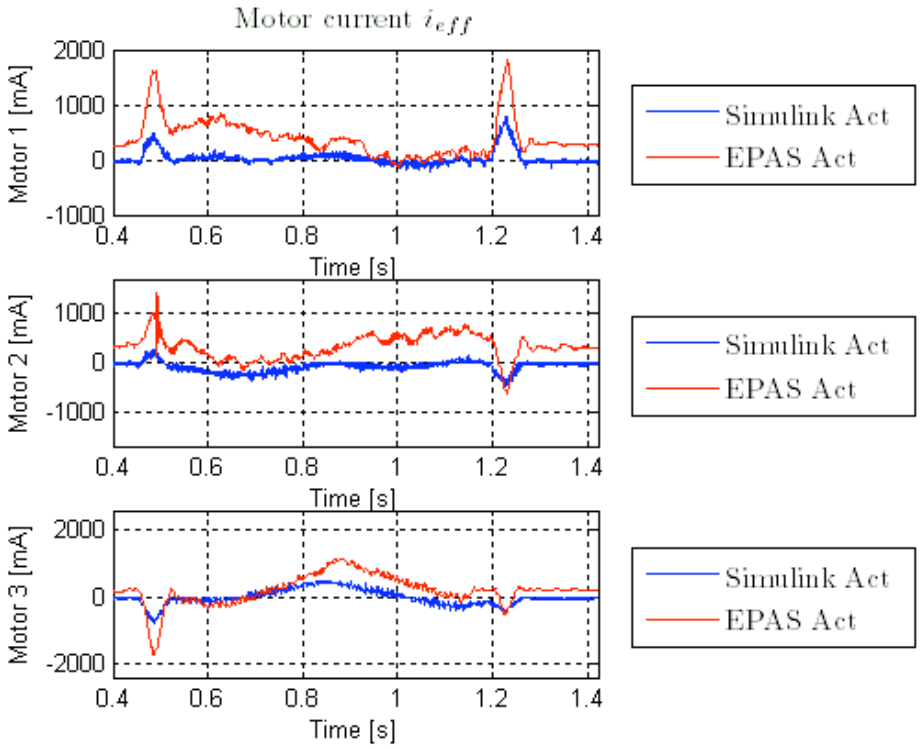


Figure 11.14, shows the current I_{eff} for the three motors.

In Figure 11.14 are the three currents shown which acts as input to each of the three motors. The blue line represents the measured current I_{eff} in the Simulink model, see Figure 8.4 and the red line represents the measured current from each real motor sampled in EPAS.

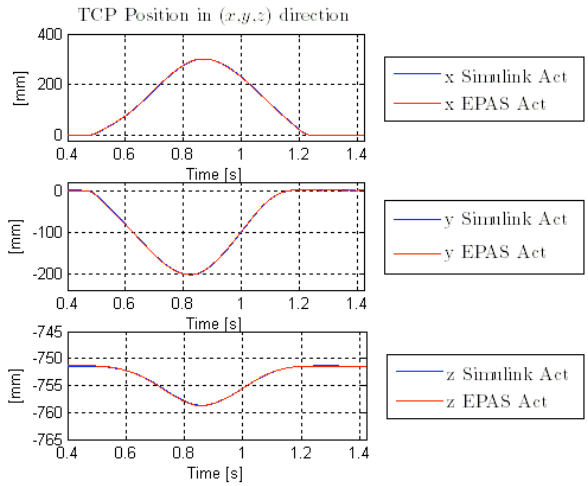


Figure 11.15, shows the TCP motion in x,y and z components of the Delta-3 robots TCP position.

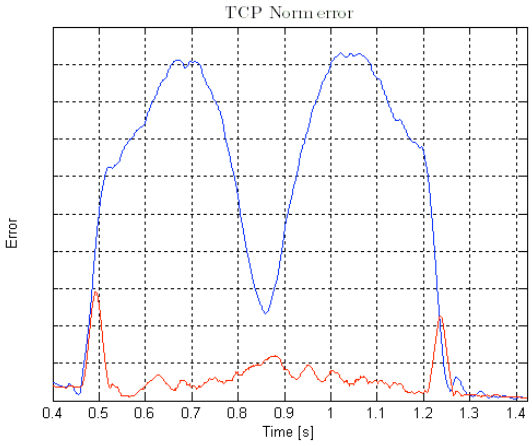


Figure 11.16, shows the norm of the TCP error. The blue line is Simulink models error and red line is TCP error from real robot controlled with EPAS.

Figure 11.15 above shows the x, y and z components of the TCP position. The blue line represents the measured actual value in the simulink model and the red line represents the

measured actual value from the real robot. The units at the y-axis are the lengths in *mm* and the x-axis units are the time in seconds.
Figure 11.16 shows the two norm of the deviation between the measured actual TCP position and the reference TCP position over the time in seconds. The blue line shows the TCP norm error for the simulink model and the red line are for the real robot.
As in experiment 1when one looking at the right plot one can see that the simulink model and the real robots norm of the TCP error don't agree. This is a result of that not all the dynamics for the Delta-3 robot are included in the simulink model.

11.1.3 SIMULINK EXPERIMENT 3

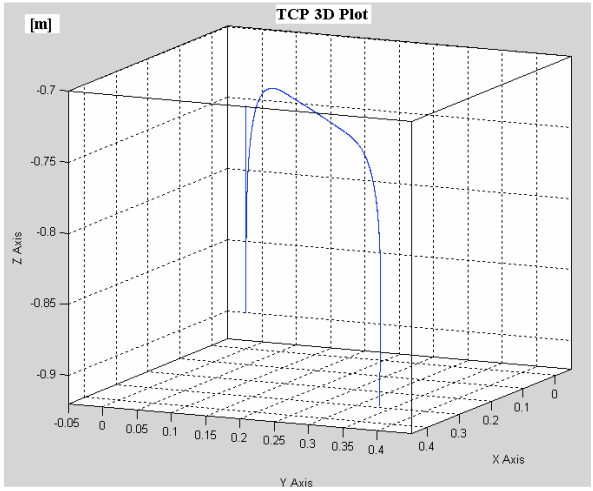


Figure 11.17, describes the motion of the TCP when simulating with the parameters, P1, P2, P3, P4.

The 3D plot above describes the TCP motion used at the real robot and in the simulink model to compare the different measured signals such as arm angles, arm velocities, motor currents and the TCP motion calculated from the measured arm angles with the forward kinematics. The TCP motion start at the point (0,0,-0.9)*m* and makes a pick and place movement from the start point to the point (0.34,0.34,-0.91)*m* and back again to the starting point, to be able to continue the same pick and place movement in another direction. Because the oscilloscope in EPAS has a limited number of sample only one pick and place direction is used in this experiment.

