THESIS REPORT

ON

Verification of Crane Control Algorithm

For,

DASA CONTROL SYSTEMS AB

ROTTNE INDUSTRIES AB

BY

Avinash (870208 – 8355)

Shravan Kumar .A (840423 – 9694)

Blekinge Institute of Technology
School of Engineering
Department of Electrical Engineering
Supervisor: Anders Hultgren
Examiner: Sven Johansson

Abstract :

In the present world the heavy machine applications have grown more sophisticated. Crane is designed in such a way to perform the tasks that are hard for human beings, in order to make the work faster and easier. In this area of crane utilization, forwarder movement is the key in performing most of the tasks. Forwarder movement plays an important role in tasks such as collecting logs from the ground and moving them from one place to another. These types of cranes find a wide range of utilization in forestry applications.

In order to perform these tasks movement of the forwarder has to be according. Depending on the loads to carry the size of the forwarder also varies. Partial automation of the crane movement in the form of forwarder movement is the area of interest. Aim of the thesis is to concentrate on automation of crane link, to be more specific, the prismatic link.

This can be achieved through the verification of the mechanism and similarity of the two algorithms in the Matlab system and in the DASA system.

Symbols

θ_1	The Swivel arm angle
θ_2	The Lift arm angle
θ_3	The Elbow arm angle
d_4	Length of prismatic joint
$\dot{\theta}_1, \dot{\theta}_2$	Angular velocities of both the links
$\dot{d}_4$	Joint velocity of the extension link
$J(q)$	Jocobian function
v	Velocity function
o_4	The crane tip position
r, z	Cartesian coordinates in r and z coordinates
$\dot{d}_4^d$	The desired prismatic joint velocity
$\dot{\theta}_{2m}, \dot{\theta}_{3m}, \dot{d}_{4m}$	Maximum joint velocities of the joints
k	Gain parameter which is adjustable
$Lambda(d_4, \dot{d}_4)$	Scalar weighting function
d_{4c}	The Center position of the prismatic joint
d_{4e}	The End position of the prismatic joint
T	The Transformation matrix
x_0, z_0	The crane coordinate planes

Table of Contents

1. INTRODUCTION..... 7

1.1 Forwarder Introduction..... 7

1.2 Area of research..... 7

1.3 Current ongoing projects on Cranes 8

1.4 Problem definition and Goals 8

1.5 Thesis Layout..... 9

2 BACKGROUND..... 11

2.1 Code Composer Studio..... 11

2.2 Automatic Conversion Tools (MCS)..... 11

2.3 Kinematics of the crane..... 12

2.3.1 Crane Geometry..... 12

2.4 D-H Table Parameters..... 13

2.5 Crane Coordinates..... 14

2.6 Forwarder Kinematics..... 14

2.7 AUTOMATIC PRISMATIC JOINT ALGORITHM IMPLEMENTATION..... 16

2.7.1 The Jacobian Matrix..... 16

2.8 DASA LAB TESTING ENVIRONMENT..... 17

2.9 THE CRANE LAB SET UP ENVIRONMENT..... 17

2.9.1 Lab Crane set up with dspace equipment..... 17

2.9.2 Replacement of dSpace system with dasa5 Equipment..... 19

3 How the Errors are overcome by using Interpolation.....21

3.1 Exporting values to Graph 21

3.2 INTERPOLATION..... 22

4 Comparison of Theoretical and practical values..... 25

4.1 Crane Trajectory & Extension Boom movement..... 25

4.2 Comparsions..... 27

5 CONCLUSION..... 29

6 REFERENCES..... 31

7 UML DIAGRAM..... 33

7.1 Flow Diagram..... 33

7.2 Step by Step implementation of Project..... 34

8 Algorithm 35


BLEKINGE TEKNISKA HÖGSKOLA
BTH

1. Introduction

1.1 Forwarder Introduction

Making machines to work for human being is one of the most interesting and continuously progressing fields. It can be any kind of machines that can be used for replacing the manpower. For example, Automation in manufacturing will reduce the number of persons required for the task completion. In order to make such machines for automation we need robotics and control systems. These days' robotics and control systems in combination with computer systems are being developed to make huge advances in various fields. One of such important areas is automation of functionality of heavy vehicles that perform tasks requiring strong power. Automation is of great interest as it will reduce the work pressure on the operator while operating the vehicle. The present work is part of a research project aiming at improving the working environment for operators controlling cranes situated on forwarders. A forwarder is used for loading, transporting, and unloading logs in forest. Forwarders are widely used in forestry. Similar cranes are used in infrastructure industry for lifting heavy materials from ground level to higher levels. As of now, normally the forwarders are being operated by an operator which exerts pressure on him. Hence, the interest for improving operator's crane control interface has been a growing interest.

In a research project run by Linnaeus University, Blekinge Institute of Technology, Rottne industry AB and Dasa Control AB improvement of the forwarder operators working environment is addressed. In this project different control algorithms have been tested in a scaled laboratory crane controlled by the software Matlab/Simulink and the hardware dSpace. Matlab and dSpace supports with a convenient working condition with rapid prototyping properties. The laboratory crane can also be controlled by electronic devices produced by Dasa Control AB used in the forwarders. In Dasa the software is developed in Visual C++ environment.

After implementing the algorithms in the Dasa control system the result has to be verified with the algorithm developed and implemented in the Matlab and dSpace system. This needed in order to prepare for experiments with the developed algorithm in a forwarder working in the forest loading and unloading timber. This report describes a thesis work aiming at verifying a developed crane control algorithms in the Dasa control system.

1.2 Area of thesis project

The work done in this thesis can be divided into three parts. Collecting data from the working Dasa system, collecting data from the Matlab simulation, and comparing the data from those two systems.

Primarily, we need to extract the values for a specified set of variables. That is being provided by a special software module in the DASA system, when the crane is run under practical environment in the crane laboratory.

Secondarily, we need to develop a code in Matlab simulating the crane and the crane control algorithm.

Thirdly the data set from the Dasa system and from the Matlab simulation should be compared in order to verify that the Dasa algorithm works in the same way as the Matlab algorithm when running the crane in simulation by using the values which have been provided by the Dasa system. Once the crane code is executed, we need to analyze the output graph and investigate whether all the forwarder links are run as per the requirement.

The special software module in the Dasa system that is used for collecting the data can collect data with short enough sampling interval only for a limited period of time. After this period of time collected data is stored in the hard disk. During this process the data acquisition system is interrupted and no data is collected for a particular period of time. In order to get data for periods long enough covering a full working cycle some data is needed to be generated by using interpolation technique.

In the second stage, we need to select the start and end points for a certain working cycles to be verified by analyzing the graph. The stored link angle measurement values are loaded in to the Matlab memory in order to be used as the input to the simulation.

1.3 Current Ongoing Projects on Cranes

In the field of automation, specifically in the case of forestry industry, usage of forwarder is playing a significant role. Some of the ongoing projects are mainly focused on improved control algorithm for the cranes. One of the main aims in such projects is to address the working conditions for the crane operator. By improving the control interface the pressure on the operator can be decreased, possibly with maintaining the crane capacity.

1.4 Problem definition & Goal

The project addresses control of the three most outer links of the crane. Presently these crane links are run based on three joysticks, one per link. All these three links will only control the crane tip in a certain vertical plane. See the crane link illustration in Figure 1. One joystick manipulates lift arm angle. Another joystick controls the Elbow arm angle. The third joystick controls the extension boom length. The swivel link can control the crane perpendicular to the vertical plane. The swivel link is not addressed in this project.

In order to position the crane tip in a plane, one vertical plane, the operator has to control these three different links. This is one more freedom than what is needed to position in two dimensions. To operate three joysticks will put load on the operator compared to the only two needed. Hence, the work for automation of the movement of extension boom has been started. This would reduce the work load for the operator and also increases the time efficiency during the operation.

The thesis project comprises of development of a verification procedure in order to in an efficient way verify the similarity of the two algorithms in the Mat lab system and in the Dasa system.

One goal is to develop a semiautomatic procedure from running the crane with the Dasa system and collecting data from specified variables in the Dasa system and then pre-process the data and load them into the developed Matlab script that compares the values calculated in the Dasa system with the values calculated in the Matlab dSpace system for the same crane trajectory.

The goals of this thesis can be divided into mainly three parts. Primarily, we should develop the required code for pre-processing of data provided by Dasa system using Matlab. Secondarily, we should develop the lab testing script for a specified crane trajectory and verify the results using Matlab. Finally, we need to compare the data set from the Dasa system and from the Matlab simulation system. Then we need judge whether Dasa algorithm works in the same way as the Matlab algorithm. Thus the Dasa algorithm is implemented in master thesis[13].

1.5 Thesis Layout

In the following chapters we have discussed various topics that are related to our thesis and finally how we reached the goals. In chapter 2, the main concentration is about the background of software that was used to implement the algorithm. We have discussed various topics in this chapter. The kinematics of the crane, Crane geometry, Crane coordinates and Forward kinematics and the present forwarders that were in use in forestry purpose.

In this chapter we also discussed in brief about Automatic Prismatic Joint Algorithm Implementation, parameters that were chosen to be configurable and detail discussion about Dasa equipment.

In Chapter 3, is mainly concerned on the topic Interpolation i.e., what is the need of using interpolation technique and how the errors are overcomed by using interpolation with some simulation outputs.

In chapter 4 the conclusion of the project of the verification of crane control algorithm is given.

In chapter 5 we list out the references and in chapter 6 we have discuss the graphical UML diagram representation of the software implemented in the project and flow diagram which also describes the flow diagram step by step instruction.

In chapter 7 we have documented the code and in chapter 8 the simulation result is attached lastly. The output result includes crane simulation when theta2 & theta3 are run simultaneously, compressions of Matlab simulation parameters with real time excel sheet parameters, different elbow end velocity's etc.

