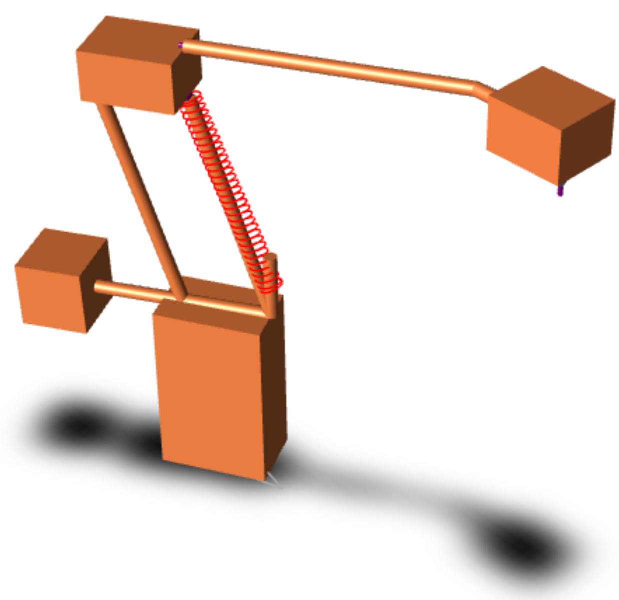
MAS416
Modeling and Simulation of Mechatronic Systems:

Six Degrees of Freedom Robotic Arm

UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE



Magnus T. Ekløff, Benjamin Knutsen,
Kristoffer S. Tronstad & Omer Zec

University of Agder
Faculty of Engineering and Science
26. November, 2017

Preface

The project is a part of the course MAS416 - Modeling and Simulation of Mechatronical Systems, for Master in Mechatronics at University of Agder.

The task in hand is to create a dynamic model of an industrial robot from ABB using SimulationX.

Contents

List of Figures

List of Tables

1 Introduction

The project objective is to create and simulate a 6 DOF industrial robot arm in SimulationX, generate a tool path in Matlab and complete the path as fast as possible. The group decided for the robot to complete the path within 5 seconds and keep the cost to a minimum.

In the project there were given four point of which the tool-tip must intersect during the motion. In addition the body IJ must remain horizontal($\pm 0.2°$) for the operation to be successful.

2 Model of Robot

The model was built in accordance with the project description. A dynamic model of an ABB industrial robot was created. This robot has six degrees of freedom and contains a parallel linkage system. A constraint from point F to point G is essential to obtain a closed loop in the parallel linkage system, as the body EFG has two parents. The finalized SimulationX 3D-model is shown in figure 1.

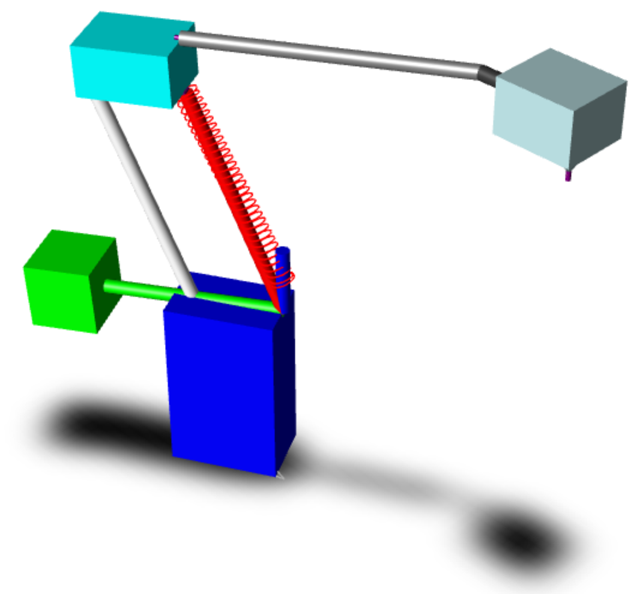


Figure 1: Model of Robot in SimulationX

Below in figure 2 corresponding SimulationX block diagram of the system is shown.