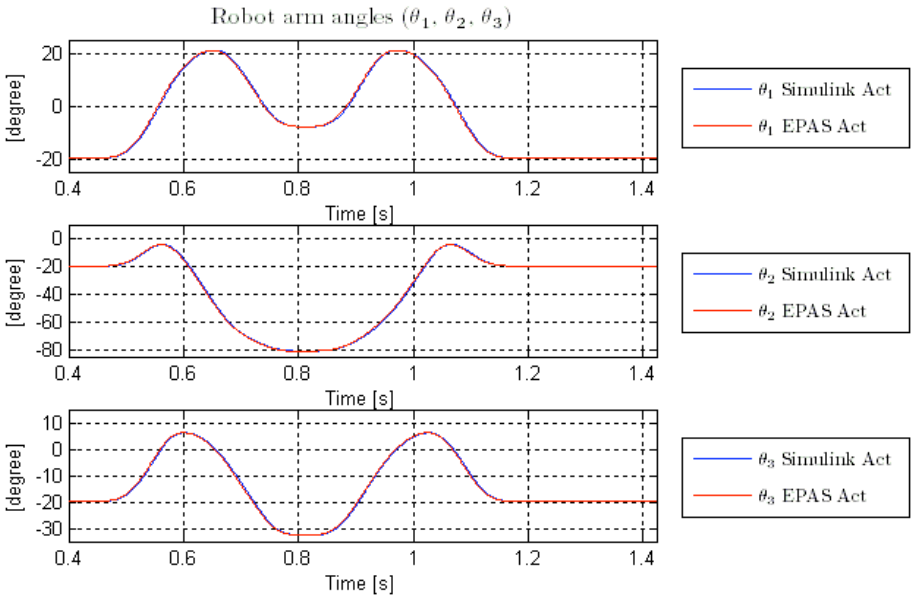


Figure 11.18, shows the motion of the three arm angles for the Delta-3 robot with the TCP motion as in Figure 9.9.

The measured actual arm angles in the simulink model are compared with the measured actual arm angles from EPAS for the real robot in Figure 11.18. The units at the y-axis are in degrees

and are shown over the time in seconds. The blue line describes the measured actual value from the three upper arms which are connected to the three motors. The red line describes the measured actual value for the real robot sampled in EPAS.

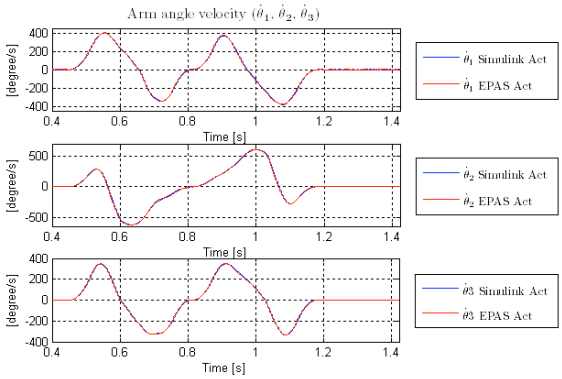


Figure 11.19, shows the velocity for the three arm angles. Compares Simulink model with real robot controlled with EPAS.

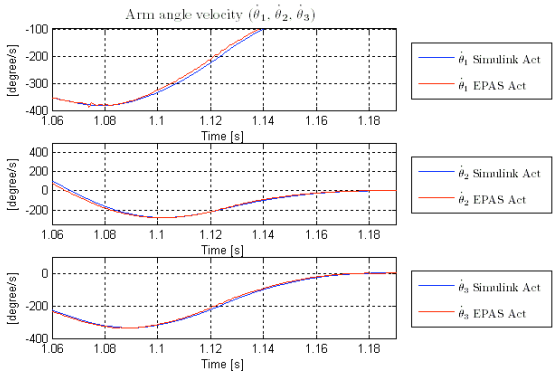


Figure 11.20, shows a zoom of Figure 11.11 of the time interval 0.95s to 1.1s.

The velocity graph above shows the angle velocity in degrees per seconds for the three upper robot arms which are connected to the three motors at the base plate, see Figure 2.2. The blue line describes the measured actual value in the simulink model and the red line the measured actual value from the real robot. One can see in the right plot that the measured velocity from the simulink model and the real robot match well. The right plot shows a zoom of the arm angle velocity for the time interval 1.06s to 1.19s. The EPAS actual signal doesn't have a large amplitude variation so this time the simulink models actual value follows the EPAS actual value better than in experiment 1 and 2.

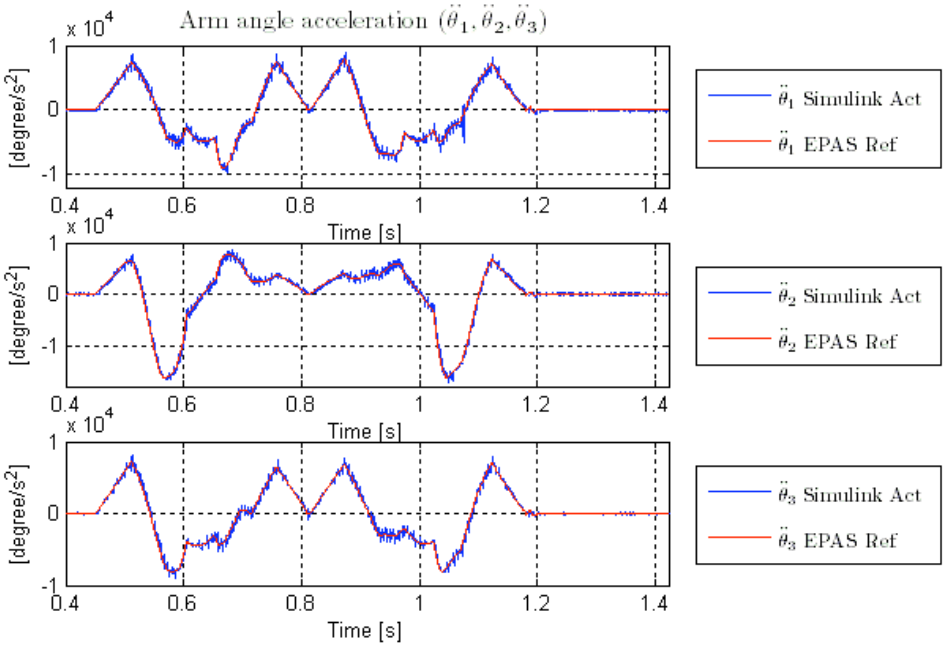


Figure 11.21, shows the acceleration for the three arm angles.

As in the Simulink experiment 1 and 2, Figure 11.21 shows the three measured upper arm actual angles accelerations from the simulink model compared to the reference upper arm angles accelerations value in EPAS. Also for this experiment one can see that the mean value of the three upper arm angle accelerations for the Delta-3 robot in the simulink model follows the measured reference acceleration value in EPAS. The arm angle accelerations are shown in degrees per square seconds over the time in second.

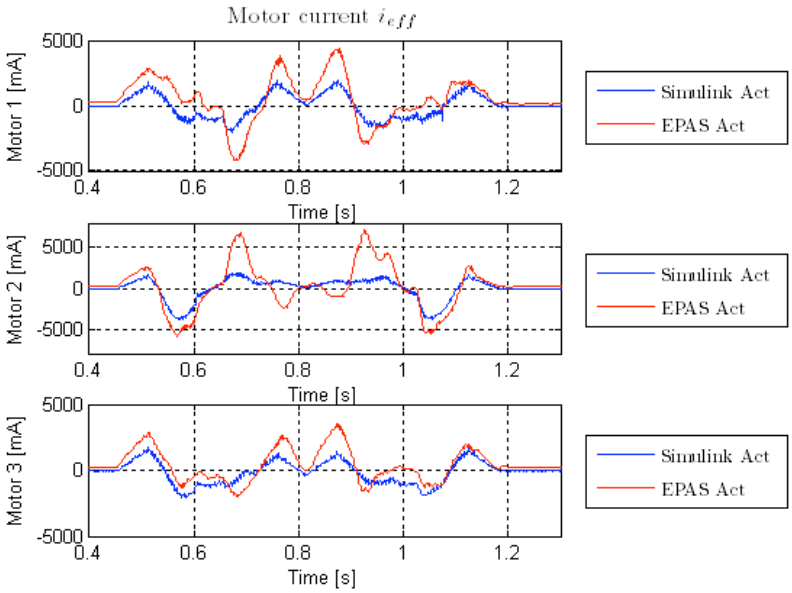


Figure 11.22, shows the current I_{eff} for the three motors.