2 BACKGROUND

2.1 Code Composer Studio (CCS)

Code Composer Studio is an integrated development environment for developing DSP code for the TMS320 DSP processor and for the processor from Texas instruments. Code composer Studio includes a real time operating system called DSP/BIOS. It has graphical capabilities and is accurate in instruction set simulators. It also contains real time debugging capabilities such as JTAG debugging. Thus, the code composer Studio is user-friendly software that can be used to build a software tool and debug programs. The Dasa control algorithm is developed as a C code in this environment and then interfaced with other software simultaneously**.**

2.2 Automatic Conversion Tools

MATLAB to C Software (MCS), provides the necessary software support tool needed for the automatic conversion of Matlab code to the C code. These tools are good in terms of efficiency and less time consuming procedures in the process of conversion. As a high-level language, Matlab facilitates design exploration. In contrast, programming in C is well suited to optimizing DSPs for performance, memory and processing power. Matlab has several advantages for design exploration, such as matrix-based functions and an interactive programming environment. During translation of an algorithm from Matlab to C, however designers face some important constraints. For eg: Matlab is a dynamically typed and C is a statically typed language.

2.3 Kinematics of the crane

2.3.1 Crane Geometry

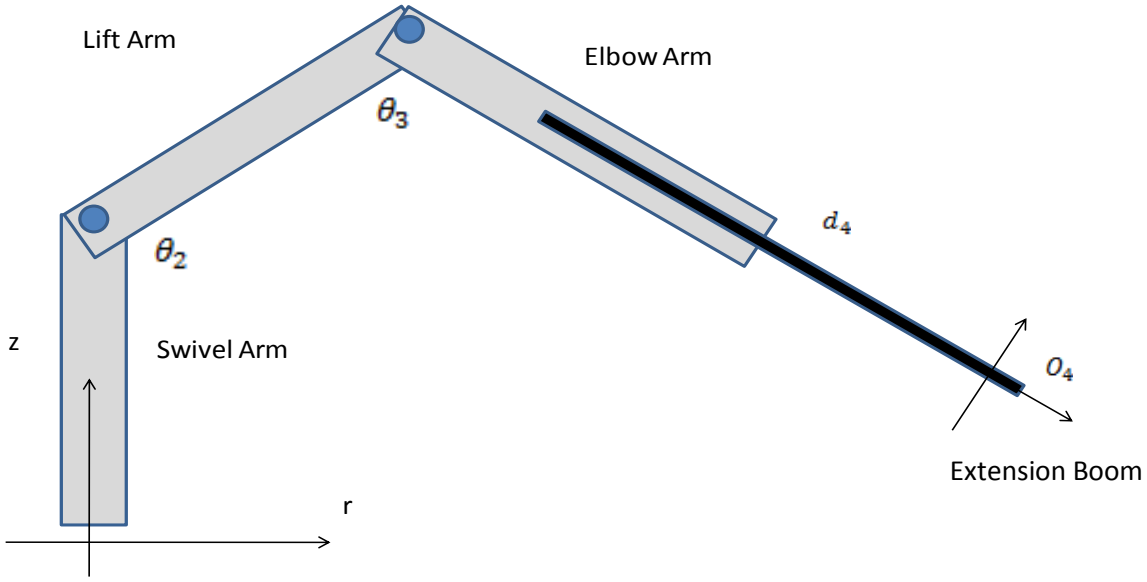

Figure 1: Forward crane geometry in two dimensions.

In Figure 1, the forwarder crane is shown in a two-dimensional structure. The swivel joint θ_1 is excluded in our case and then making the crane tip control into a two dimensional problem. The crane tip position o_4, is represented in Cartesian Coordinates, r and z or in crane coordinates θ_2, θ_3 and d_4. The crane coordinates are indicated in Figure 1. θ_2, θ_3 are the angles of rotation of the respective links, d_4 which is the length of the extension boom.

The crane coordinates are, θ_1 is the swivel arm angle in radians,

θ_2 is the lift arm angle in radians,

θ_3 is the elbow angle in radians,

d_4 is the length of the extension boom in meters,

o_4 is the crane tip position in meters.

2.4 D-H Parameters

Local Cartesian coordinates on each of the links can be defined using different standards. Here we use the Denavite-Hartenberg convention for arranging the local coordinate systems. There are few rules to be taken in consideration during the selection of coordinate system. The Cartesian coordinate system axes are denoted as x-axis, y-axis and z-axis[12].

The coordinate systems are numbered as 1 for the swivel link, 2 for the lift link 3 for the elbow link, and 4 for the extension link.

In the case of prismatic link, the z-axis is defined in the way that prismatic joint is moving only in the along the z-axis. In the case of a revolute joint the z-axis is defined in the way that the rotation axis is oriented as the z-axis.

The x-axis of the coordinate system *i* is chosen to be perpendicular to the z-axis of the coordinate system *i-1*.

The y-axis comes from the x and z-axis by selecting it to be right handed coordinate system.

Once the coordinate frames are specified, inter link transformation are performed by only four parameters. Those are r, d, θ and α.

2.5 Crane Coordinates

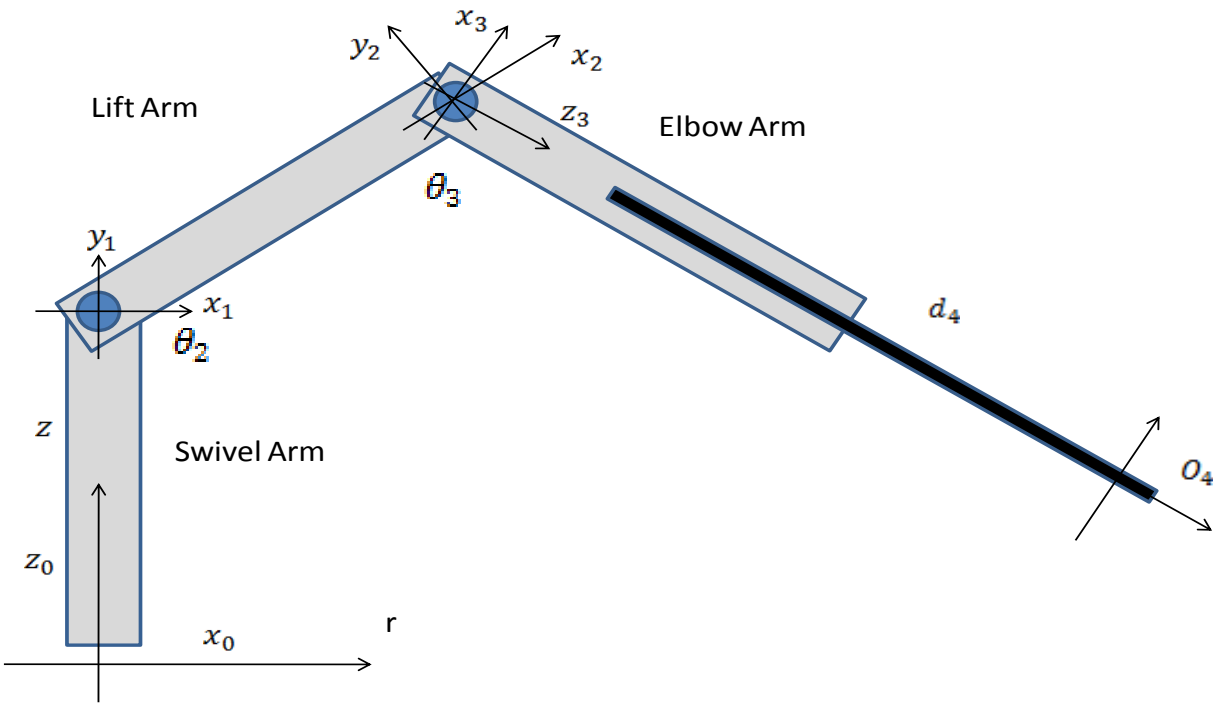

Figure 3: Crane coordinate according to Denavite-Hartenberg Table.

In the figure 3, we can see the crane coordinates in 2 dimensions. x_0 and z_0 are the coordinates for swivel arm. And x_1 and y_1 are the coordinates for lift arm are in parallel to x_0 and z_0 respective. x_2 and y_2 are the coordinates for the lift arm. x_3 and y_3 are the coordinates for elbow arm in parallel to lift arm.

2.6 Forwarder Kinematics

The forward kinematics of the crane can be derived, which will be valid for the full crane running in two dimensions by substituting $\theta_1 = 0$. According to Denavite-Hartenberg, the matrix T which is known as transformation matrix, transforming the coordinates between two subsequent coordinates system from i to i-1 system, is given according to the equation given below,

$$T_{i-1}= \qquad T_i^{i-1} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

The corresponding parameter values of **D-H table** are given in table 1.

DH- Table, Crane	θ	d	a	α
Swivel arm, link1	$\theta_1 = 0$	$d_1 = d_1^*$	$a_1 = 0$	$\alpha_1 = \frac{\pi}{2}$
Lift arm, link2	$\theta_2 = 0$	$d_2 = 0$	$a_2 = a_2^*$	$\alpha_2 = 0$
Elbow arm, link3	$\theta_3 = 0$	$d_3 = 0$	$a_3 = 0$	$\alpha_3 = \frac{\pi}{2}$
Extension link,link4	$\theta_4 = 0$	$d_4 = d_4^*$	$a_4 = 0$	$\alpha_4 = 0$

Table 1. Denavite-Hartenberg Table.

The transformation matrices for the crane model are given based on the D-H table values as, T_0, T_1, T_2, T_3. Where T_0 is defined for the Swivel arm i.e., for link1 by substituting values from DH table.

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And T_1 is defined for Lift arm i.e., for link2 by substituting values from DH table.

$$T_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And T_2 is defined for Elbow arm i.e., for link3 by substituting values from DH table.

$$T_2 = \begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & \cos\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And T_3 is defined for Extension link i.e., for link4 by substituting values from DH table.

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation from the O_4 coordinate system to the O_0 coordinates system is given according to the derivation as,

$$T_4^0 = T_0 T_1 T_2 T_3 =$$

$$\begin{pmatrix} \cos(\theta_2+\theta_3) & 0 & \sin(\theta_2+\theta_3) & a_2\cos\theta_2 + d_4\sin(\theta_2+\theta_3) \\ \sin(\theta_2+\theta_3) & 1 & -\cos(\theta_2+\theta_3) & a_2\sin\theta_2 - d_4\cos(\theta_2+\theta_3) \\ \sin(\theta_2+\theta_3) & 0 & -\cos(\theta_2+\theta_3) & d_1 + a_2\sin\theta_2 - d_4\cos(\theta_2+\theta_3) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The forward kinematic for the crane when θ_1=0 is given according to equation,

$$\begin{pmatrix} x_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} a_2\cos\theta_2 + d_4\sin(\theta_2+\theta_3) \\ d_1 + a_2\sin\theta_2 - d_4\cos(\theta_2+\theta_3) \end{pmatrix} \qquad (2)$$

2.7 AUTOMATIC PRISMATIC JOINT ALGORITHM IMPLEMENTATION:

2.7. 1 Jacobian Matrix:

Applying derivative operation to the corresponding forward kinematic equation 2, we can obtain the Jacobian equation for the crane operating in the two dimensional space spanned by (x_o, z_o).This is performed when swivel joint i.e., θ_1= 0 and the coordinates x_o coincides with the coordinates **r**. Hence the Cartesian velocity relation can be derived as,

$$\mathbf{v} = J(q)\dot{q}$$

Where **v** is Cartesian velocity and $\dot{q}$ is the matrix representing the angular velocity of the each link.