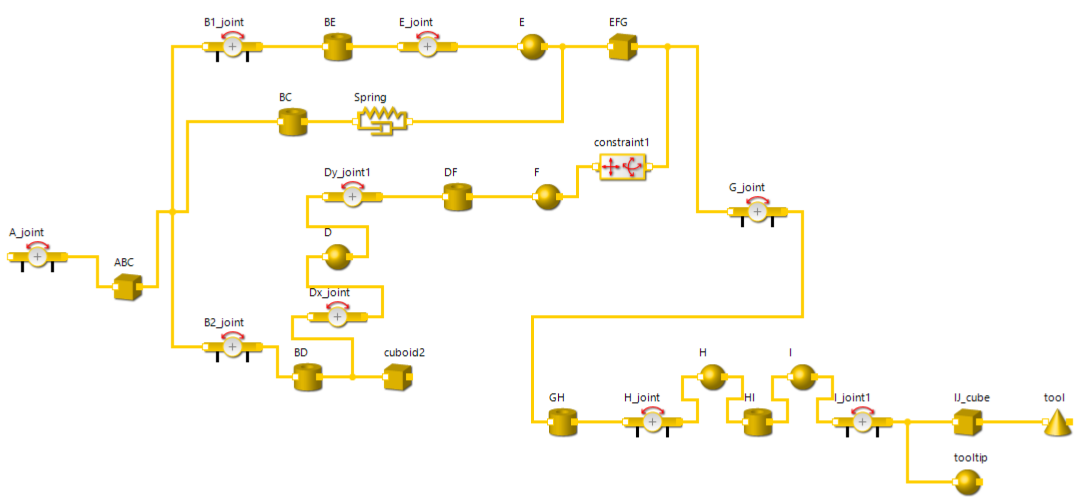


Figure 2: Block Diagram

3 Path

A tool path, corresponding to the given points was generated in Matlab using a polynomial approach. The time precision used when logging the robot was $0.1s$. The data collected after running the polynomial estimation where exported to a .csv text file. Then the csv-file was imported to SimulationX.

The preset function makes the robot follow the generated path and outputs the position, velocity and acceleration for the traced joint or point in the model. The collected data sets, was used to find the motor and gearbox needed in the different joints.

3.1 3D Path

Consider the problem of moving the tool from the initial position to the end position in a predefined time. In order to do this, a polynomial for each of the three dimensions are needed.

The degree, n, of a polynomial describes the behavior and characteristics of the movement. In order to maintain a continuous acceleration profile, with initial and finale values for the acceleration, a polynomial of at least fifth order has to be created. Which results in six boundary conditions. This is only necessary in the last segment of the path since the acceleration in the final point must be *zero* for the robot to come to rest. The first two segments can be described with a fourth order polynomial, where the kinematics in the last point, is the start kinematics in the next path.

For the two first segments of the path, a matrix with five rows is constructed as shown in equation 1.In addition matrix of six rows is constructed in equation 2 for the third and final segment. The three first rows describes position, velocity and acceleration for the initial point. While the last three describes position, velocity and acceleration for the end point for a arbitrary constant T.

$$\begin{bmatrix} X_0 \\ X_0' \\ X_0'' \\ X_T \\ X_T' \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 \\ 0 & 1 & 2t & 3^2 & 4t^3 \\ 0 & 0 & 2 & 6t & 12t^2 \\ 1 & T & T^2 & T^3 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 \end{bmatrix} \cdot \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} X_0 \\ X_0' \\ X_0'' \\ X_T \\ X_T' \\ X_T'' \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 \\ 0 & 1 & 2t & 3^2 & 4t^3 \\ 0 & 0 & 2 & 6t & 12t^2 \\ 1 & T & T^2 & T^3 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 \\ 0 & 0 & 2 & 6T & 12T^2 \end{bmatrix} \cdot \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{bmatrix} \tag{2}$$

3.2 Path Optimization

A simple path can be constructed by moving in a straight line between two points and select the velocity at the last point to zero. This path will be highly ineffective, as the robot will come to rest in each point. As a result of stopping in every point and then accelerating rapidly to next point, the robots main parts, such as gears and bearings, will be worn out quicker. This may also result in bigger deflections and backlashes in the robot while operating other precision tasks. Moving from one point to another in a straight line path, requires more energy and time than a smooth continuous path.

To ensure great performance and reliability, there was construct a smother path. The method used to generate the best path is polynomial estimation. The path must be designed in a way that the robot intersects every point with a accuracy of $\pm 1mm$.

To create a visualization of the path, a Matlab script where created and plotted. The script used the matrices mentioned in the previous chapter. In addition the script creates a .csv file with the coordinated for the tool tip to follow. The coordinates was imported into SimulationX. It also creates a 3D-plot, shown in figure 3, for the path the tool tip will follow.