In Figure 11.22 are the three currents shown which acts as input to each of the three motors. The blue line represents the measured current I_{eff} in the Simulink model, see Figure 8.4 and the red line represents the measured current from each real motor sampled in EPAS.

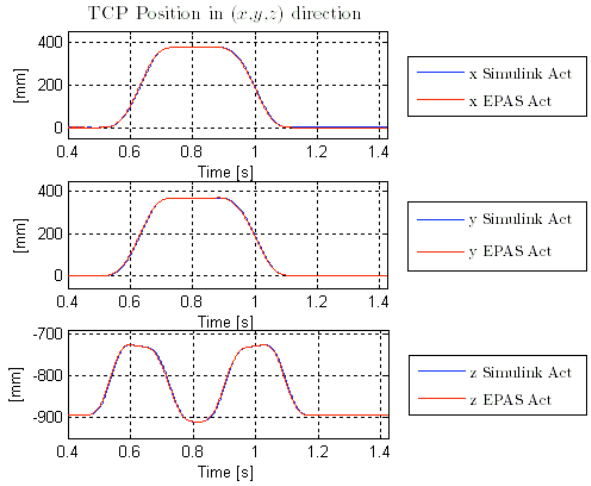


Figure 11.23, shows the TCP motion in x, y and z components of the Delta-3 robots TCP position.

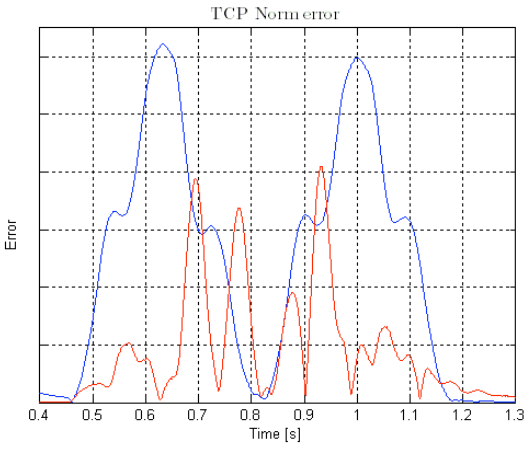


Figure 11.24, shows the norm of the TCP error. The blue line is Simulink models error and red line is TCP error from real robot controlled with EPAS.

Figure 11.23 shows the x, y and z components of the TCP position. The blue line represents the measured actual value in the simulink model and the red line represents the measured actual value from the real robot. The y-axis units are the lengths in *mm* and the x-axis units are the time in seconds.
Figure 11.24 shows the two norm of the deviation between the measured actual TCP position and the reference TCP position over the time in seconds. The blue line shows the TCP norm error for the simulink model and the red line are for the real robot.

11.1.4 DISCUSSION

In experiment 1-3 the trajectory graphs for the TCP position and the arm angles positions shows that the model in simulink follows the desired trajectories well. But when comparing the norm of the TCP tracking error for the Simulink model and the real robot, one can see that there is a large deviation, see Figure 11.8, Figure 11.16 and Figure 11.24. This deviation is caused because there are dynamics missing for the robot model in Simulink. This can also be seen in the graphs for the current where the Simulink models actual value has a large deviation from the actual value measured from the real robot, see Figure 11.6, Figure 11.14 and Figure 11.22.

11.2 JACOBIAN MATRIX RESULTS IN EPAS

This section shows the results of the Jacobian implementation in EPAS for the same TCP trajectory used in experiment 1, 2 and 3, see section 11.1.1 to 11.1.3. First are the arm angle trajectory and the respectively TCP trajectory which are used in the experiment shown. Then are the joint velocities, $(\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)$ shown together with the percentage deviation between the calculated and the reference velocity. At last the calculated joint accelerations, $(\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3)$ compared to its reference value in EPAS shown.
The joint velocity is calculated with known x, y and z components of the TCP position, the three joint positions, $(\theta_1, \theta_2, \theta_3)$ and the TCP velocity, $(\dot{x}, \dot{y}, \dot{z})$. The joint acceleration is calculated with known x, y and z components of the TCP position, the three joint positions, $(\theta_1, \theta_2, \theta_3)$ and the TCP acceleration, $(\ddot{x}, \ddot{y}, \ddot{z})$.

11.2.1 JACOBIAN EXPERIMENT 1

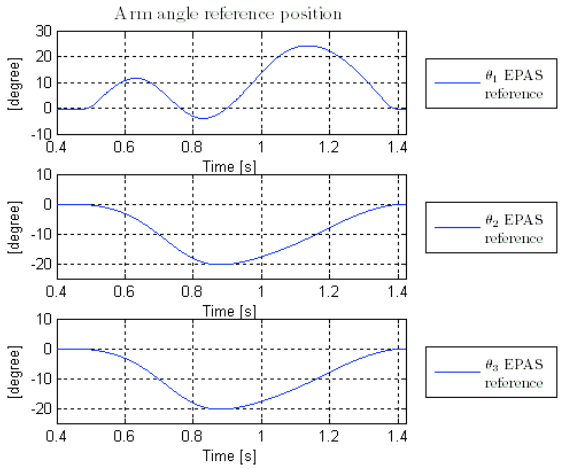


Figure 11.25, shows the three arm angles used to calculate velocity and acceleration with the Jacobian.

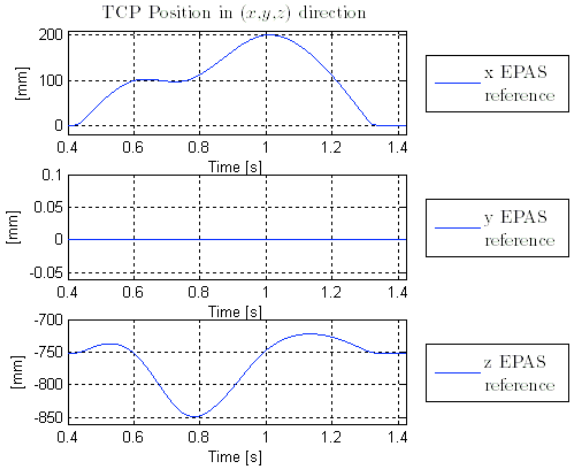


Figure 11.26, shows the TCP position used to calculate the velocity and acceleration with the Jacobian matrix.

The Figure 11.25 and Figure 11.26 above shows the three arm angle trajectories used as reference in the experiment and the TCP trajectory for the directions, (x,y,z). The left plot shows the three arm angle positions in degrees over the time in seconds. The right plot shows the x, y and z components of the position of the TCP in the world coordinate frame, see Figure 4.1.