The Jacobian matrix is obtained by using the above expression as,

$$J(q) = \begin{pmatrix} -a_2\sin\theta_2 + d_4\cos(\theta_2+\theta_3) & d_4\cos(\theta_2+\theta_3) & \sin(\theta_2+\theta_3 \\ a_2\cos\theta_2 + d_4\sin(\theta_2+\theta_3) & d_4\sin(\theta_2+\theta_3) & -\cos(\theta_2+\theta_3) \end{pmatrix}$$

Where

$$q = \begin{pmatrix} \theta_2 \\ \theta_3 \\ d_4 \end{pmatrix}$$

The Cartesian velocities in 2D case is given by the above expressions as follows,

$$v = \begin{pmatrix} v_r \\ v_z \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} r \\ z \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} x_0 \\ z_0 \end{pmatrix}$$

and

$$\dot{q} = \begin{pmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{d}_4 \end{pmatrix}$$

2.8 DASA LAB TESTING ENVIRONMENT:

The algorithm of the crane project is developed through various phases. Initially, the algorithm is developed in Visual Studio C++ software in DASA's lab testing environment.

2.9THE CRANE LAB SETUP ENVIRONMENT

2.9.1 Lab Crane Set Up With the dSpace equipment:

In Figure 4, block diagram shows the lab crane system when controlled by the dSpace system. In the Crane Lab system. The crane setup is provided with the dasa5 system units, d5MC and d51012. Among the two d5MC is the main controller which is provided with d5 software. Microcontroller and micro processors are used for communication with the other device. A CAN bus is used to connect d5MC with d51012. This are used to carry out the input and output operations. The d51012 is the one in which actual software is executed for the crane setup, which is connected with the dSpace system. The replacing of the dSpace system with the dasa system is described very much focused on next chapter.

Using the dSpace System in Crane Lab Set up:

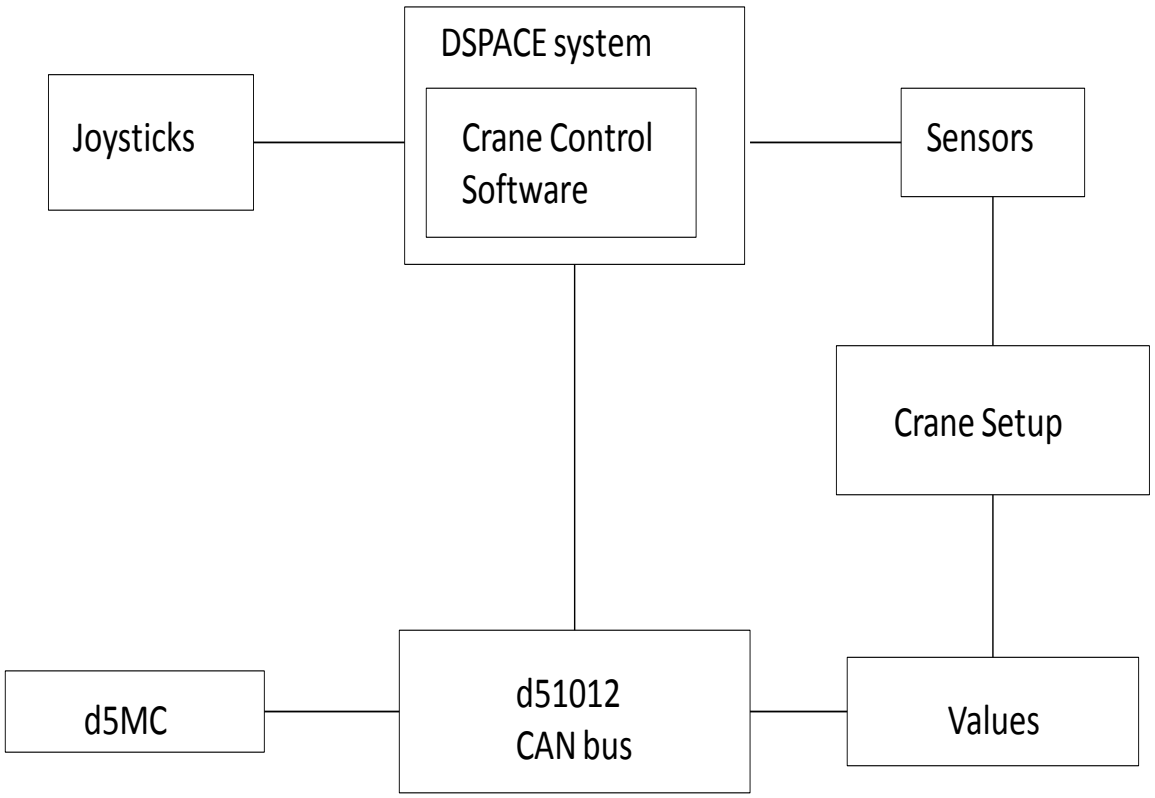

Figure 4: Lab crane system with dSpace system

2.9.2 Replacement of dSpace system with dasa5 Equipment:

In the figure 5, the joysticks are connected to the D51048 unit, which is an i/o unit generally used for low power application. Then the D51048 is connected to D5mc unit by using CAN bus, which is the main controller used for controlling various units by using a DSP processor. It is also used as interface with the crane software(i.e., crane controal algorithm). Once the crane software is interfaced with D5mc unit, now it is connected to the d5c112 unit which is also an i/o unit used to control various parameters. Now the d5c112 is connected to the crane valves to obtain the eight current outputs.

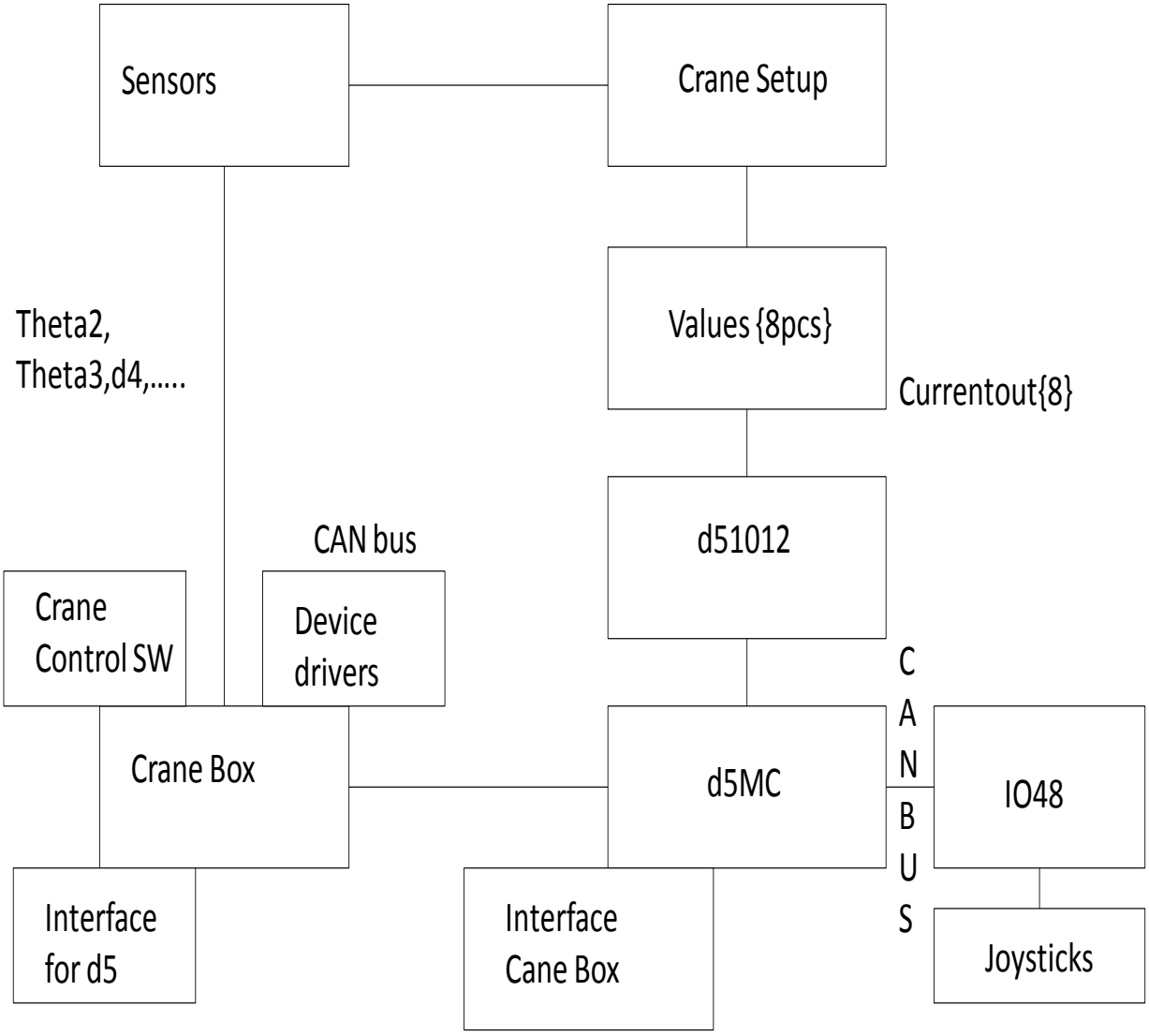

Figure 5: Lab crane system with dasa5 system

3 HOW THE ERRORS ARE OVERCOMED BY USING INTERPOLATION:

Our first task was to extract the excel sheet values generated by the Dasa system to Matlab. The DASA date sampling system generates only shortly undisturbed sequences, some seconds. Plotting the data, e.g., for theta2, theta3 and d4 respectively, we observe that for some instant of time there is sudden change in the graph. The main reason behind this problem may be due to loss of data in the data sampling as it was extracted from Dasa system.