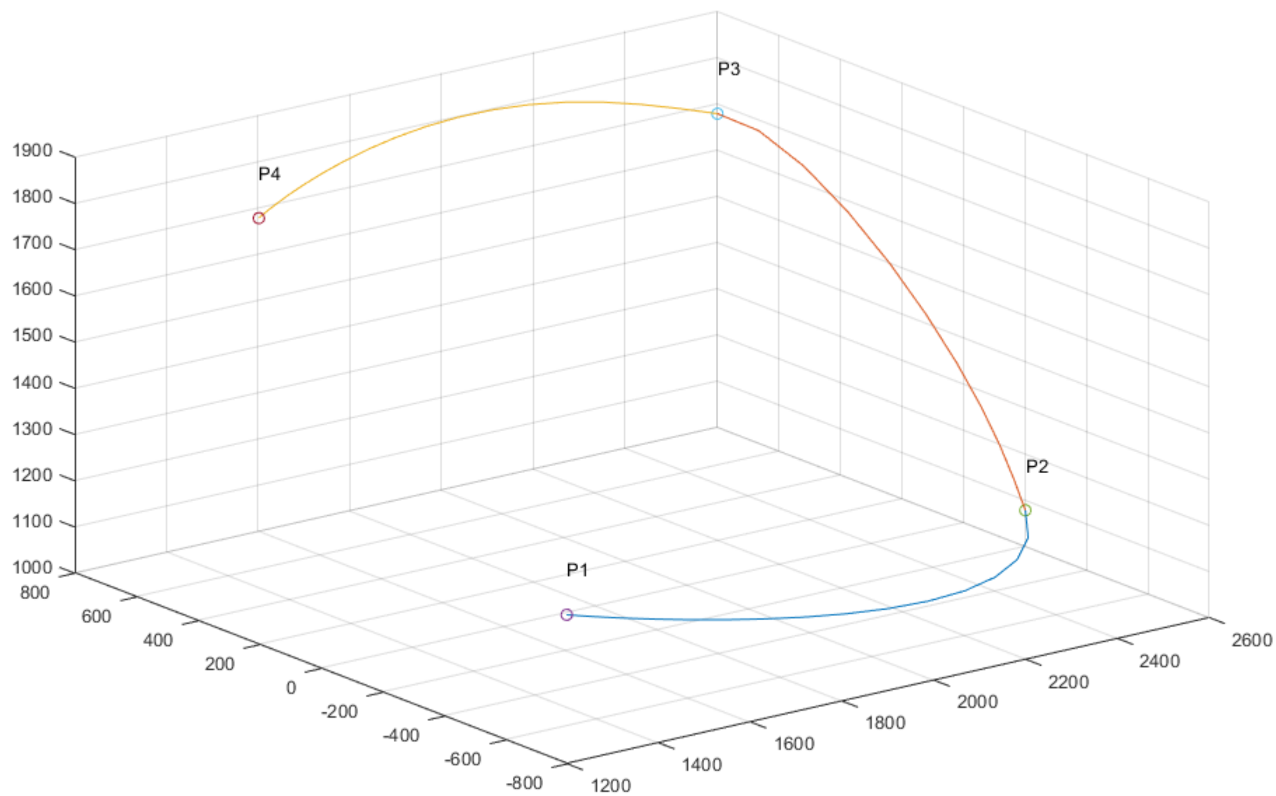


Figure 3: 3D Model of The Path

The full Matlab scripts' .m file is attached in the digital version uploaded.

3.3 Final Path

To ensure better adjustment of the path, two additional points were added to the path. By having the extra points it will be easier to change the path to maximize the robots performance.

Figure 4 show the finalized path used to further optimize the robots motion.

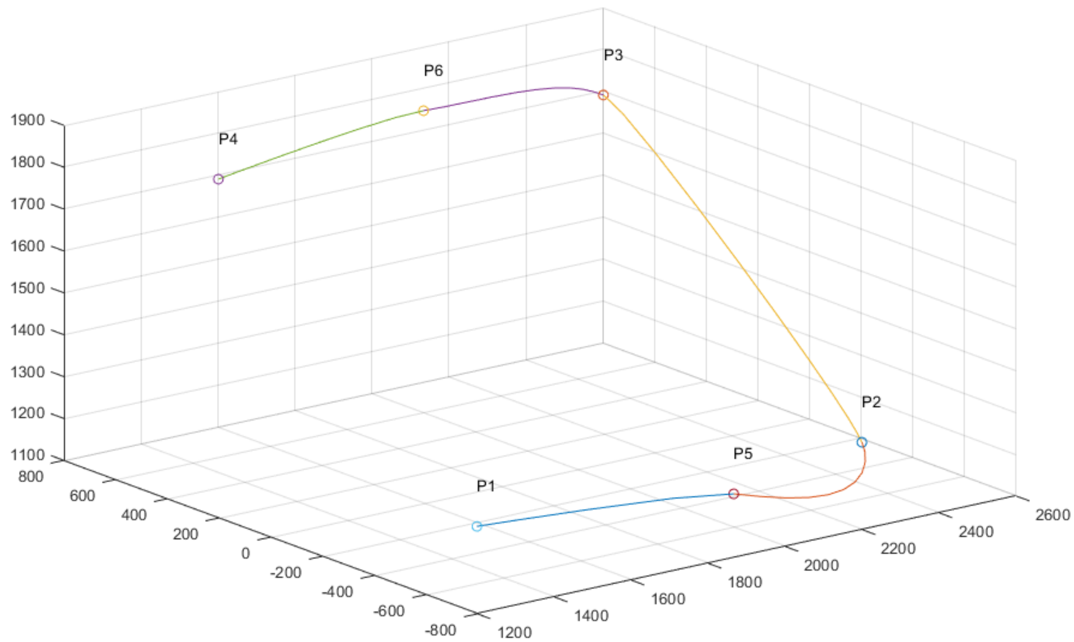


Figure 4: Final Path

In the final path, the robot gathers speed by swiping down in between P5 and P2. This makes for a smoother transition in one of the critical points of the motion.

P6 is used in a similar fashion. It is positioned further in the y-direction to smooth out what would be a sharper turn.

4 Inverse Kinematics

To calculate a motion for the robot for a specific motion for the tool-tip, inverse kinematics is necessary. A copy of the robot is created, and is forced to copy the original robots movement. Since the tool path is already exported to SimulationX, the robot can be forced to follow the tool path. The motion of the joints can be recorded and the necessary data can be used to calculate which gearboxes and motors necessary to achieve the desired characteristics. The robot was forced to follow the path with a preset for each axis in the defined space. Or as dean Michael R. Hansen called it in his lecture, "The Hand of God". The block diagram for this setup is shown in figure 5.

