THE DEPARTMENT OF AUTOMOTIVE ENGINEERING
CLEMSON UNIVERSITY
AuE 8220: Autonomy: Mobility and Manipulation, Fall 2022
Homework #3: Interpolation, Inverse Kinematics - Solution
Assigned on: September 27th 2022 Due: October 4th 2022, 1:00 PM

Problem 1 (Also in earlier homework)

The relative orientation of two frames of reference is given as follows. Frame A forms the base/reference frame while Frame B is obtained by taking Frame A and rotating it by the following three incremental relative rotations ZYZ: R[z,pi/4] followed by R[y,pi/6] and finally R[z,pi/9].

Assume that we interpolate from Frame A (t=0) to Frame B (t=10) smoothly using:

A) Interpolation of the elements of the relative-orientation matrix (from Frame A initially coincident with the inertial frame to Frame B
B) Interpolation of absolute XYZ Euler Angles representation
C) Interpolation of the elements of the relative-orientation matrix (from Frame A initially coincident with the inertial frame to Frame B using Peter Corke's Robotics Toolbox and compare results obtained from part A
D) Interpolation of absolute XYZ Euler Angles representation using Peter Corke's Robotics Toolbox and compare results obtained from part B

For each of the above cases plot the orientation of the intermediate frames at t= [0:2:10] secs (if possible on the same graph; if not in 4 subplots). You can use Matlab coding for Part A and Part B calculations. Bonus points: Include an animation of the interpolation as an MP4 as well as your code files (which should be setup to run automatically). Use a GUI!!

Solution:

1A,C)
Step 1:

Since the Frame A -> Frame B undergoes successive relative rotations R[z,pi/4] followed by R[y,pi/6] and finally R[z,pi/9], we can obtain the final rotation matrix R as below:

$$R = \begin{bmatrix} \cos(\frac{pi}{4}) & -sin(\frac{pi}{4}) & 0 \\ \sin(\frac{pi}{4}) & cos(\frac{pi}{4}) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\frac{pi}{6}) & 0 & sin(\frac{pi}{6}) \\ 0 & 1 & 0 \\ -sin(\frac{pi}{6}) & 0 & cos(\frac{pi}{6}) \end{bmatrix} * \begin{bmatrix} \cos(\frac{pi}{9}) & -sin(\frac{pi}{9}) & 0 \\ sin(\frac{pi}{9}) & cos(\frac{pi}{9}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The final rotation matrix can be obtained by using matrix multiplication operation using Matlab.

Step 2:
Given the intermediate frames at 0, 2, 4, 6, 8, 10 seconds respectively, each intermediate angle for the rotations about each ZYZ axis can be obtained using linear interpolation. (linspace function Matlab).

Below shown is the code:

```
theta1= linspace(0,pi/4,5);
theta2= linspace(0,pi/6,5);
theta3= linspace(0,pi/9,5);
```

Step 3:
Rotation about each axis can be obtained using:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

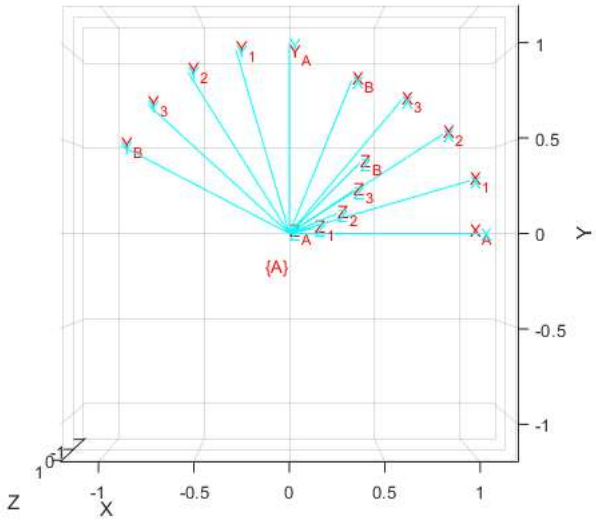
$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4:
Unit vectors [1,0,0], [0,1,0] and [0,0,1] are obtained along x axes, y axes and z axes. The corresponding vector in each intermediate frame is calculated as $p_{x'}=Rp_x$, $p_{y'}=Rp_y$, $p_{z'}=Rp_z$ Where p_x, p_y and p_z *are* the x, y and z unit vectors in the initial frame. The intermediate axes are then plotted on the same plot to demonstrate the change in orientation of frame from A to B.

```
len=length(theta1);
origin=[0;0;0];
x0=[1;0;0];
y0=[0;1;0];
z0=[0;0;1];
xt=zeros(3,len);
yt=zeros(3,len);
```

```
zt=zeros(3,len);
Rt=zeros(3,3);
for i=1:len
      Rt=rotz(theta1(i))*roty(theta2(i))*rotz(theta3(i));
      xt(:,i)=Rt*x0;
      yt(:,i)=Rt*y0;
      zt(:,i)=Rt*z0;
```