The loss of data in the Dasa system may occur due to the following reason i.e., Shortage of primary memory space in the Dasa system forces to write a sampled data batch into the hard disk. During the writing period the sampling of data is not taking place and information is lost. So we have used interpolation technique in matlab to reconstruct the data which was loosed during the operation.

3.1 Exporting values to Graph

Figure 6 is a collection of 3 signals, green indicates theta2 and blue line represents theta3 based on those values we can see simultaneously changing red line that is extension boom.

In Figure 6, some period of time are plotted with circles. Those plotted circles are the time period where there is a sudden change in data for certain time interval. We can overcome this problem by using any of the three techniques. They are

- Interpolation.
- By reducing the number of parameters during the Crane lab experiment.
- By enhancing the memory size of the Dasa system.

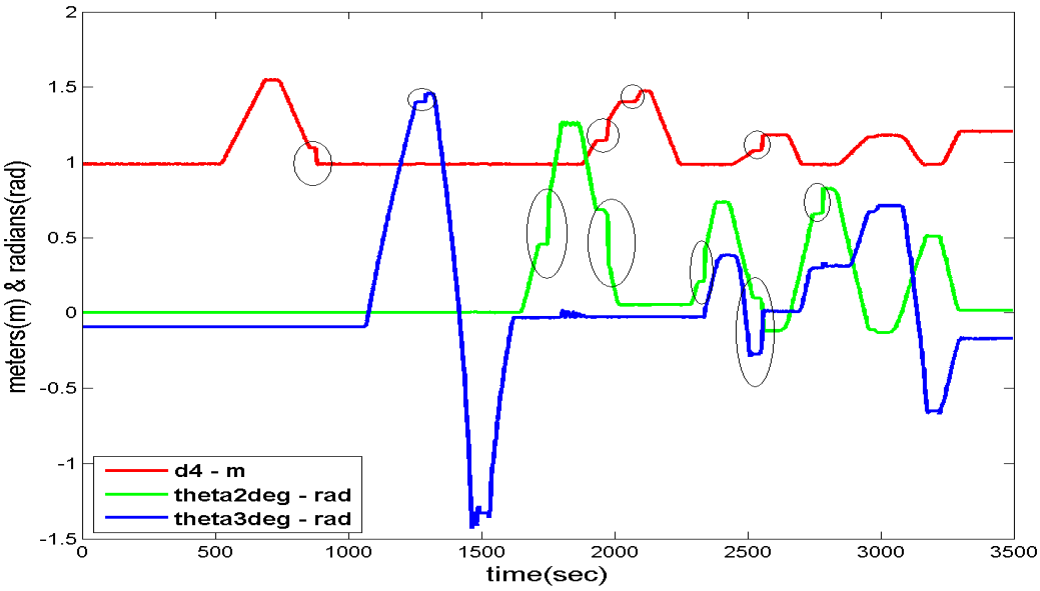

Figure 6. Before Interpolation with circles(where circles represents missing data).

3.2 INTERPOLATION:

Interpolation is a technique used for constructing new data points within the range of discrete set of known data points. There are different types of interpolation techniques used for various application, such has linear interpolation, piecewise constant interpolation and polynomial interpolation etc. In general linear interpolation is used for numerical analysis by applying different mathematical calculations.

In most of the cases liner interpolation technique is used to reconstruct the lost data for a particular cycle of time which may caused due to some technical problem raised during the operation.

In our project we have used this technique to reconstruct the lost dat. During this process when the crane is operated depending on the crane movement the related parametric values are stored in an external memory, provided by the Dasa systems. During this process due to shortage of memory space all the parametric values are not stored in the external memory.

The following Figure 7. is the simulation output for the values extracted from the excel sheet for theta2, theta3 and extension boom (before interpolation). Here we can observe there is some data is missing.

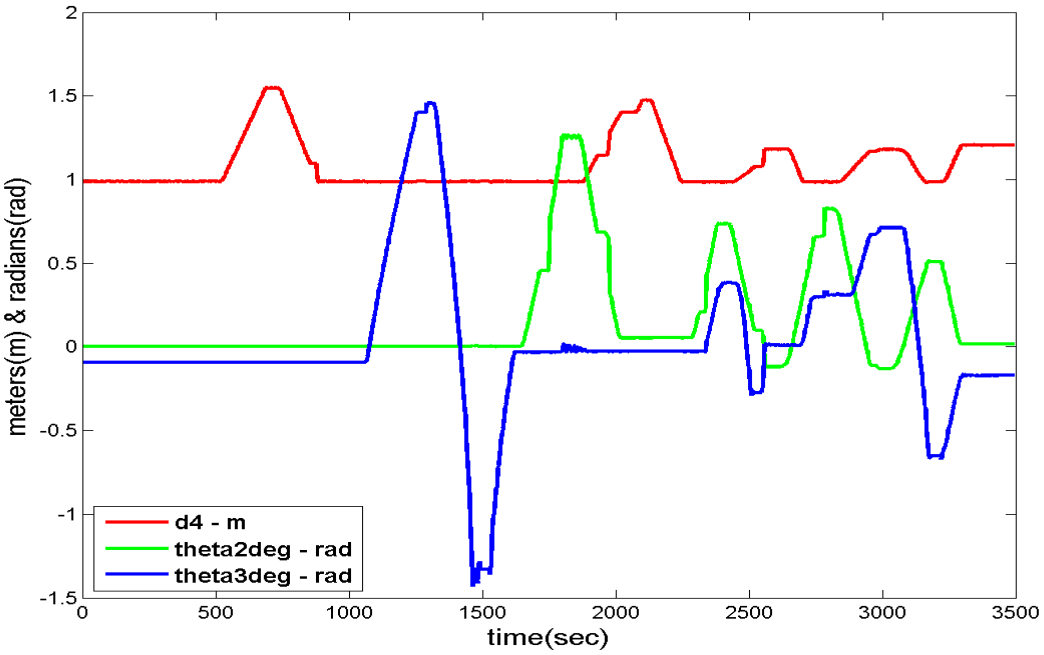

Figure 7. Before Interpolation.

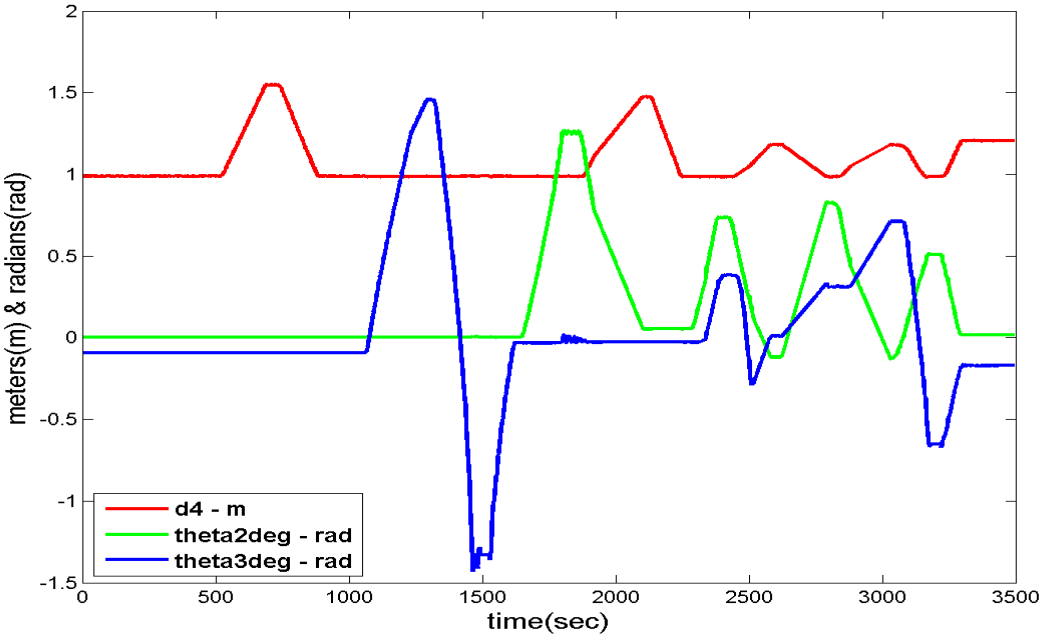

Figure 8. After Interpolation.

In the above diagram, the green line indicates theta2, the blue line indicates theta3 and the red line indicates d4 (extension boom).

4. Comparison of Theoretical and practical values:

After using interpolation technique we started to compare practical values with theoretical values by using Matlab. Practical values are exported to Matlab and theoretical values are directly generated. In this chapter we can see how the extension boom is moved based on the movement of the two joints (theta2 and theta3). Extension boom will reach to its maximum position when theta2 decreases and theta3 increase and visa-versa.

4.1 Crane Trajectory & Extension Boom movement:

The extension boom movement is based on the movements of theta2 and theta3 respectively. If you refer the Figure 9, the extension boom movement will start when theta2 decreases and theta3 increases simultaneously. In our project the starting point for the extension boom is 200 sec. This is the point from where the extension boom is stretched to its maximum position as the angle of theta2 decreases and theta3 increases to its maximum extent. The extension boom will come back to its original position when theta2 increases and theta3 decreases simultaneously. In the Figure 9, if you observe around 800 sec the extension boom came back.