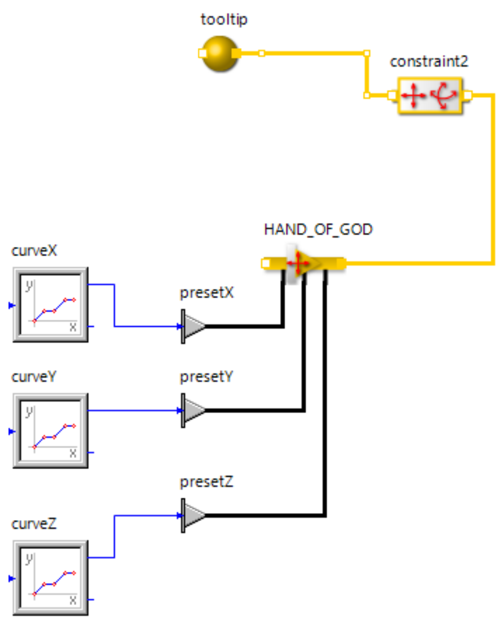


Figure 5: Hand of God

5 Simulation

In this chapter gearbox and servo motors are selected for each joint. The gears and motors must be chosen in consideration of price and speed.

5.1 Choosing Servo Motor

When choosing a servo motor for each joint a "trail-and-error"-method is used. It begins by using the Hand of God models' angular velocity for each joint into their respective joint in the "realistic" model.

Figure 6 show the block diagram of the servo control for the robot. All control circuit are modeled in the same manner. It starts by using the angular velocity as an input function for the servo motor which is then sent to a gear-module with a per-fixed ratio. The the angular rotation is then measured (and multiplied with the gear ratio) and used as a reference to the angular rotation from the Hand of God model. The difference is the fed through a PID-controller and added to the angular velocity to correct the error that may ocour.

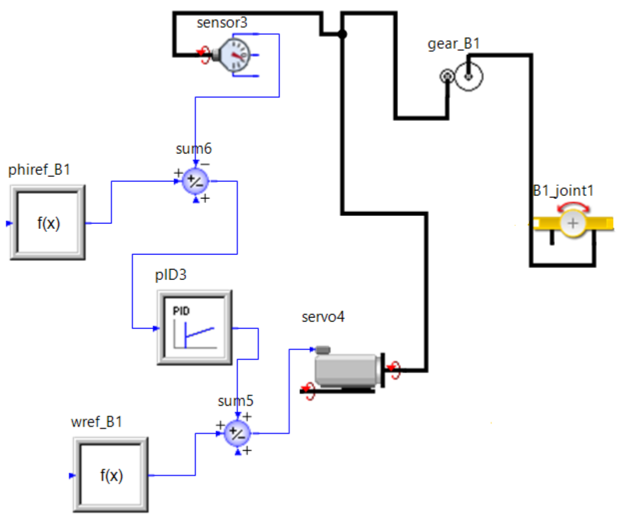


Figure 6: Block Diagram of Servo Controller

Table 1 show the servo motors and gearboxes chosen befor tuning the regulators.

Table 1: Servo and Gears

Joint	**Servo** $[Nm/A]$	**Gear ratio**
A	1.5	200
B1	1.5	200
B2	1.5	200
G	1.5	200
H	1.5	100
I	1.5	100

5.2 Tuning

Further on the tuning of the controller were done. It were done by changing the velocity and acceleration for each point in the Matlab-script, which then exported a new set of coordinates to the Hand of God-model. The angular rotation of the joints in the Hand of God-model were piloted versus the realistic models'. Then the two graphs were compared to figure out what needed to be changed in the PID-controller to correct the result.

Figure 7 shows the angular rotation of revolute joint $B1$ before tuning. As seen on the graph, the green line from the realistic model is offset from the red Hand of God model.

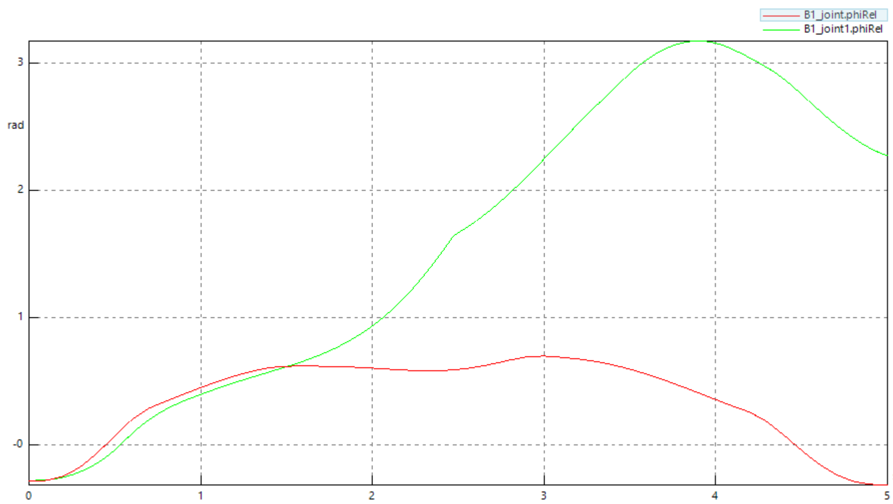


Figure 7: Angular Rotation of Joint B1 in Hand of God and Realistic Model

By decreasing the integrating time on the PID-controller the realistic model will follow the Hand of God exact. The results after tuning the PID-controller is shown in figure 8.