Below shown is the output using matlab plot functions:



1B,D) Similar to 1A, for absolute XYZ angles 1st,

For absolute XYZ Euler Angles, first the final rotation matrix about the absolute frame for angles R[x,Φ], R[y,θ] and R[z,ψ] as: $R=R[z,\psi]*R[y,\theta]*R[x,\varphi]$
The final rotation matrix for absolute XYZ Euler angles expand as:

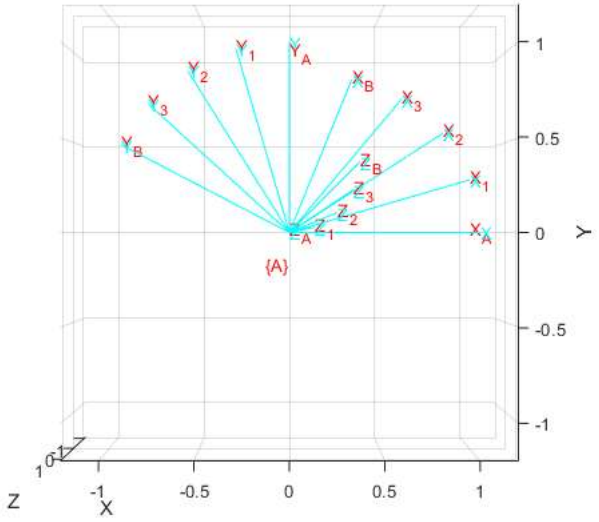
$$R = \begin{bmatrix} cos\psi cos\theta & -sin\psi cos\varphi + cos\psi sin\theta sin\varphi & sin\psi sin\varphi + cos\psi sin\theta cos\varphi \\ sin\psi cos\theta & cos\psi cos\varphi + sin\psi sin\theta sin\varphi & -cos\psi sin\varphi + sin\psi sin\theta cos\varphi \\ -sin\theta & cos\theta sin\varphi & cos\theta cos\varphi \end{bmatrix}$$

θ = asin(R[3,1]) = 0.4891 ψ= atan2 (R[2,1]R[1,1]) = 1.1833 φ = atan2(R[3,2]R[3,3])=0.1950.

The angles ψ, φ and θ are then interpolated for time t=0, t=2.5, t=5, t=7.5 and t=10 seconds and the rotation matrix for each intermediate angle is determined as: $R=R[z,\psi]*R[y,\theta]*R[x,\varphi]$
Unit vectors [1,0,0], [0,1,0] and [0,0,1] are obtained along x axes, y axes and z axes.
The corresponding vector in each intermediate frame is calculated as $px'=Rpx$ $py'=Rpy$ $pz'=Rpz$
Where px, py and pz are the x, y and z unit vectors in the initial frame. The intermediate axes are then plotted on the same plot to demonstrate the change in orientation of frame from A to B.
Code is shown below:

```
theta1=pi/4;
theta2=pi/6;
theta3=pi/9;
Rt=rotz(theta1)*roty(theta2)*rotz(theta3);
tpsi=atan2(Rt(2,1),Rt(1,1))
tphi=atan2(Rt(3,2),Rt(3,3))
ttheta=asin(-Rt(3,1))
psi=linspace(0,tpsi,5);
phi=linspace(0,tphi,5);
theta=linspace(0,ttheta,5);
len=length(psi);
Rt_eu=zeros(3,3);
xt=zeros(3,len);
yt=zeros(3,len);
zt=zeros(3,len);
x0=[1;0;0];
y0=[0;1;0];
z0=[0;0;1];
for i=1:len
Rt_eu=rotz(psi(i))*roty(theta(i))*rotx(phi(i))
xt(:,i)=Rt_eu*x0;
yt(:,i)=Rt_eu*y0;
zt(:,i)=Rt_eu*z0;
```

Output is shown below:



Note: The above codes use the Peter Corke's Robotics Toolbox function rotz and roty. To perform operations manually, code the rotation matrices by hand and then perform the matrix multiplication operations instead of using the PTRTB tollbox function directly which should be used for Part C and D only.