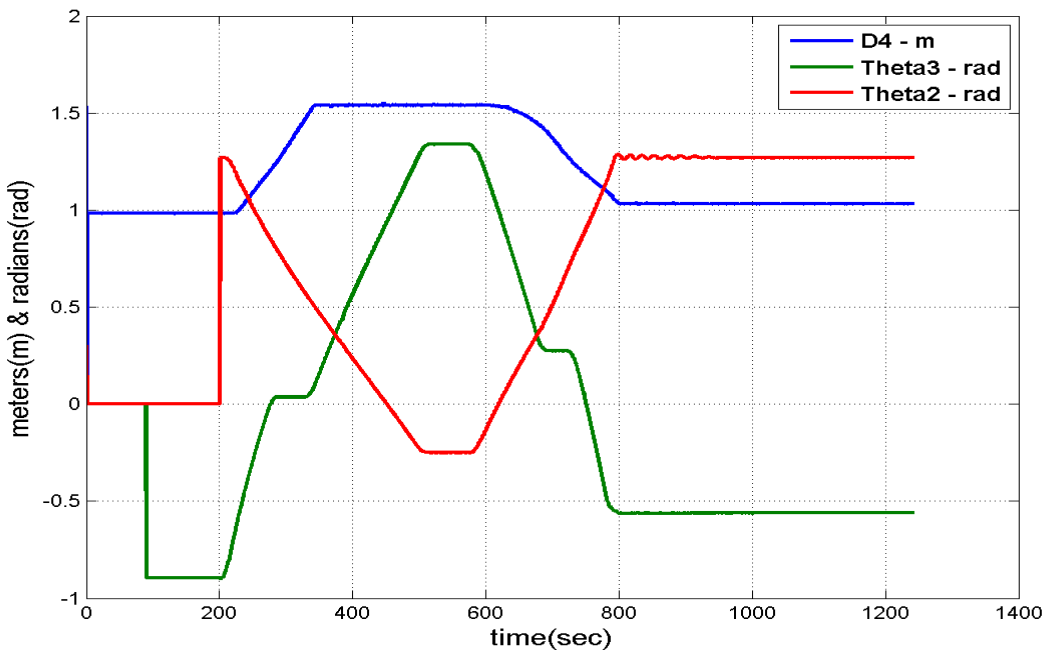

Figure 9. At the time instant 200 a stretching of the links is starting. Between 500 and 600 the rane stays contant in its stretched position. From 600 to 800 it folds back.

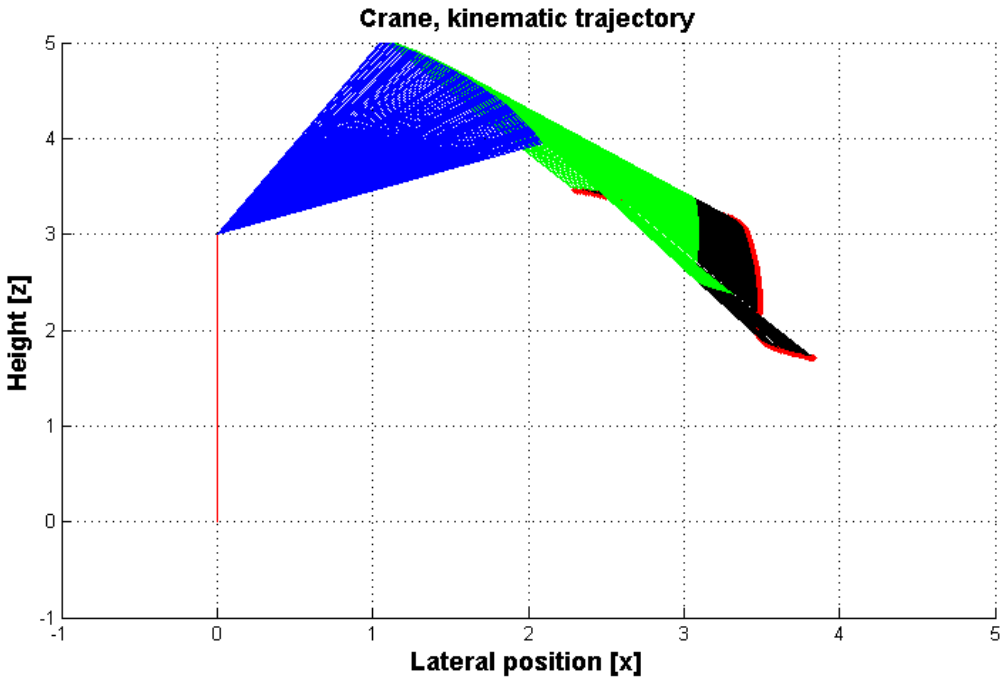

Figure 10. Movement of extension boom based on theta2, theta3 and the extension boon. The lift arm is coloured blue, the elbow arm is coloured green, the extension arm is coloured black, and the top tip is coloured red. The folded crane tool tip is sutured at the coordinates (2.3,3.5) and the stretched at (3.8,1.7)

In the Figure 10, the change of link variables shown in Figure 9 between the time instants 400 and 600 seconds is shown in animation in Matlab.

4.2 Comparisons:

As comparison is the elbow end velocity chosen. Another data set is used compared to the one shown in figure 9. A Matlab model of the crane is fed with the angular measurement from a crane experiment. Using the angular measurements the elbow end velocity in cylindrical coordinates are simulated in Matlab, shown as red in Figure 11 and Figure 12. The simulated data is compared to the calculated values of the elbow end velocity performed in the DASA control system. It can be seen in Figure 11, that the elbow end velocity has similar behaviour in the DASA calculations as in the Matlab calculations. This implies that the algorithms are almost equal.

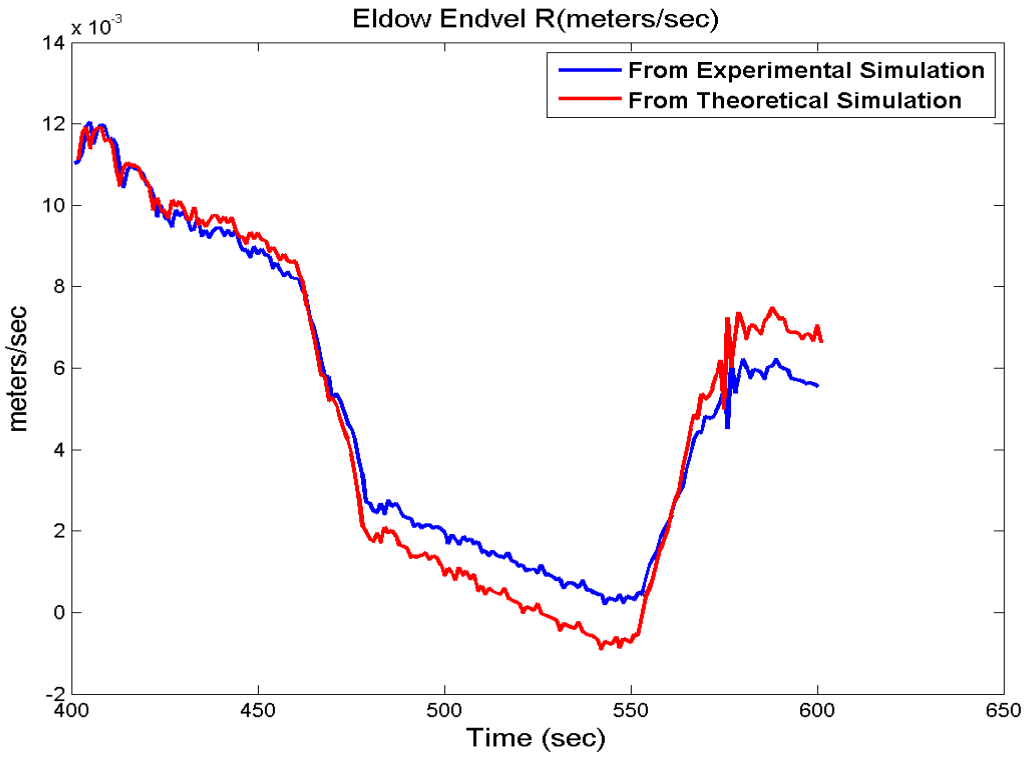

Figure 11: Elbow End Velocity R(meter/sec). The Matlab algorithm and the DASA algorithms are fed with the same theta 2 and theta 3 data.

`

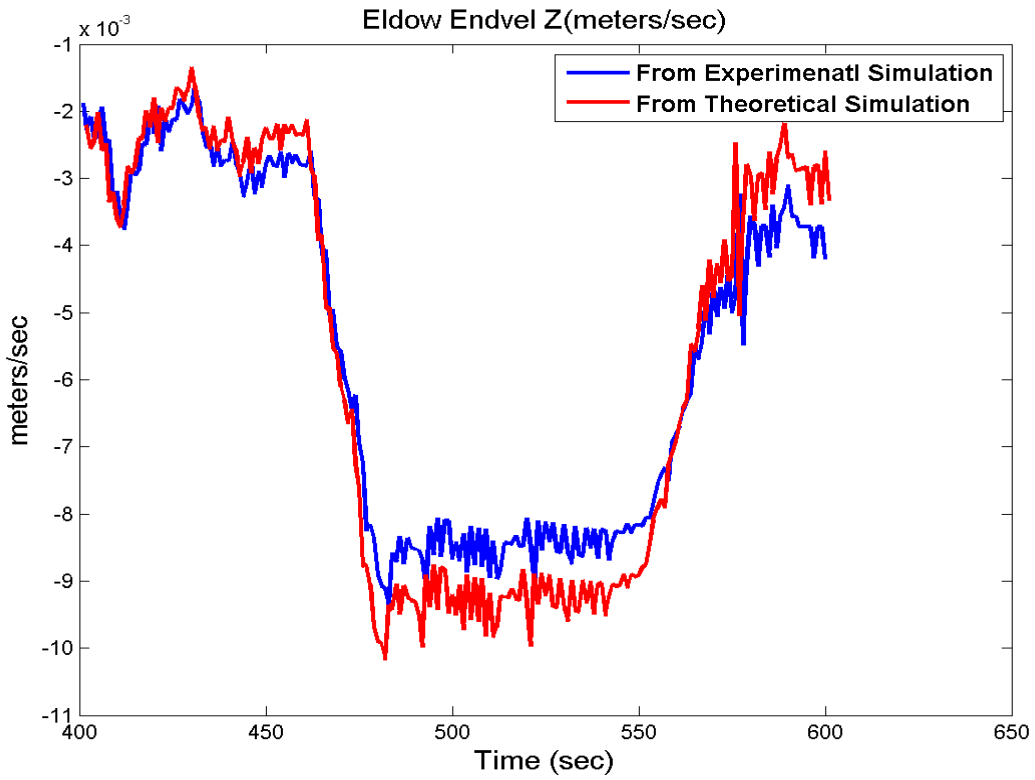

Figure 12 : Elbow End Velocity Z(meter/sec). The Matlab algorithm and the DASA algorithms are fed with the same theta 2 and theta 3 data.

5 CONCLUSIONS:

To achieve a specific control function running a forwarder crane, we have verified the similarity of two algorithms in Matlab system and in DASA system. For that we have developed pre-processing code for data and testing script for crane trajectory and verified results using Matlab. The algorithm is developed in lab crane system and it shows characteristics similar to that of running the algorithm in an automated system. We have verified the function of two implementations of a control algorithm running on two different systems, Matlab and DASA system. After running the algorithm on two different systems under the required conditions, we have compared the result sets from DASA system and Matlab simulation system. It is concluded from the results that the algorithm performs in an similar way both the simulation environments in Matlab and in the real time crane control electronics DASA.