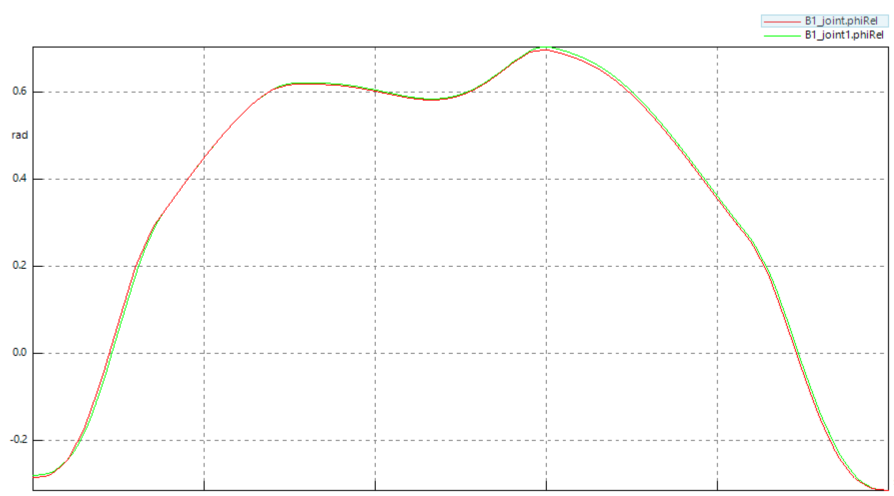


Figure 8: Angular Rotation of Joint B1 in Hand of God and Realistic Model After Tuning

It is now seen that the lines are following each other exactly, which mean that the realistic-model is equal to the Hand of God-model.

5.3 Final Model

After optimizing and motor tuning a final model were presented. The model can be view in detail in the SimulationX-file attached.

The finalized motors and gearbox is displayed in table 2.

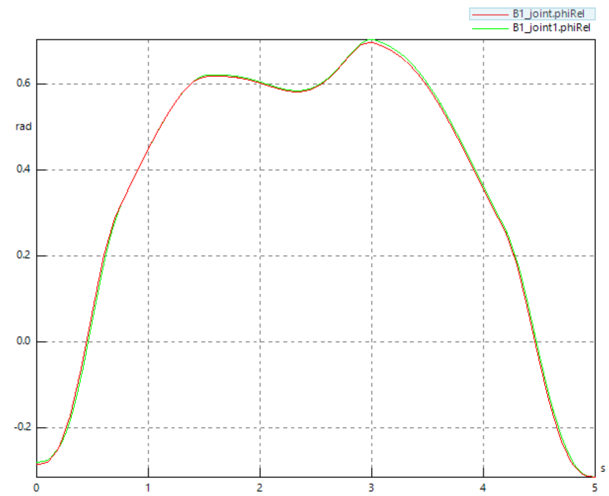
Table 2: Finalized Servo and Gears

Joint	**Servo** $[Nm/A]$	**Gear ratio**	**Dimensionless Cost**
A	1.5	200	4
B1	1.5	200	4
B2	1.5	200	4
G	0.5	50	2.25
H	0.5	50	2.25
I	0.5	50	2.25
Sum			18.75

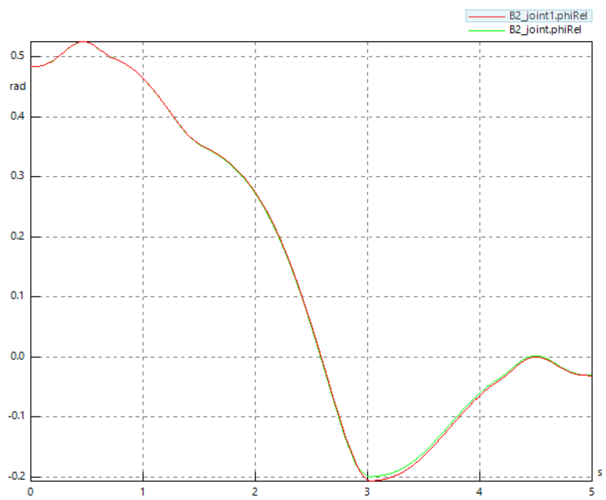
Which renders a total cost factor of 18.75.

Figure 9, on the next page, show the angular rotation of every revolute joint in the Hand of God model versus the Realistic model. It can be seen from the plots that both models ration are equal.