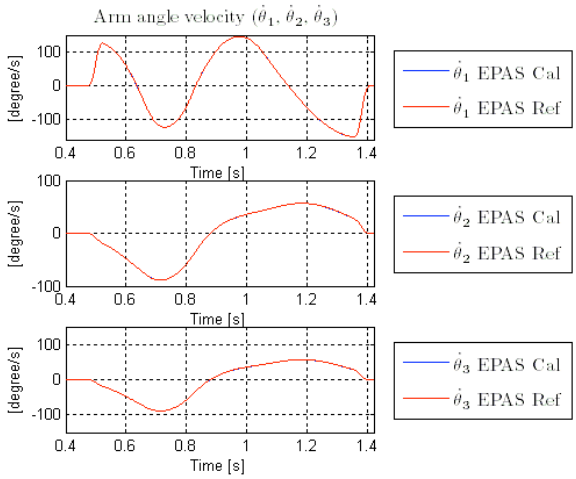


Figure 11.27, the blue line shows the velocity calculated with the Jacobian matrix, and the red line shows the reference velocity from EPAS.

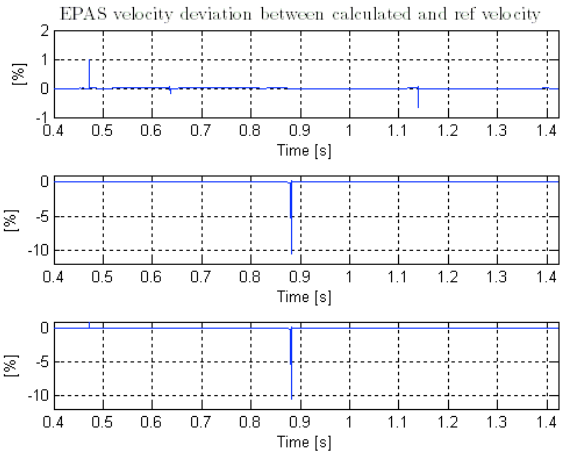


Figure 11.28, shows deviation between calculated and reference velocity. Where the calculated is computed with the Jacobian.

Figure 11.27 shows the three different arm joint velocities in the unit degree/s over the time in seconds. To make it easier to see how big the deviation is between the EPAS reference value of the velocity and with Jacobian matrix calculated velocity Figure 11.28 shows the percentage deviation between EPAS reference value and with the Jacobian calculated value.

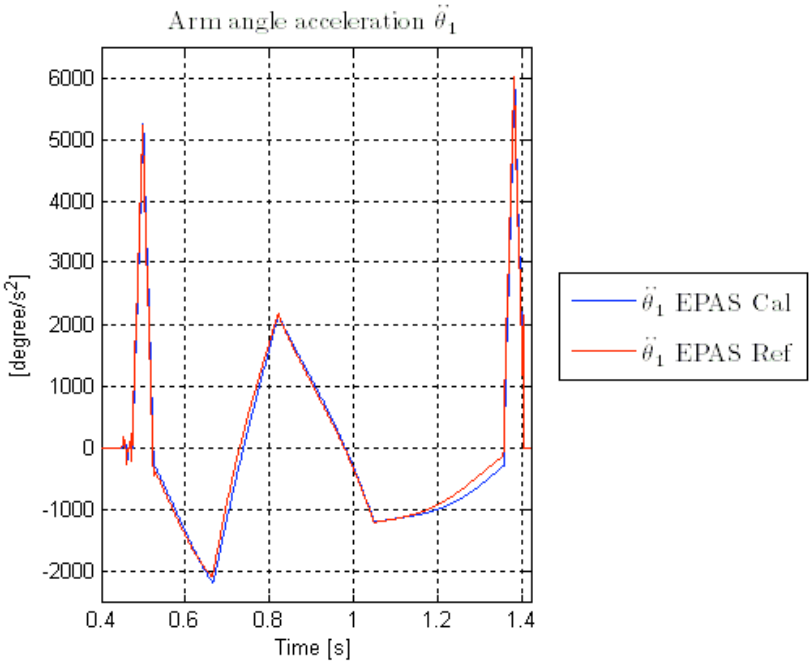


Figure 11.29, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.29 the blue line represents the calculated joint acceleration for arm 1 of the Delta-3 robot and the red line describes the reference acceleration in EPAS. If one looks at the joint velocity of joint 1 there is a smaller percentage deviation (Figure 11.28) between the calculated

and the reference value than for joint 2 and 3. This is also shown in the acceleration graph for joint 2 where the calculated value match better with the reference value than for joint 2 and 3.

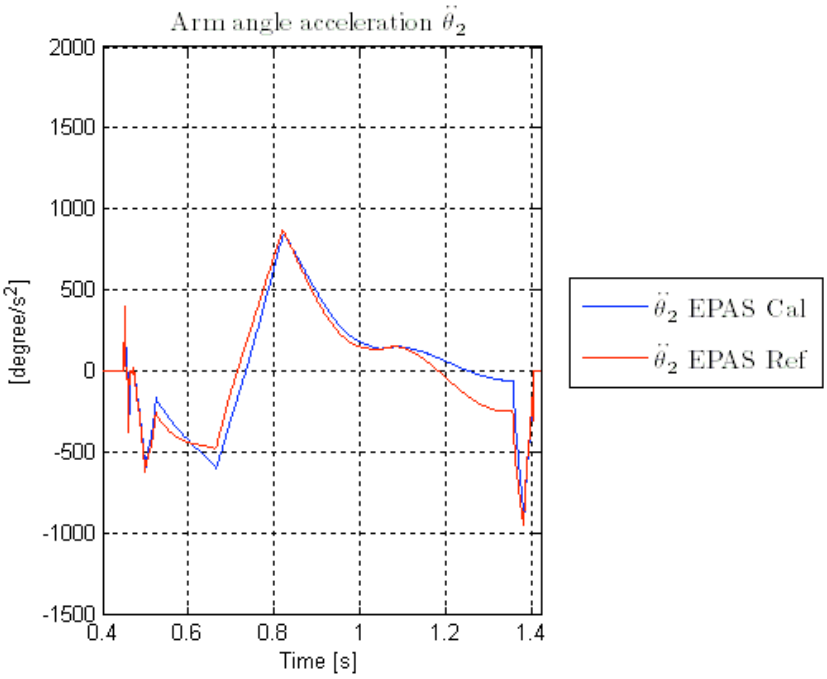


Figure 11.30, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.30 the blue line represents the calculated joint acceleration for arm 2 of the Delta-3 robot and the red line describes the reference acceleration in EPAS.