BLEKINGE TEKNISKA HÖGSKOLA
BTH

6 REFRENCES

[1] M.W Spong, M. Vidyasagar, Robot Modelling and Control, Wiley, 2006.

[2] Hari Krishna N. and Mahender R. K., "Observer design for the Base-Compliance of a Hydraulic Crane", Master thesis, University of Kalmar, September 2005.

[3] Sigvardsson M. and Olsson T., "Modelling and Simulation of a Hydraulic Crane", Master thesis, University of Kalmar, February 2005.

[4] Ekevid T.'On optimal control of hydraulic crane", Proc.19th Nordic Seminar on Computational Mechanics, NSCM-19, Lund, Sweden, October 2006, pp 105-115.

[5] Fazuluila M. and Srikanth K., "Mathematical modelling and simulation in Dymola of a laboratory crane", Master thesis, University of Kalmar, June 2006.

[6] Heinze A., " Friction modelling in a hydraulic crane", Master thesis, Vaxjo university, 2007.

[7] Zhamykhanova A.B. "Modelling of a laboratory crane", Master thesis, University o Kalmar, 2008.

[8] Pim Mennon, Project report, Fontys Hochschule, Eindhoven, The Netherlands.

[9] Narra Suresh. "Modelling and simulation of a laboratory crane", Master thesis, In progress Blekinge Tekniska Hogskola.

[10] Rick Simon, Project report, Fontys Hochschule, Eindhoven, The Netherlands.

[11] Available: http://www.dasa.se/

[12] Avaiable: http://en.wikipedia.org/wiki/Denavit-Hartenberg_Parameters

[13] Naga Praveen Parchuru and Jagadeesh Thati., "DSP Implementation of a Control Algorithm for a Forwarder Crane", Master thesis, Blekinge Insititute of Techonology, 2009


BLEKINGE TEKNISKA HÖGSKOLA
BTH

7 UML DIAGRAM:

7.1 Flow diagram for Mat lab Simulation:

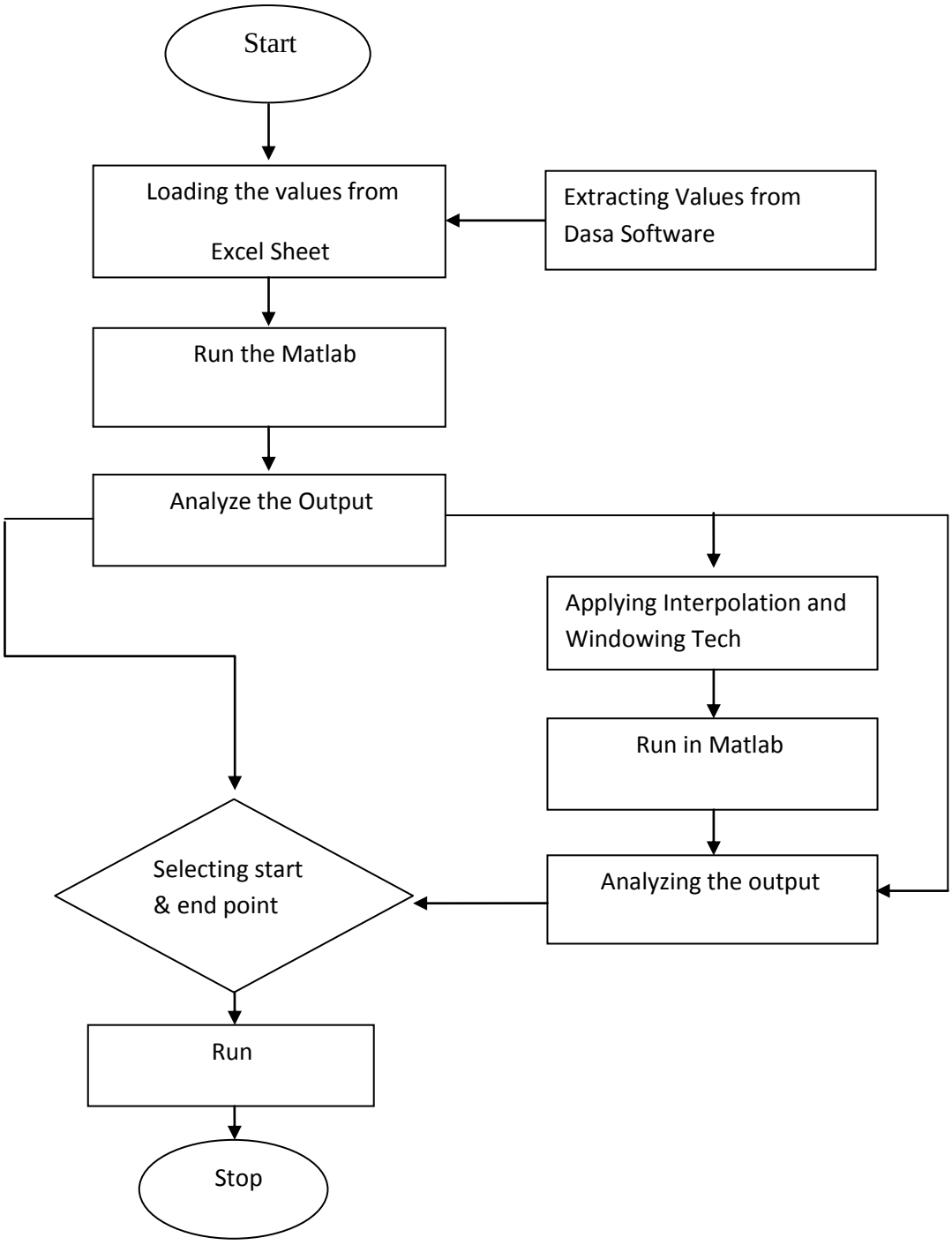

Figure 17. Flow Diagram.

7.2 FLOW DIAGRAM STEP BY STEP INSTRUCTION:

- In the lab when the crane is operated in various directions depending on the crane arm moments different parametric values are generated, this values are stored in the external disk provided by Dasa system. This values are retrieved from the external disk in the form of excel sheet.
- The collected excel sheet values are loaded in matlab and runned.
- Once the excel sheet values are runned in matlab. We have to analyze the output and we have to judge whether all crane arm moments are runned in the proper way.
- If they are not proper then we need apply interpolation and different windowing technique to retrieve all the values which are lost for a particular interval of time. Run in Matlab. If there are no errors, select the start point and end point. This start point and end point may be theta2 or theta3 depending on the crane simulation requirement.
- After analyzing the output if there are some errors such as missing of data for certain cycle of time. For such cases we need apply interpolation and different windowing technique to retrieve all the values which are lost for a particular interval of time.
- Again run the values in Matlab. If there are no errors, select the start point and end point. This start point and end point may be theta2 or theta3 values depending on the crane simulation requirement.
- Run the matlab M-file and stop.

8 Algorithm:

```
%Transformation matricies
T10=dht(theta1,d1,a1,alpha1);
T21=dht(theta2,d2,a2,alpha2);
T32=dht(theta3,d3,a3,alpha3);
T43=dht(theta4,d4,a4,alpha4);

%Links
swivelarm=T10*[0 0;0 -swivellength;0 0;1 1];
liftarm=T10*T21*[0 -liftlength;0 0;0 0;1 1];
elbowarm=T10*T21*T32*[0 0;0 0;0 elbowlength;1 1];
prismaticarm=T10*T21*T32*T43*[0 0;0 0;0 -d4+elbowlength;1 1];

%Links Cartesian
swcart=hom2cart(swivelarm);
licart=hom2cart(liftarm);
elcart=hom2cart(elbowarm);
prcart=hom2cart(prismaticarm);

%Forward kinematics
x=liftlength*cos(theta2)+(0+d4)*sin(theta2+theta3);
z=swivellength+liftlength*sin(theta2)-(0+d4)*cos(theta2+theta3);

%Forward velocity calculated by the Jacobian
q=[theta2;theta3;d4];
qdot=[dtheta2;dtheta3;dd4];
v=qdot2v(liftlength,elbowlength,q,qdot);

%Forward velocity calculated by numeric derivative of o4 position
dx=x-xold;
dz=z-zold;


%World coordinatesystem xz
figure(1)
plot(swcart(1,:),swcart(3,:),'r')
plot(licart(1,:),licart(3,:),'b')
plot(elcart(1,:),elcart(3,:),'g')
plot(prcart(1,:),prcart(3,:),'k')
plot(x,z,'r.')

figure(2)
%plot(k,dx,'b*',k,dz,'go',k,v(1),'m*',k,v(2),'co',k,dd4,'kx')
plot(k,dd4,'kx')

%Animation sampling
pause(0.5)

%dd4desire calculation
gain=1;
dd4m=1;
dtheta2m=1;
dtheta3m=1;
d4c=2;
d4e=4;
```

```
lambda=lambdafun(d4c,d4e,d4,dd4)
qdotm=[dtheta2m;dtheta3m;dd4m];
dd4desire=dd4desirefun(gain,qdotm,lambda,liftlength,elbowlength,q,qdot)

dd4=min([max([-dd4m dd4desire]) dd4m])

%Angle and length update
theta1 = theta1 + dtheta1;
theta2 = theta2 + dtheta2;
theta3 = theta3 + dtheta3;
d4=d4 + dd4;

%o4 position update
xold=x;
zold=z;

end
figure(1)
hold off
figure(2)
hold off
```

DHT Function:

```
function T=dht(theta,d,a,alpha)
Rz=[cos(theta) -sin(theta) 0 0
    sin(theta) cos(theta) 0 0
    0 0 1 0
    0 0 0 1];
Tz=[1 0 0 0
    0 1 0 0
    0 0 1 d
    0 0 0 1];
Tx=[1 0 0 a
    0 1 0 0
    0 0 1 0
    0 0 0 1];
Rx=[1 0 0 0
    0 cos(alpha) -sin(alpha) 0
    0 sin(alpha) cos(alpha) 0
    0 0 0 1];
T=Rz*Tz*Tx*Rx;
```