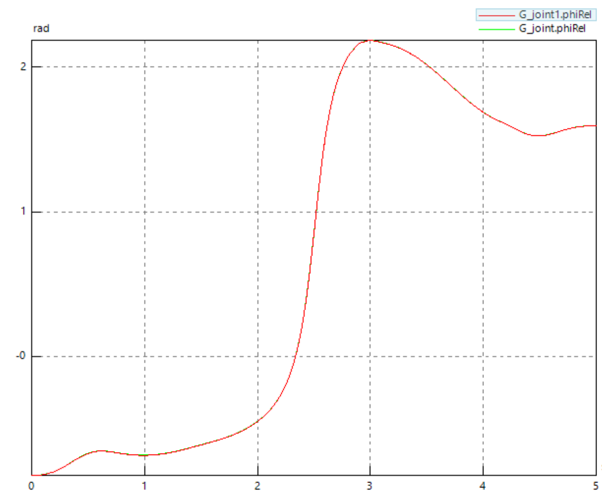
(a) Revolute Joint A



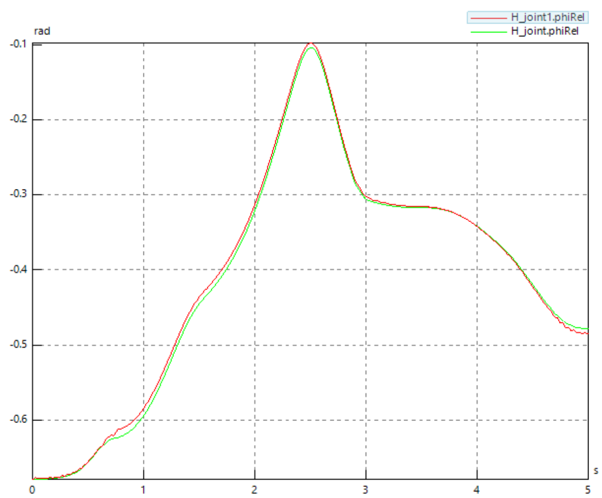
(b) Revolute Joint B1



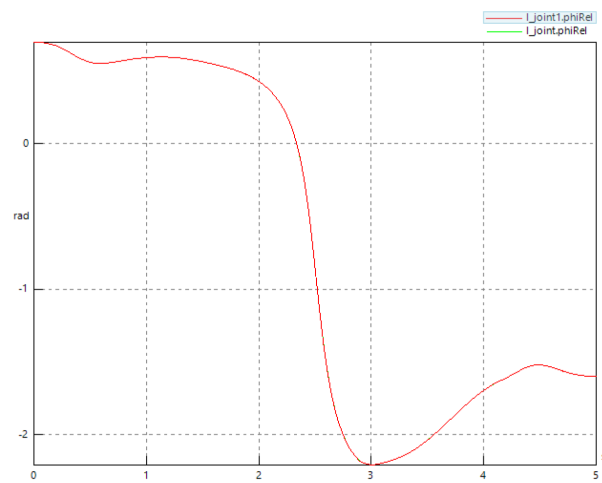
(c) Revolute Joint B2



(d) Revolute Joint G



(e) Revolute Joint H



(f) Revolute Joint I

Figure 9: Angular Rotation of Revolute Joints

Lastly figure 10 shows the difference in position of the tool tip in the Hand of God model(yellow) versus the realistic model(blue).

Figure 10: Tooltip End Position

From the graph it can be seen that the tips end position is $0.3mm$ off in the realistic model.

6 Discussion

There are a lot of different combination relative to speed and cost for this robot to perform the motion. The group has chosen to prioritize a fast robot above a cheap one.

The path had to be more optimized, since the motors could not follow the first path. By implementing two extra points in the robot path, it allowed us to control the robot behaviour before running trough the given points. In the first path only the velocities in the endpoint could be determined, and it was difficult for the motors to follow the motions. It resulted in a robot that missed almost all the points. The reason why the motors could not follow the first path, were the required torques. The torque requested from the "Hand of God" were much higher than the motors could produce in combination with the gearbox. The path was to rough in the first place and it became more smoother when the extra points were implemented. The required torque was reduced and the motors could follow. There was also added a PID-controller for those motors that could not follow the path in some regions.

The robot accuracy was not good in the beginning. After optimization, the robot went trough all the points with very small errors. The biggest error in the accuracy was in the ending point. The robot started off with 15mm error in the ending point. After PID-tuning, the robot ended with a error of $0.3mm$. This was acceptable and the end path was determined.

7 Conclusion

The robot followed the path and simulated the motion, as described in the task. The tool tip intersected the four points, within the given criterion of $\pm 1mm$. At the end of the motion, the tool tip settled $0.3mm$ away from the endpoint.

The group chose to complete the path in 5 seconds, while keeping the cost of the robot components to a absolute minimum. After optimization, the path took a total of 5 seconds to complete. And the total cost ended at 18.75. In comparison, the maximal possible cost was 24 and the minimal was 12.