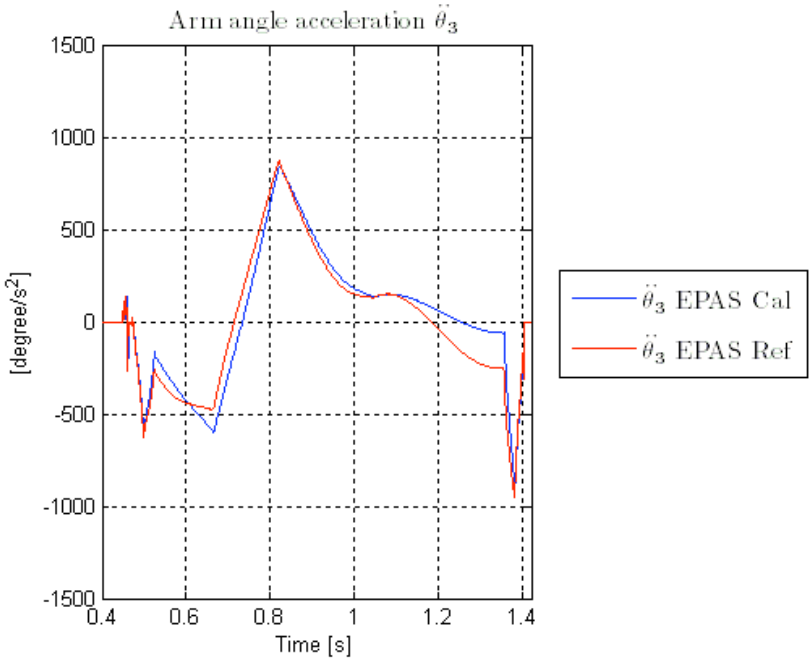


Figure 11.31, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

The joint acceleration for the third joint at the Delta-3 robot is shown in Figure 11.31. The blue line represents the calculated joint acceleration with the Jacobian matrix compared with the red line which is the reference acceleration from EPAS. One can see that joint 2 and 3 has the same motion in Figure 11.25 and this is also seen in the arm acceleration graphs for joint 2 and 3 where the joint acceleration is the same.

11.2.2 JACOBIAN EXPERIMENT 2

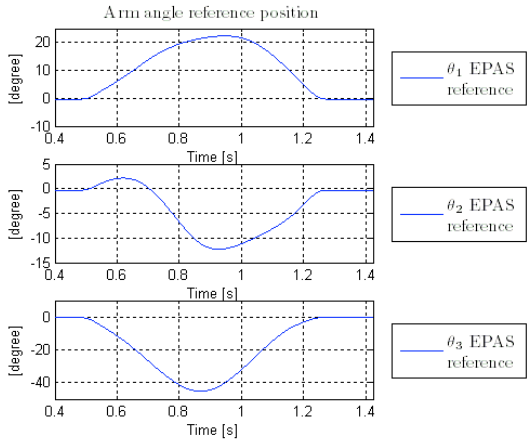


Figure 11.32, shows the three arm angles used to calculate velocity and acceleration with the Jacobian.

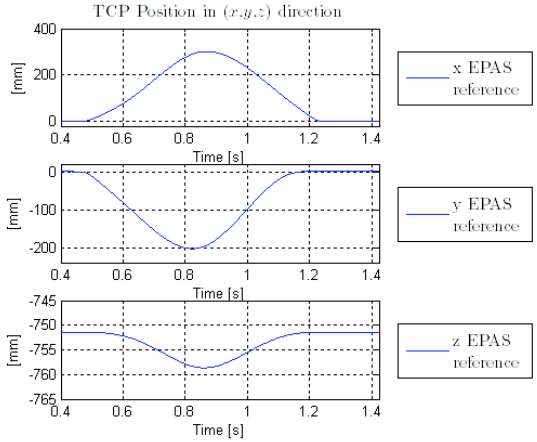


Figure 11.33, shows the TCP position used to calculate the velocity and acceleration with the Jacobian matrix.

Figure 11.32 and Figure 11.33 above shows the three arm angle trajectories used as reference in the experiment and the TCP trajectory for the three components, (x,y,z). The left plot (Figure 11.32) shows the three arm angle positions in degrees over the time in seconds. The right plot (Figure 11.33) shows the x, y and z components of the position of the TCP in the world coordinate frame, see Figure 4.1.

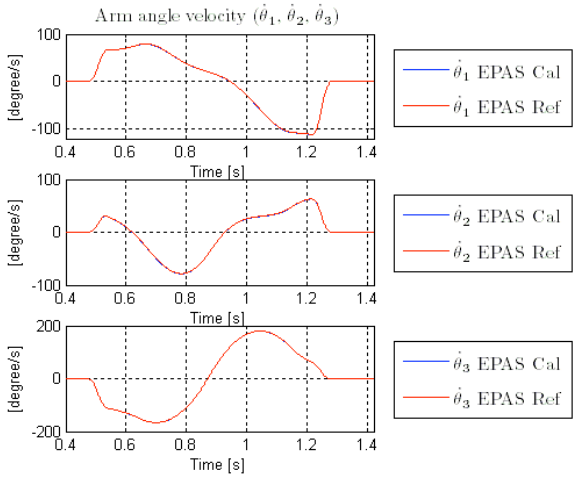


Figure 11.34, the blue line shows the velocity calculated with the Jacobian matrix, and the red line shows the reference velocity from EPAS.

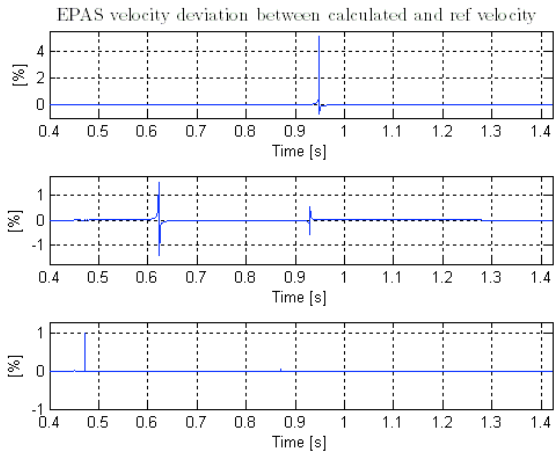


Figure 11.35, shows deviation between calculated and reference velocity. Where the calculated is computed with the Jacobian.

Figure 11.34 shows the three different arm joint velocities in the unit degree/s over the time in seconds. To make it easier to see how big the deviation is between the EPAS reference value of the velocity and with Jacobian matrix calculated velocity Figure 11.35 shows the percentage deviation between EPAS reference value and with the Jacobian calculated value.

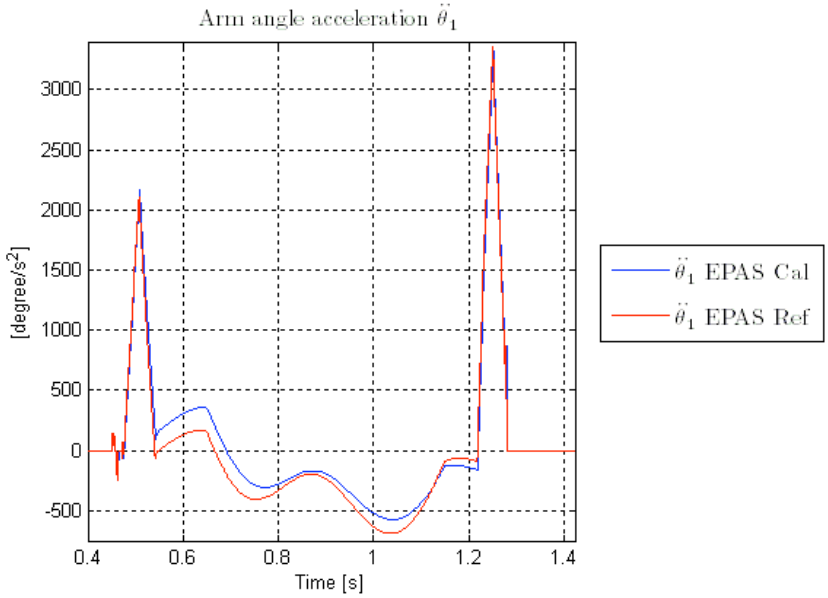


Figure 11.36, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.36 the blue line represents the calculated joint acceleration for arm 1 of the Delta-3 robot and the red line describes the reference acceleration in EPAS. Here one can see that the deviation is larger between the calculated acceleration and the reference acceleration for joint 1 before and after where there is a larger percentage deviation of the calculated velocity.