Homogeneous to Cartesian coordinate:

```
function cart=hom2cart(arm)
cartx=arm(1,:)./arm(4,:);
carty=arm(2,:)./arm(4,:);
cartz=arm(3,:)./arm(4,:);
cart=[cartx;carty;cartz];
```

Crane Jocobian in 2 dimensional case:

```
function v=qdot2v(liftlength,elbowlength,q,qdot)
%Crane Joacobian in the 2 dimensional case, theta1=0
%liftlength
%elbowlength
%q=[theta2;theta3;d4]
```

```
%dq=[dtheta2;dtheta3;dd4]

theta2=q(1);
theta3=q(2);
d4=q(3);%
J=[-liftlength*sin(theta2)+d4*cos(theta2+theta3) d4*cos(theta2+theta3)
sin(theta2+theta3)
   liftlength*cos(theta2)+d4*sin(theta2+theta3) d4*sin(theta2+theta3) -
cos(theta2+theta3)];
J1=[-liftlength*sin(theta2)+d4*cos(theta2+theta3) d4*cos(theta2+theta3)
sin(theta2+theta3)
   liftlength*cos(theta2)+d4*sin(theta2+theta3) d4*sin(theta2+theta3) -
cos(theta2+theta3)];
v=J*qdot;
```

Lamda Function

```
function lambda=lambdafun(d4c,d4e,d4,dd4)
%weighting function for prismatic velocity
%d4c=center position of prismatic link
%d4=prismatic link position
%dd4=prismatic link velocity

if d4>d4c
    if dd4>0
        lambda=(d4e-d4)/(d4e-d4c);
    else
        lambda=1;
    end
elseif dd4<0
    lambda=1-(d4c-d4)/(d4c);
else
    lambda=1;
end
```

dd4 Desire Function

```
function dd4desire=dd4desirefun(gain,qdotm,lambda,liftlength,elbowlength,q,qdot)
%function calculating the desired velocity of the prismatic link
%gain is a adjustable parameter
%qdotm = maximum speed of the joints 2, 3 and 4
%lambda = weighting function output
%q = crane coordinates
%qdot

%Jacobian
%Forward velocity calculated by the Jacobian

qdot23=[qdot(1);qdot(2);0];
v23=qdot2v(liftlength,elbowlength,q,qdot23);

d4hat=[0;0;1];
v4=qdot2v(liftlength,elbowlength,q,d4hat);

qdot23m=[qdotm(1);qdotm(2);0];
```

```
v23m=qdot2v(liftlength,elbowlength,q,qdot23m);

qdot4m=qdotm(3);
v4m=qdot2v(liftlength,elbowlength,q,qdot4m);

%dd4 desire
dd4desire=gain*norm(v4m)/norm(v23m)*lambda*v23'*v4;
```

Crane simulation by applying interpolation and windowing technique:

```
clear all
close all
clc

data = xlsread('KranData2010Mars02');
scale=[1;1e-3;1e-5;1e-3;1e-4;1e3;(1/4)*1e-4;(1/4)*1e-4;1e-6;1e-6;1e-2;1e-2;1e-5;1e-
5;1e-3];
datacal=data;
for k=2:14
    datacal(:,k)=data(:,k).*scale(k);
end

data2 = (round(100*(datacal(:,1))));

t = 0:1:round(100*datacal(end,1));
s = length(t);
data_imp = zeros(s,15);

data_imp(:,1) = t';
j = 1;

for i = 1:round(100*datacal(end,1))
    while data_imp(i) == data2(j);
        data_imp(i,2:15) = datacal(j,2:15);
        j = j+1;
    end
end

if data_imp(end,1) == data2(j)
    data_imp(end,2:15) = datacal(j,2:15);
end

missnum = find(data_imp(:,2) == 0);
mn = (missnum - 1);

data_imp(missnum(1:end),:) = [];

for k = 1:size(mn)
    data_imp1 = interp2(2:15,data_imp(:,1),data_imp(:,2:15),2:15,mn(k));
    datamn(k,:) = data_imp1;
end

x = [mn datamn];
data_exp = [data_imp;x];
data_srt = sortrows(data_exp,1);
```

```
figure()
grid on

plot(data_srt(:,2),'r');
hold on
plot(data_srt(:,13),'g');
hold on
plot(data_srt(:,14));
legend('d4','theta2deg','theta3deg',3);
hold off

data1 = data_srt;

%4240
q1 = [1421:1470 2051:2150 2855:2930 3200:3500 3855:3900 4200:4300 4375:4650
4800:5050];
data1(q1,:) = [];

for k = 1:1201
    data_imp1 = interp2(2:15,data1(:,1),data1(:,2:15),2:15,q1(k));
    datamn1(k,:) = data_imp1;
end

x1 = [q1' datamn1];
data_exp1 = [data1;x1];
data_srt1 = sortrows(data_exp1,1);

figure()

plot(data_srt1(:,2),'r');
hold on
plot(data_srt1(:,13),'g');
hold on
plot(data_srt1(:,14));
legend('d4','theta2deg','theta3deg',3);
hold off


der_d4 = diff(data_srt1(:,2));
der_theta2deg = diff(data_srt1(:,13));
der_theta3deg = diff(data_srt1(:,14));

figure()
plot(der_d4,'r');
hold on
plot(der_theta2deg,'g');
hold on
plot(der_theta3deg);

legend('d4','theta2','theta3',3)
%
% % Plotting derivatitive of d4
%
% figure()
% plot(der_d4,'r');
% title('Derivative of d4');
%
```

```
% %Plotting derivatitive of thete2 deg
%
% figure()
% plot(der_theta2deg,'g');
% title('Derivative of theta2');
%
% %Plotting derivatitive of thete3 deg
%
% figure()
% plot(der_theta3deg);
% title('Derivative of theta3');


l = 20;
w = window(@hamming,l);
figure();
der_d4conv = conv(der_d4,w);
plot(der_d4conv)
title('Hamming Window for d4 ');

figure();
der_theta2 = conv(der_theta2deg,w);
plot(der_theta2)
title('Hamming Window for Theta2 ');

figure();
der_theta3 = conv(der_theta3deg,w);
plot(der_theta3)
title('Hamming Window for Theta3');

## Crane Simulation for Theta2

%Laboratory crane
%Experiment 2

% Loading the data values from the previous output
% Differnating those values and assigning
% This for Theta2 from 2800 to 3500
% Instal values are theta2 = 0.21 and theta3 = -0.03



clear all
close all
clc

load data;

%Link length
swivellength=3;
liftlength=2.3;
elbowlength=2;

%DH-table
%row 1
theta1=0; d1=swivellength; a1=0; alpha1=pi/2;
```

```
%row 2
theta2=0; d2=0; a2=liftlength; alpha2=0;

%row 3
theta3=0; d3=0; a3=0; alpha3=pi/2;

%row 4
theta4=0; d4=elbowlength; a4=0; alpha4=0;

%Initial conditions
theta1=0;
theta2=0.21;
theta3=-0.03;
 d4=elbowlength;

%Movement, incremental
dtheta1=0;
dtheta3 = diff(data_srt1(2800:3500,14));
dtheta2 = diff(data_srt1(2800:3500,13));
dd4 = 0;

xold=0;
zold=0;
N=700;

%Figures
figure(1),clf, axis([-1 5 -1 5]),grid,hold on
title('Crane, kinematic trajectory, go to a logg on the ground')
xlabel('lateral position, x'),ylabel('height, z')


%Animation loop
for k = 1:N

%Transformation matricies
T10=dht(theta1,d1,a1,alpha1);
T21=dht(theta2,d2,a2,alpha2);
T32=dht(theta3,d3,a3,alpha3);
T43=dht(theta4,d4,a4,alpha4);

%Links
swivelarm=T10*[0 0;0 -swivellength;0 0;1 1];
liftarm=T10*T21*[0 -liftlength;0 0;0 0;1 1];
elbowarm=T10*T21*T32*[0 0;0 0;0 elbowlength;1 1];
prismaticarm=T10*T21*T32*T43*[0 0;0 0;0 -d4+elbowlength;1 1];

%Links Cartesian
swcart=hom2cart(swivelarm);
licart=hom2cart(liftarm);
elcart=hom2cart(elbowarm);
prcart=hom2cart(prismaticarm);

%Forward kinematics
x=liftlength*cos(theta2)+(0+d4)*sin(theta2+theta3);
z=swivellength+liftlength*sin(theta2)-(0+d4)*cos(theta2+theta3);

%Forward velocity calculated by the Jacobian
```

```
q=[theta2;theta3;d4];
qdot=[dtheta2(k);dtheta3(k);dd4];
v=qdot2v(liftlength,elbowlength,q,qdot);

%Forward velocity calculated by numeric derivative of o4 position
dx=x-xold;
dz=z-zold;


%World coordinatesystem xz
figure(1)
plot(swcart(1,:),swcart(3,:),'r')
plot(licart(1,:),licart(3,:),'b')
plot(elcart(1,:),elcart(3,:),'g')
plot(prcart(1,:),prcart(3,:),'k')
plot(x,z,'r.')

%Animation sampling

%dd4desire calculation
gain=1;
dd4m=1;
dtheta2m=1;
dtheta3m=1;
d4c=2;
d4e=4;
lambda=lambdafun(d4c,d4e,d4,dd4)
qdotm=[dtheta2m;dtheta3m;dd4m];
dd4desire=dd4desirefun(gain,qdotm,lambda,liftlength,elbowlength,q,qdot)

dd4=min([max([-dd4m dd4desire]) dd4m])

%Angle and length update
theta1 = theta1 + dtheta1;
theta2 = theta2 + dtheta2(k);
theta3 = theta3 + dtheta3(k);
d4=d4 + dd4;

%o4 position update
xold=x;
zold=z;

end
figure(1)
hold off


## Crane Simulation for Theta3:

%Laboratory crane
%Experiment 1

% Loading the data values from the previous output
% Differnating those values and assigning
% This for Theta3 from 1800 to 2700
% Instal values are theta3 = 0.06 and theta2 = 0
% Max angle for theta3 =  85deg and min = - 83
```


```
clear all
close all
clc

load data;

%Link length
swivellength=3;
liftlength=2.3;
elbowlength=2;

%DH-table
%row 1
theta1=0; d1=swivellength; a1=0; alpha1=pi/2;

%row 2
theta2=0; d2=0; a2=liftlength; alpha2=0;

%row 3
theta3=0; d3=0; a3=0; alpha3=pi/2;

%row 4
theta4=0; d4=elbowlength; a4=0; alpha4=0;

%Initial conditions
theta1=0;
theta2=0;
theta3=0.06;
 d4=elbowlength;

%Movement, incremental
dtheta1=0;

dtheta3 = diff(data_srt1(1800:2700,14));
dtheta2 = diff(data_srt1(1800:2700,13));


dd4 = 0;

xold=0;
zold=0;
N=900;

%Figures
figure(1),clf, axis([-1 5 -1 5]),grid,hold on
title('Crane, kinematic trajectory, go to a logg on the ground')
xlabel('lateral position, x'),ylabel('height, z')

figure(2),clf, axis([0 N -.05 .1]),grid,hold on
title('Crane, kinematic trajectory, go to a logg on the ground')
xlabel('k'),ylabel('o4 velocity')

%Animation loop
for k = 1:N

%Transformation matricies
T10=dht(theta1,d1,a1,alpha1);
T21=dht(theta2,d2,a2,alpha2);
T32=dht(theta3,d3,a3,alpha3);
T43=dht(theta4,d4,a4,alpha4);

%Links
```

```
swivelarm=T10*[0 0;0 -swivellength;0 0;1 1];
liftarm=T10*T21*[0 -liftlength;0 0;0 0;1 1];
elbowarm=T10*T21*T32*[0 0;0 0;0 elbowlength;1 1];
prismaticarm=T10*T21*T32*T43*[0 0;0 0;0 -d4+elbowlength;1 1];

%Links Cartesian
swcart=hom2cart(swivelarm);
licart=hom2cart(liftarm);
elcart=hom2cart(elbowarm);
prcart=hom2cart(prismaticarm);

%Forward kinematics
x=liftlength*cos(theta2)+(0+d4)*sin(theta2+theta3);
z=swivellength+liftlength*sin(theta2)-(0+d4)*cos(theta2+theta3);

%Forward velocity calculated by the Jacobian
q=[theta2;theta3;d4];
qdot=[dtheta2(k);dtheta3(k);dd4];
v=qdot2v(liftlength,elbowlength,q,qdot);

%Forward velocity calculated by numeric derivative of o4 position
dx=x-xold;
dz=z-zold;


%World coordinatesystem xz
figure(1)
plot(swcart(1,:),swcart(3,:),'r')
plot(licart(1,:),licart(3,:),'b')
plot(elcart(1,:),elcart(3,:),'g')
plot(prcart(1,:),prcart(3,:),'k')
plot(x,z,'r.')
%Animation sampling

%dd4desire calculation
gain=1;
dd4m=1;
dtheta2m=1;
dtheta3m=1;
d4c=2;
d4e=4;
lambda=lambdafun(d4c,d4e,d4,dd4);
qdotm=[dtheta2m;dtheta3m;dd4m];
dd4desire=dd4desirefun(gain,qdotm,lambda,liftlength,elbowlength,q,qdot);


dd4=min([max([-dd4m dd4desire]) dd4m]);

%Angle and length update
theta1 = theta1 + dtheta1;
theta2 = theta2 + dtheta2(k);
theta3 = theta3 + dtheta3(k);
d4=d4 + dd4;

%o4 position update
xold=x;
zold=z;

end
figure(1)
hold off
```