Appendices

Appendix A: Matlab Code

```
close all;
clear;

%% Coordinates of path points relative to reference coordinate system %%
P1=[1400 -500 1200];
P2=[2400 -500 1200];
P3=[2400 500 1800];
P4=[1400 500 1800];

P5=[2000 -600 1180];
P6=[2000 600 1820];

%%%%%%%%%%%%%%CONSTANTS DATE%%%%%%%%%%%%%%
%%%%%XP1%%%%%%
x0=1400;
x0Dot=0;
x0DotDot=0;

%%%%%YP1%%%%%%
y0=-500;
y0Dot=0;
y0DotDot=0;

%%%%%ZP1%%%%%%
z0=1200;
z0Dot=0;
z0DotDot=0;

%%%%%XP5%%%%%%
xP5=2000;
xP5Dot=500;
xP5DotDot=50;

%%%%%YP5%%%%%%
yP5=-600;
yP5Dot=-300;
yP5DotDot=50;

%%%%%ZP5%%%%%%
zP5=1180;
zP5Dot=-100;
zP5DotDot=10;


%%%%%XP2%%%%%%
xP2=2400;
```

```
xP2Dot=100;
xP2DotDot=50;

%%%%YP2%%%%%
yP2=-500;
yP2Dot=300;
yP2DotDot=50;

%%%%ZP2%%%%%
zP2=1200;
zP2Dot=100;
zP2DotDot=50;

%%%%XP3%%%%%
xP3=2400;
xP3Dot=-100;
xP3DotDot=-10;

%%%%YP3%%%%%
yP3=500;
yP3Dot=200;
yP3DotDot=50;

%%%%ZP3%%%%%
zP3=1800;
zP3Dot=50;
zP3DotDot=20;


%%%%XP6%%%%%
xP6=2000;
xP6Dot=-500;
xP6DotDot=50;

%%%%YP6%%%%%
yP6=600;
yP6Dot=50;
yP6DotDot=50;

%%%%ZP6%%%%%
zP6=1820;
zP6Dot=10;
zP6DotDot=10;

%%%%XP4%%%%%
xP4=1400;
xP4Dot=0;
xP4DotDot=0;

%%%%YP4%%%%%
yP4=500;
```

```
yP4Dot=0;
yP4DotDot=0;

%%%%ZP4%%%%%
zP4=1800;
zP4Dot=0;
zP4DotDot=0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TIME
T0=0;
T5=0.5;
T1=1.5;
T2=2.5;
T6=3.5;
T3=4.3;

t1=[T0:0.1:T5];
t5=[T5:0.1:T1];
t2=[T1:0.1:T2];
t3=[T2:0.1:T6];
t6=[T6:0.1:T3];

Step=0.1;
Counter=1;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X1=[x0
    x0Dot
    x0DotDot
    xP5
    xP5Dot]

X5=[xP5
    xP5Dot
    xP5DotDot
    xP2
    xP2Dot]


X2=[xP2
    xP2Dot
    xP2DotDot
    xP3
    xP3Dot]

X3=[xP3
    xP3Dot
    xP3DotDot
```

```
        xP6
        xP6Dot]

X6=[xP6
        xP6Dot
        xP6DotDot
        xP4
        xP4Dot
        xP4DotDot]


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y1=[y0
        y0Dot
        y0DotDot
        yP5
        yP5Dot]

Y5=[yP5
        yP5Dot
        yP5DotDot
        yP2
        yP2Dot]

Y2=[yP2
        yP2Dot
        yP2DotDot
        yP3
        yP3Dot]

Y3=[yP3
        yP3Dot
        yP3DotDot
        yP6
        yP6Dot]

Y6=[yP6
        yP6Dot
        yP6DotDot
        yP4
        yP4Dot
        yP4DotDot]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z1=[z0
        z0Dot
        z0DotDot
        zP5
        zP5Dot]

Z5=[zP5
```

```
      zP5Dot
      zP5DotDot
      zP2
      zP2Dot]


Z2=[zP2
      zP2Dot
      zP2DotDot
      zP3
      zP3Dot]

Z3=[zP3
      zP3Dot
      zP3DotDot
      zP6
      zP6Dot]

Z6=[zP6
      zP6Dot
      zP6DotDot
      zP4
      zP4Dot
      zP4DotDot]


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FI1=[1   T0   T0.^2   T0.^3     T0.^4
     0   1    2*T0    3*T0.^2   4*T0.^3
     0   0    2       6*T0      12*T0.^2
     1   T5   T5.^2   T5.^3     T5.^4
     0   1    2*T5    3*T5.^2   4*T5.^3]

 FI5=[1   T5   T5.^2   T5.^3     T5.^4
     0   1    2*T5    3*T5.^2   4*T5.^3
     0   0    2       6*T5      12*T5.^2
     1   T1   T1.^2   T1.^3     T1.^4
     0   1    2*T1    3*T1.^2   4*T1.^3]


FI2=[1  T1  T1.^2   T1.^3     T1.^4
     0  1   2*T1    3*T1.^2   4*T1.^3
     0  0   2       6*T1      12*T1.^2
     1  T2  T2.^2   T2.^3     T2.^4