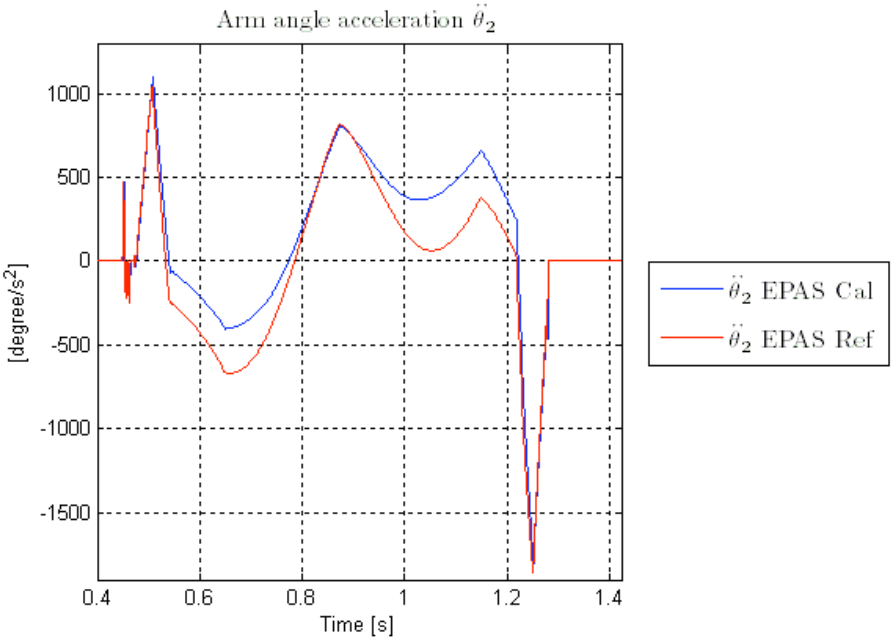


Figure 11.37, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.37 the blue line represents the calculated joint acceleration for arm 2 of the Delta-3 robot and the red line describes the reference acceleration in EPAS. If one look at the joint velocity of joint 2 there is a smaller percentage deviation (Figure 11.35) between the calculated and the reference value than for joint 1. This is also shown in the acceleration graph for joint 2 where the calculated value match better with the reference value than for joint 1.

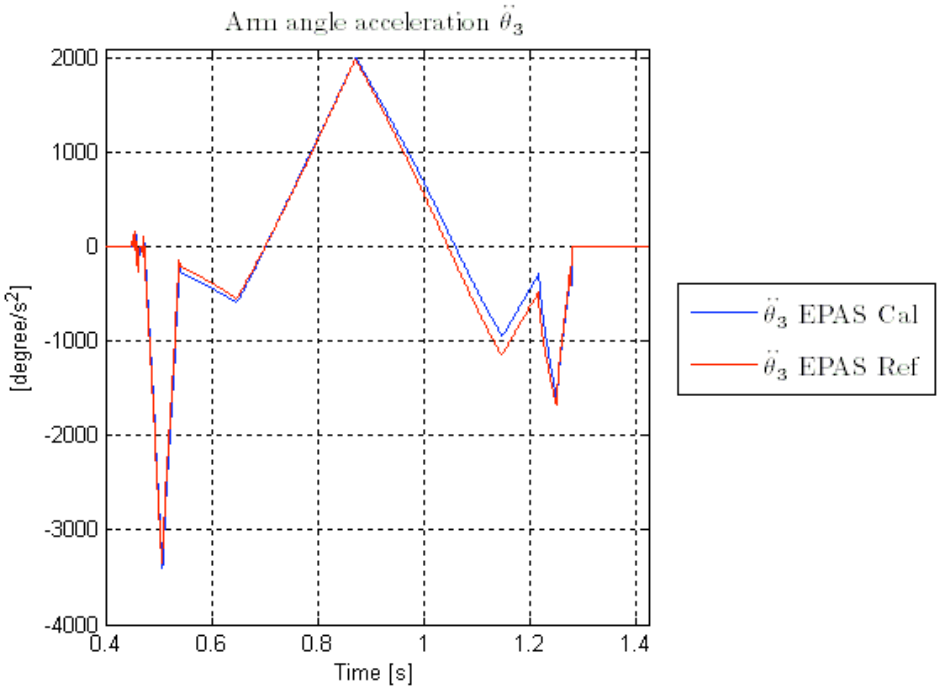


Figure 11.38, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

The joint acceleration for the third joint at the Delta-3 robot is shown in Figure 11.38. The blue line represents the calculated joint acceleration with the Jacobian matrix compared with the red line which is the reference acceleration from EPAS. The percentage deviation of the joint 3 velocity is small which also can be seen in the graph above where the deviation between the calculated and the reference acceleration is smaller.

11.2.3 JACOBIAN EXPERIMENT 3

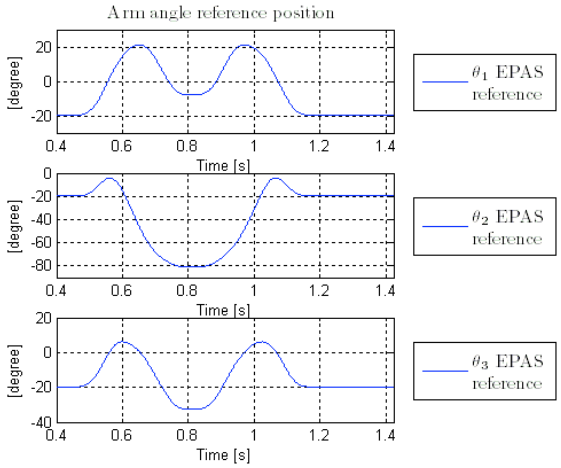


Figure 11.39, shows the three arm angles used to calculate velocity and acceleration with the Jacobian.

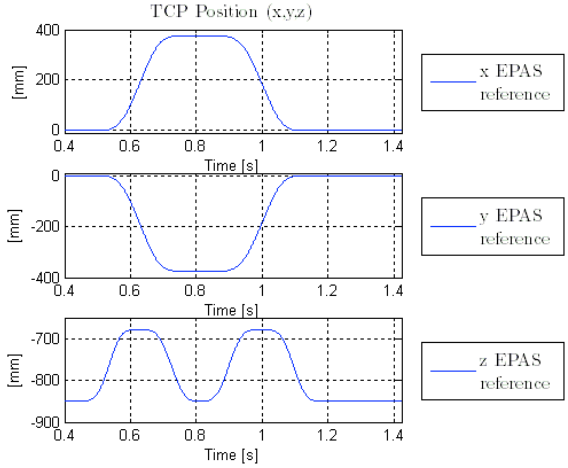


Figure 11.40, shows the TCP position used to calculate the velocity and acceleration with the Jacobian matrix.