Crane simulation and comparison without applying interpolation and windowing technique:


```
clear all
close all
clc

%Link length
swivellength=3;
liftlength=2.3;
elbowlength=2;

data = xlsread('Kranmätning2010-10-27_143308 (hämtning)_A2');

%************************************************

figure(1);
plot(data(:,5),'r');
hold on
plot(data(:,3),'g');
hold on
plot(data(:,4));
hold off
grid
legend('D4','Theta2','Theta3',3)

%************************************************

N1 = input('Enter starting point : ');
N2 = input('Enter ending point : ');

%************************************************

t = 1:N2-N1;

figure(2)
%subplot(121)
plot(data(N1:N2-1,10));
hold on
title('Eldow Endvel R(rad/s)');

figure(3)
%subplot(121)
plot(data(N1:N2-1,11));
hold on
title('Eldow Endvel Z(rad/s)');

figure(4)
%subplot(121)
plot(data(N1:N2-1,12));
hold on
title('PrismEndVelR(rad/s)');

figure(5)
%subplot(121)
plot(data(N1:N2-1,13));
hold on
title('PrismEndVelZ(rad/s)');

figure(8)
plot(data(N1:N2-1,6));
hold on

figure(9)
plot(data(N1:N2-1,7));
hold on
```

```
figure(10)
plot(data(N1:N2-1,2));
hold on

figure(11)
plot(data(N1:N2-1,5));
hold on

figure(12)
plot(data(N1:N2-1,9));
hold on

figure(13)
plot(data(N1:N2-1,8));
hold on

% Extracting data from Kranmätning2010-10-27_142919_A2

% DH-table
% row 1
theta1=0; d1=swivellength; a1=0; alpha1=pi/2;

% row 2
theta2=0; d2=0; a2=liftlength; alpha2=0;

% row 3
theta3=0; d3=0; a3=0; alpha3=pi/2;

% row 4
theta4=0; d4=elbowlength; a4=0; alpha4=0;

%Initial conditions
theta1 = 0;
theta2 = data(N1,3);
theta3 = data(N1,4);
d4 = elbowlength;

dtheta1 = 0;
dtheta2 = diff(data(N1:N2,3));
dtheta3 = diff(data(N1:N2,4));
dd4 = 0;

xold = 0;
zold = 0;
N = N2-N1;

%Figures
figure(6),clf, axis([-1 5 -1 5]),grid,hold on
title('Crane, kinematic trajectory, go to a logg on the ground')
xlabel('lateral position, x'),ylabel('height, z')

figure(7),clf, axis([0 N -.05 .1]),grid,hold on
title('Crane, kinematic trajectory, go to a logg on the ground')
xlabel('k'),ylabel('o4 velocity')

%Animation loop
for k = 1:N

%Transformation matricies
T10 = dht(theta1,d1,a1,alpha1);
T21 = dht(theta2,d2,a2,alpha2);
T32 = dht(theta3,d3,a3,alpha3);
```

```
T43 = dht(theta4,d4,a4,alpha4);

%Links
swivelarm = T10*[0 0;0 -swivellength;0 0;1 1];
liftarm = T10*T21*[0 -liftlength;0 0;0 0;1 1];
elbowarm = T10*T21*T32*[0 0;0 0;0 elbowlength;1 1];
prismaticarm = T10*T21*T32*T43*[0 0;0 0;0 -d4+elbowlength;1 1];

%Links Cartesian
swcart = hom2cart(swivelarm);
licart = hom2cart(liftarm);
elcart = hom2cart(elbowarm);
prcart = hom2cart(prismaticarm);

%Forward kinematics
x = liftlength*cos(theta2)+(0+d4)*sin(theta2+theta3);
z = swivellength+liftlength*sin(theta2)-(0+d4)*cos(theta2+theta3);

%Forward velocity calculated by the Jacobian
q = [theta2;theta3;d4];
qdot = [dtheta2(k);dtheta3(k);dd4];
v = qdot2v(liftlength,elbowlength,q,qdot);

%Forward velocity calculated by numeric derivative of o4 position
dx = x-xold;
dz = z-zold;


%World coordinatesystem xz
figure(6)
plot(swcart(1,:),swcart(3,:),'r')
plot(licart(1,:),licart(3,:),'b')
plot(elcart(1,:),elcart(3,:),'g')
plot(prcart(1,:),prcart(3,:),'k')
plot(x,z,'r.')

figure(7)
%plot(k,dx,'b*',k,dz,'go',k,v(1),'m*',k,v(2),'co',k,dd4,'kx')
plot(k,dd4,'kx')

%Animation sampling
%pause(0.5)

%dd4desire calculation
gain = 1;
dd4m = 1;
dtheta2m = 1;
dtheta3m = 1;
d4c = 2;
d4e = 4;
lambda = lambdafun(d4c,d4e,d4,dd4);
qdotm = [dtheta2m;dtheta3m;dd4m];
[dd4desire v23 v4 i]= dd4desirefun(gain,qdotm,lambda,liftlength,elbowlength,q,qdot);


ElbowEndVelR(k) = v23(1);
ElbowEndVelZ(k) = v23(2);

PrismEndVelR(k) = v4(1);
PrismEndVelZ(k) = v4(2);
DD4desire_sim(k) = dd4desire;
i_sim(k) = i;

dd4 = min([max([-dd4m dd4desire]) dd4m]);
```


```
%Angle and length update
theta1 = theta1 + dtheta1;
theta2 = theta2 + dtheta2(k);
theta3 = theta3 + dtheta3(k);
d4 = d4 + dd4;

d4_sim(k) = d4;
dd4_sim(k) = dd4;

%o4 position update
xold = x;
zold = z;

end

ElbowR = 80*ElbowEndVelR;
ElbowZ = 80*ElbowEndVelZ;
PrismR = 1000*PrismEndVelR;
PrismZ = 1000*PrismEndVelZ;


figure(2)
grid on
plot(t+2,ElbowR,'r');
hold off
title('ElbowEndVelR(rad/s) From Simulation');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(3)
grid on
plot(t+2,ElbowZ-0.25,'r');
hold off
title('ElbowEndVelZ(rad/s) From Simulation');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(4)
grid on
plot(PrismR,'r');
hold off
title('PrsimEndVelR(rad/s) From Simulation');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(5)
grid on
plot(PrismZ,'r');
hold off
title('PrismEndVelZ(rad/s) From Simulation');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(8)
plot(t+2,100*dtheta2,'r');
hold off
title('dTheta2');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(9)
plot(t+2,100*dtheta3,'r');
hold off
```


```
title('dtheta3');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(10)
plot((2000*DD4desire_sim)+0.8,'r');
hold off
title('dd4 desire');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(11)
plot((d4_sim/1.64)-0.060,'r');
hold off
title('d4');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

figure(12)
plot(t+2,(i_sim*100),'r');
hold off
title('dd4 desire raw');
grid
legend('From Excel Sheet','From Matlab Simulation',2)

dd4_sim1 = diff(dd4_sim)/0.01;

figure(13)
plot((10*dd4_sim1),'r');
hold off
title('dd4');
grid
legend('From Excel Sheet','From Matlab Simulatoin',2)
```