     0  1   2*T2    3*T2.^2   4*T2.^3]


 FI3=[1   T2  T2.^2   T2.^3     T2.^4
       0  1   2*T2    3*T2.^2   4*T2.^3
       0  0   2       6*T2      12*T2.^2
       1  T6  T6.^2   T6.^3     T6.^4
```

```
        0   1    2*T6      3*T6.^2      4*T6.^3]


  FI6=[1   T6   T6.^2     T6.^3        T6.^4        T6.^5
        0   1    2*T6      3*T6.^2      4*T6.^3      5*T6.^4
        0   0    2         6*T2         12*T2.^2     20*T2.^3
        1   T3   T3.^2     T3.^3        T3.^4        T3.^5
        0   1    2*T3      3*T3.^2      4*T3.^3      5*T3.^4
        0   0    2         6*T3         12*T3.^2     20*T3.^3]


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CX1=inv(FI1)*X1
CY1=inv(FI1)*Y1
CZ1=inv(FI1)*Z1

CX5=inv(FI5)*X5
CY5=inv(FI5)*Y5
CZ5=inv(FI5)*Z5

CX2=inv(FI2)*X2
CY2=inv(FI2)*Y2
CZ2=inv(FI2)*Z2

CX3=inv(FI3)*X3
CY3=inv(FI3)*Y3
CZ3=inv(FI3)*Z3

CX6=inv(FI6)*X6
CY6=inv(FI6)*Y6
CZ6=inv(FI6)*Z6

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p1X=CX1(1)+CX1(2)*t1+CX1(3)*t1.^2+CX1(4)*t1.^3+CX1(5)*t1.^4
p1Y=CY1(1)+CY1(2)*t1+CY1(3)*t1.^2+CY1(4)*t1.^3+CY1(5)*t1.^4
p1Z=CZ1(1)+CZ1(2)*t1+CZ1(3)*t1.^2+CZ1(4)*t1.^3+CZ1(5)*t1.^4

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p5X=CX5(1)+CX5(2)*t5+CX5(3)*t5.^2+CX5(4)*t5.^3+CX5(5)*t5.^4
p5Y=CY5(1)+CY5(2)*t5+CY5(3)*t5.^2+CY5(4)*t5.^3+CY5(5)*t5.^4
p5Z=CZ5(1)+CZ5(2)*t5+CZ5(3)*t5.^2+CZ5(4)*t5.^3+CZ5(5)*t5.^4

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p2X=CX2(1)+CX2(2)*t2+CX2(3)*t2.^2+CX2(4)*t2.^3+CX2(5)*t2.^4
p2Y=CY2(1)+CY2(2)*t2+CY2(3)*t2.^2+CY2(4)*t2.^3+CY2(5)*t2.^4
p2Z=CZ2(1)+CZ2(2)*t2+CZ2(3)*t2.^2+CZ2(4)*t2.^3+CZ2(5)*t2.^4

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p3X=CX3(1)+CX3(2)*t3+CX3(3)*t3.^2+CX3(4)*t3.^3+CX3(5)*t3.^4
p3Y=CY3(1)+CY3(2)*t3+CY3(3)*t3.^2+CY3(4)*t3.^3+CY3(5)*t3.^4
p3Z=CZ3(1)+CZ3(2)*t3+CZ3(3)*t3.^2+CZ3(4)*t3.^3+CZ3(5)*t3.^4
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p6X=CX6(1)+CX6(2)*t6+CX6(3)*t6.^2+CX6(4)*t6.^3+CX6(5)*t6.^4+CX6(6)*t6.^5
p6Y=CY6(1)+CY6(2)*t6+CY6(3)*t6.^2+CY6(4)*t6.^3+CY6(5)*t6.^4+CY6(6)*t6.^5
p6Z=CZ6(1)+CZ6(2)*t6+CZ6(3)*t6.^2+CZ6(4)*t6.^3+CZ6(5)*t6.^4+CZ6(6)*t6.^5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TimeResults=[t1 t5 t2 t3 t6];

XResults=[p1X p5X p2X p3X p6X];

YResults=[p1Y p5Y p2Y p3Y p6Y];

ZResults=[p1Z p5Z p2Z p3Z p6Z];

Results=[t1   t5   t2      t3     t6
         p1X  p5X  p2X     p3X    p6X
         p1Y  p5Y  p2Y     p3Y    p6Y
         p1Z  p5Z  p2Z     p3Z    p6Z]';


%Write the result matrix to a csv file
csvwrite('Complete_Path.csv',Results)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%PLOTTING PATH%%%%%%%%%%%%%%%%%
plot3(p1X,p1Y,p1Z);
hold on;
plot3(p5X,p5Y,p5Z);
hold on;
plot3(p2X,p2Y,p2Z);
hold on;
plot3(p3X,p3Y,p3Z);
hold on;
plot3(p6X,p6Y,p6Z);
hold on;

%plot Posisjon
plot3(1400,-500,1200,'marker','o');
text(1400,-500,1300,'P1')
hold on;
plot3(2000,-600,1180,'marker','o');
text(2000,-600,1280,'P5')
hold on;
plot3(2400,-500,1200,'marker','o');
text(2400,-500,1300,'P2')
hold on;
plot3(2400,500,1800,'marker','o');
text(2400,500,1900,'P3')
```

```
hold on;
plot3(2000,600,1820,'marker','o');
text(2000,600,1920,'P6')
hold on;
plot3(1400,500,1800,'marker','o');
text(1400,500,1900,'P4')
grid;
```