The plots above shows the three arm angle trajectories used as reference in the experiment and the TCP trajectory for the directions, (x,y,z). The left plot shows the three arm angle positions in degrees over the time in mille seconds. The right plot shows the x,y and z components of the position of the TCP in the world coordinate frame, see Figure 4.1.

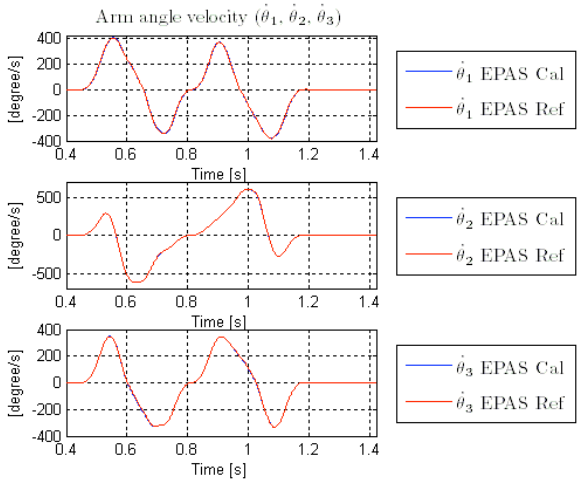


Figure 11.41, the blue line shows the velocity calculated with the Jacobian matrix, and the red line shows the reference velocity from EPAS.

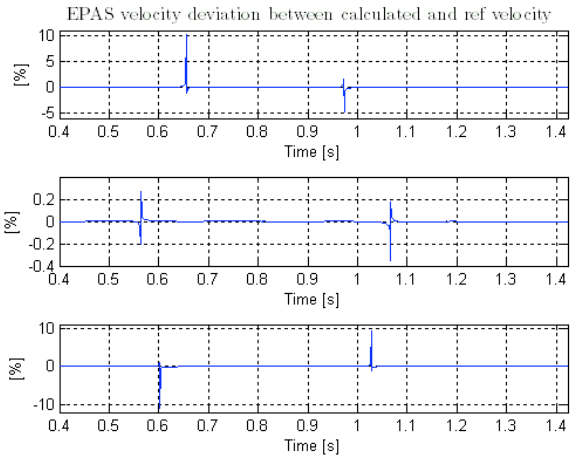


Figure 11.42, shows deviation between calculated and reference velocity. Where the calculated is computed with the Jacobian.

Figure 11.41 shows the three different arm joint velocities in the unit degree/s over the time in mille seconds. To make it easier to see how big the deviation is between the EPAS reference value of the velocity and with Jacobian matrix calculated velocity Figure 11.42 shows the percentage deviation between EPAS reference value and with the Jacobian calculated value.

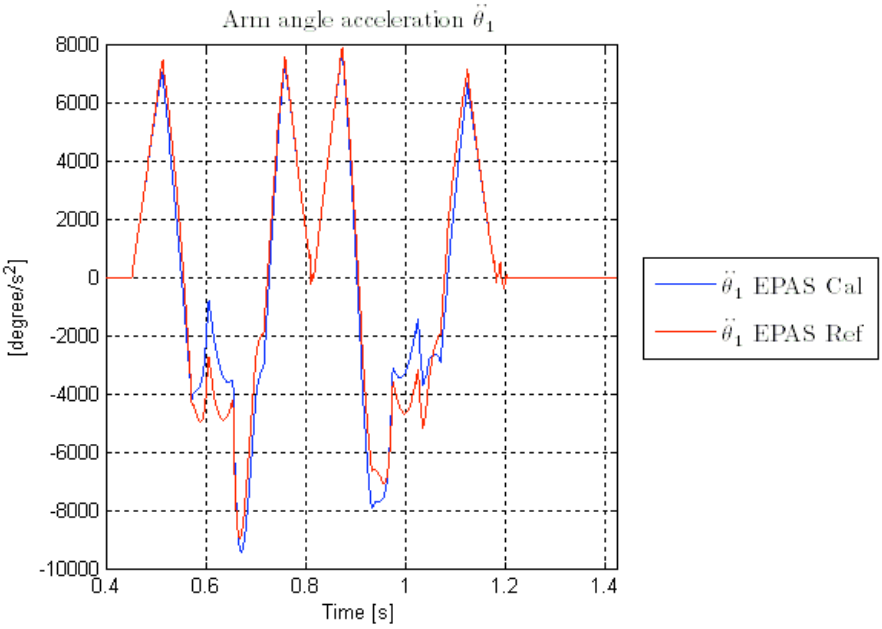


Figure 11.43, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.43 the blue line represents the calculated joint acceleration for arm 1 of the Delta-3 robot and the red line describes the reference acceleration in EPAS. One can see that where the deviation is larger between the calculated acceleration and the reference acceleration of joint 1 there is also a larger percentage deviation of the calculated velocity.

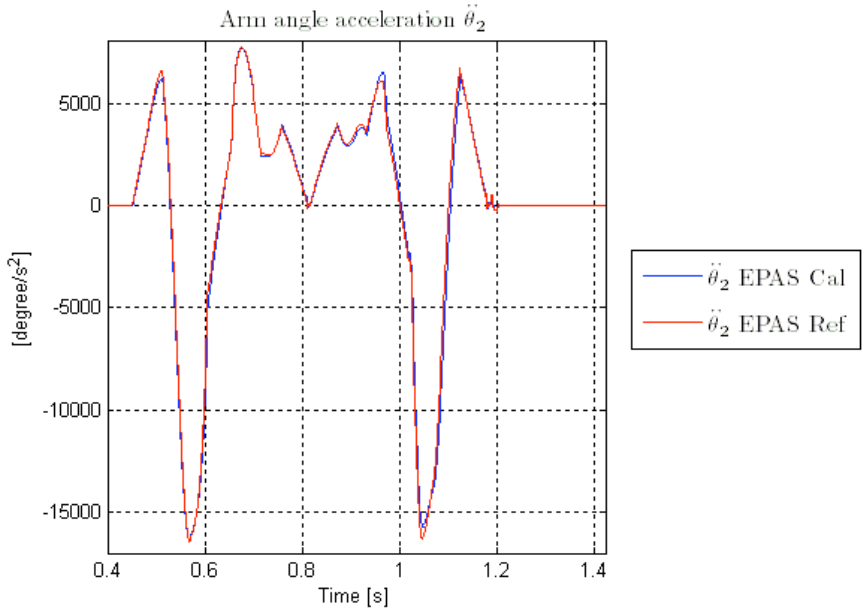


Figure 11.44, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

In Figure 11.44 the blue line represents the calculated joint acceleration for arm 2 of the Delta-3 robot and the red line describes the reference acceleration in EPAS. If one look at the joint velocity of joint 2 there is a smaller percentage deviation (Figure 11.42) between the calculated and the reference value than for joint 1. This is also shown in the acceleration graph for joint 2 where the calculated value match better with the reference value than for joint 1.

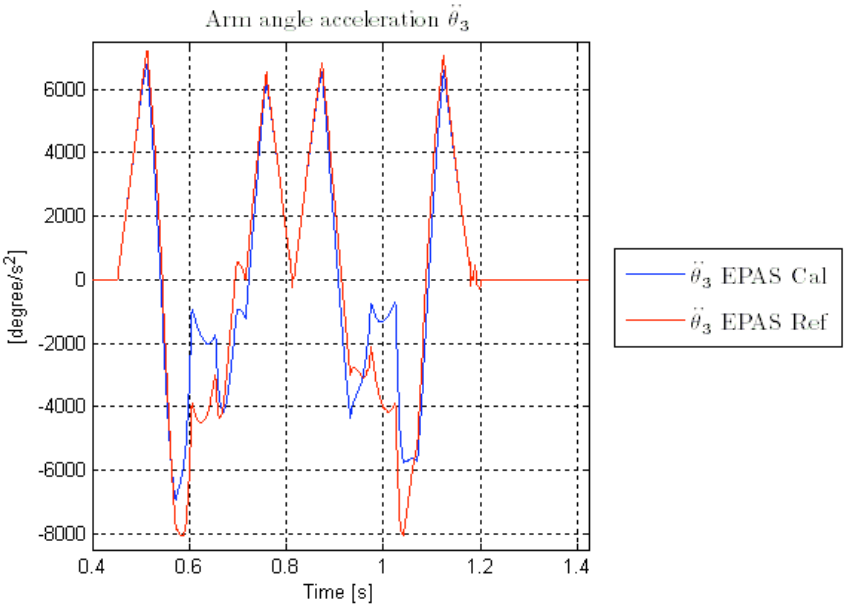


Figure 11.45, the blue line shows the calculated acceleration and the red line shows the reference acceleration from EPAS.

The joint acceleration for the third joint at the Delta-3 robot is shown in Figure 11.45. The blue line represents the calculated joint acceleration with the Jacobian matrix compared with the red line which is the reference acceleration from EPAS. As for joint 1 also the percentage velocity deviation for joint 3 is larger at the time interval where the acceleration deviation is larger.

11.2.4 DISCUSSION

When using the Jacobian matrix in EPAS to calculate the TCP velocity respectively the angular velocity for the upper arms one can see a small deviation between the measured value and the calculated value in all three experiments above. The deviation for experiment 1 and 2 is as most 10% and for experiment 2 is the largest deviation around 5%. In general is the arm angle velocity deviation smaller in experiment 2 than for experiment 1 and 3. One reason for this is that the angle velocity of the three arms is smaller than for the two other experiments.
If one considers Figure 11.42 (shows the angle velocity deviation in percentage) when looking at the joint acceleration plots for the three arms one can see that where the joint acceleration has a big deviation also the velocity has a larger deviation. Figure 11.44 shows when the calculated velocity has a small deviation also the calculated acceleration will have a very small deviation.
A factor which affects the calculated values is the kinematic parameters used in the Jacobian matrix. The kinematic parameters are given from the manufacture of the robot arms and they

also have a small deviation from reality which affects the calculated values. This could be one factor why the calculated value isn't exact with the reference value from EPAS. Another factor which also affects the results is the singularity of the Jacobian. When the Jacobian becomes more singular the calculated value will also deviate more. These singularities arise when the TCP is near the working surface limits.

12 CONCLUSIONS

In section 11.1 one can see that the Arm angle positions and the TCP positions are following the measured value from the real robot quit well. But when one calculates the norm of the TCP error and compare this between the Simulink model and the real robot it shows that there is a deviation. This deviation depends on the model in Simulink is missing some dynamics of the real robot.
One solution to become more dynamics in the Simulink model is to estimate the parameters in P4 which contains parameters for the motors used to actuate the robot arms. This method have been tested with the function *lsqnonlin* in Matlab. The parameters which have been estimated are the *R, L Fv, Fs* and I_m. The result from these estimations is that Matlab gives back the same initial value used for the different parameters. A reason for this is that the minimization function which Matlab tries to minimize already is in a local minimum.

When using the Jacobian matrix in EPAS to calculate the velocity of the TCP and the arm angles, respectively one can see a really god estimation. But for the acceleration calculation the result is not so close to the measured value. One cause could be the estimation of the Delta-3 robots kinematics. If the accuracy of the kinematics parameters estimation isn't good enough this will also contribute to a deviation when using the Jacobian matrix for velocity or acceleration calculations. Another cause of the accuracy loss is the simplification of the kinematic model which have been made in section 4.1 for the Delta-3 robot. The model assumptions include that the travelling plate is replaced by a point *P*, which the three forearms are connected to (TCP) and that the forearms are substituted with simple rods, affects the models accuracy according to the real robot when calculating the velocity or acceleration in joint space or Cartesian space.

13 REFERENCES

1. **Murray, Richard M., Zexiang, Li and Sastry, Shankar S.** *A Mathematical Introduction to Robotic Manipulation.* Florida : CRC Press Inc., 1994.

2. **Craig, J. John.** *Introduction to Robotics Mechanics & Control* : Addison-Wesley Publishing Company Inc, 1986.

3. **ELAU, GmbH.** Robotics. *Embedded robotic functionality for packaging machinery.* Marktheidenfeld, Germany : ELAU GmbH, 2008.

4. **ELAU, GmbH**. PacDrive Automation Toolkit EPAS-4. *SERCOS Interface.* Marktheidenfeld, Germany : ELAU GmbH, 2007.

5. **Spong, Mark W. and Vidyasagar, M.** *Robot Dynamics and Control.* : John Wiley & Sons Inc, 1989.

6. **Ecorchard, G. and Maurine, P.** Self-Calibration of Delta Parallel Robots with Elastic Deformation Compensation. *Intelligent Robots and Systems.* 2005, pp. 462-467.

7. **Codourey, Alain.** Dynamic Modeling of Parallel Robots for Computed-Torque Control Implementation. *The International Journal of Robotics Research.* 1998, Vol. 17, 12, pp. 1325-1336.

8. **Wang, J and Liu, X -J.** Analysis of a novel cylindrical 3-DoF parallel robot. *Robotics and Autonomous Systems.* 2003, Vol. 42, 1, pp. 31-46.

9. **Sciavicco, L. and Siciliano, B.** *Modelling and Control of Robot Manipulators.* London : Springer-Verlag London Ltd, 2000.

10. **Codourey, A.** Dynamic modelling and mass matrix evaluation of the DELTA parallel robot for axes decoupling control. *Intelligent Robots and systems '96.* 1996, Vol. 3, pp. 1211-1218.

11. **Gross, Hauger, Schröder, Wall.** *Technische Mechanik 3.* Berlin : Springer-Verlag, 2008. ISBN:978-3-540-68422-0.

12. **ELAU, GmbH.** PacDrive Automation Toolkit EPAS-4. *IEC Library Robotic - General.* Marktheidenfeld, Germany : ELAU GmbH, 2007.

13. **ELAU GmbH.** PacDrive AutomationToolkit EPAS-4. *IEC Library Robotics US - Rapid9.* Marktheidenfeld, Germany : ELAU GmbH, 2007.

14. **Sauer, Timothy.** *Numerical Analysis.* USA : Pearson Addison Wesley, 2006.

15. **ELAU, GmbH.** Operating Manual. *PacDrive SH Motor.* Marktheidenfeld, Germany : ELAU GmbH, 2006.

16. **Lundquist, Christian and Rothstrand, Mats.** *Modelling and Parameter Identification of the Mechanism and the Synchronous Motor of the Active Steering.* Göteborg : Chalmers, 2003.