Problem 2

Given that the manipulator structural parameters are a_1= 2, a_2= 1 and that this manipulator is required to move between position $P_1 = (x_1, y_1,) = (-1, 1)$ and $P_2 = (x_2, y_2,) = (2, 1)$ in 10 seconds. However, its motion between these initial and final positions can be achieved in a variety of ways and you will explore some of these during this homework problem. In particular, you will try the following schemes:

A) Assuming that the manipulator starts and stops from rest – fit a cubic polynomial between the start and finish positions in each of the Cartesian coordinates.
B) Assuming that the manipulator starts and stops from rest – fit a quintic polynomial between the start and finish configurations in terms of the joint coordinates.
C) Assuming that the manipulator starts and stops from rest – fit a cubic polynomial between the start and finish positions in each of the Cartesian coordinates using Peter Corke's Robotics Toolbox and compare the results from part A
D) Assuming that the manipulator starts and stops from rest – fit a quintic polynomial between the start and finish positions in each of the joint coordinates using Peter Corke's Robotics Toolbox and compare the results from part B

In separate graphs, plot (i) X vs t for all interpolation schemes; (ii) Y vs t for all interpolation schemes; and (iii) Y vs X for all interpolation schemes. Similarly in separate graphs plot the various interpolated joint trajectories (iv) θ_1 vs t for all interpolation schemes; (v) θ_2 vs t for all interpolation schemes; Submit a total of 5 graphs. For plotting some of the results, make sure that you choose an adequate resolution for the problem.

You can use Matlab coding for Part A and Part B calculations. Bonus points: Include an animation of the interpolation as an MP4 as well as your code files (which should be setup to run automatically). Use a GUI!!

Solution:

Step1: The inverse kinematics of the mechanism was determined, by calculating the joint angles $\theta 1$ and $\theta 2$ of the mechanism, when the end effector is at (-1,1) as well as when the end effector is at (2,1). At (-1,1), $\theta 1 = 2.8429$ and $\theta 2 = -2.1489$. At (2,1), $\theta 1 = 0.9723$ and $\theta 2 = -1.5708$ in radians
At (-1,1), $\theta 1 = 162.8856$ and $\theta 2 = -138.5904$. At (2,1), $\theta 1 = 53.1301$ and $\theta 2 - 90$ in degree.

Step2: The joint angles $\theta 1$ and $\theta 2$ were interpolated smoothly for 10 intervals between time t=0 and t=10 using linspace() function of MATLAB

Step 3: The forward kinematics of the end effector at each time 't', for the corresponding values of $\theta 1$ and $\theta 2$, was then determined to obtain the trajectory of the robot in cartesian space.

Step 4: Cartesian coordinates X and Y were plotted against time 't'. Cartesian coordinates X was plotted against Y. Joint variables $\theta 1$ and $\theta 2$ were plotted against time 't'.

Step 5:
For a cubic polynomial:
$S(t) = At^3 + Bt^2 + Ct + d$
On differentiating w.r.t t
$\dot{S}(t) = 3At^2 + 2Bt + C$

For a quintic polynomial
$S(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$

In order to find out the coefficients:

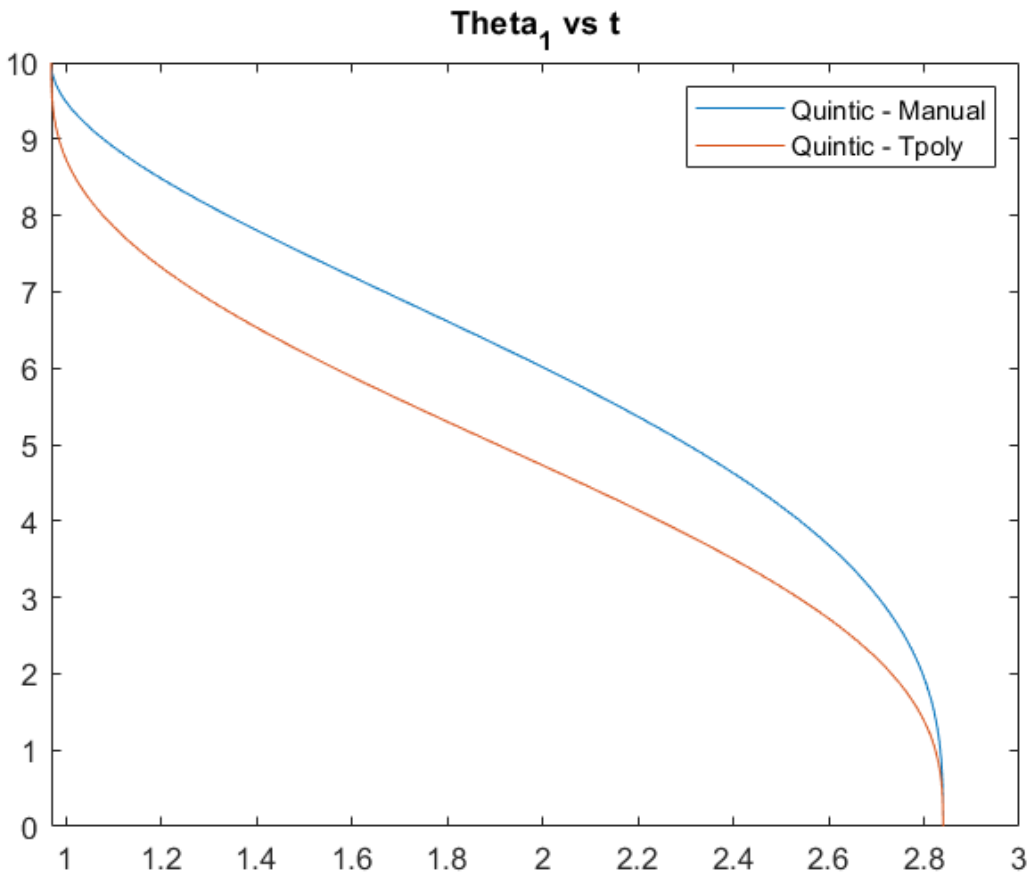
Cubic polynomial:
We no the start and end positions in X, Y, theta1 and theta2 (Given in the question)
We know that that start and velocity in both cartesian and joint space is 0
Hence, we have four equations and coefficient s can be obtained by using Matlab operations:

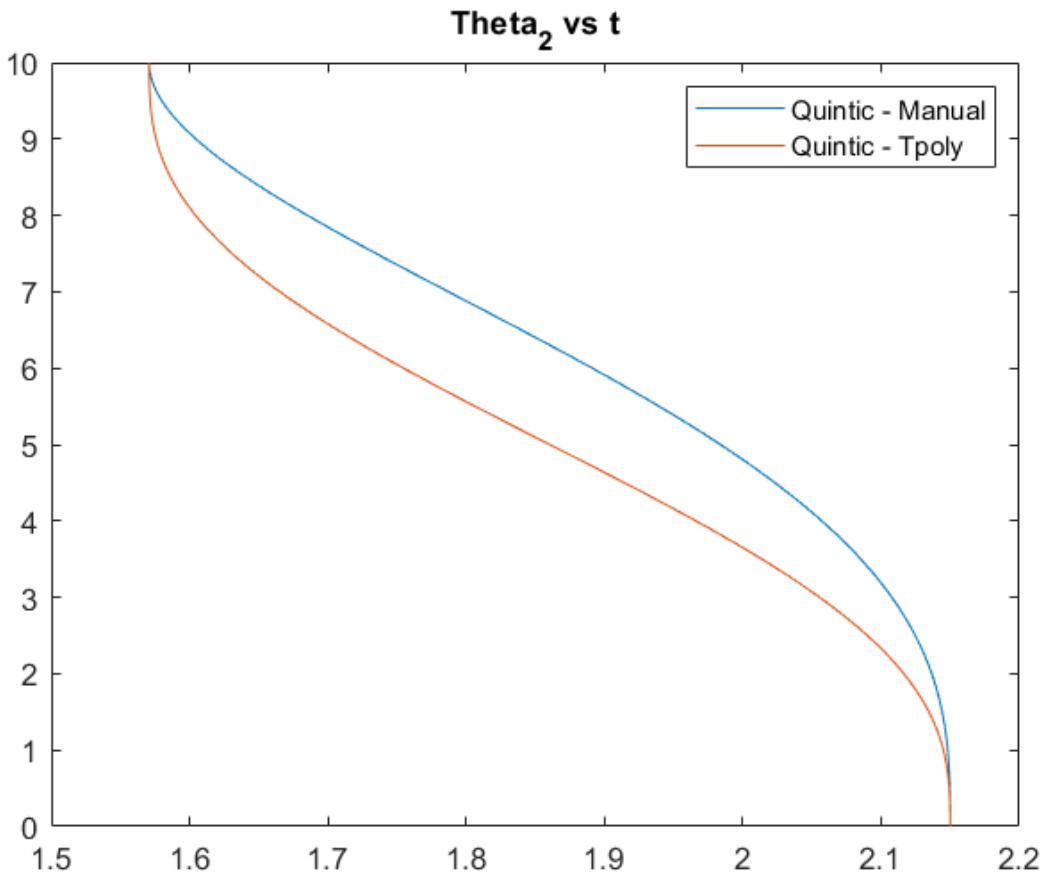
Similarly for a quintic polynomial:
We differentiate upto the acceleration level:
$\dot{S}(t) = 5At^4 + 4Bt^3 + 3Ct^2 + 2Dt + E$
$\ddot{S}(t) = 20At^3 + 12Bt^2 + 6Ct + 2D$

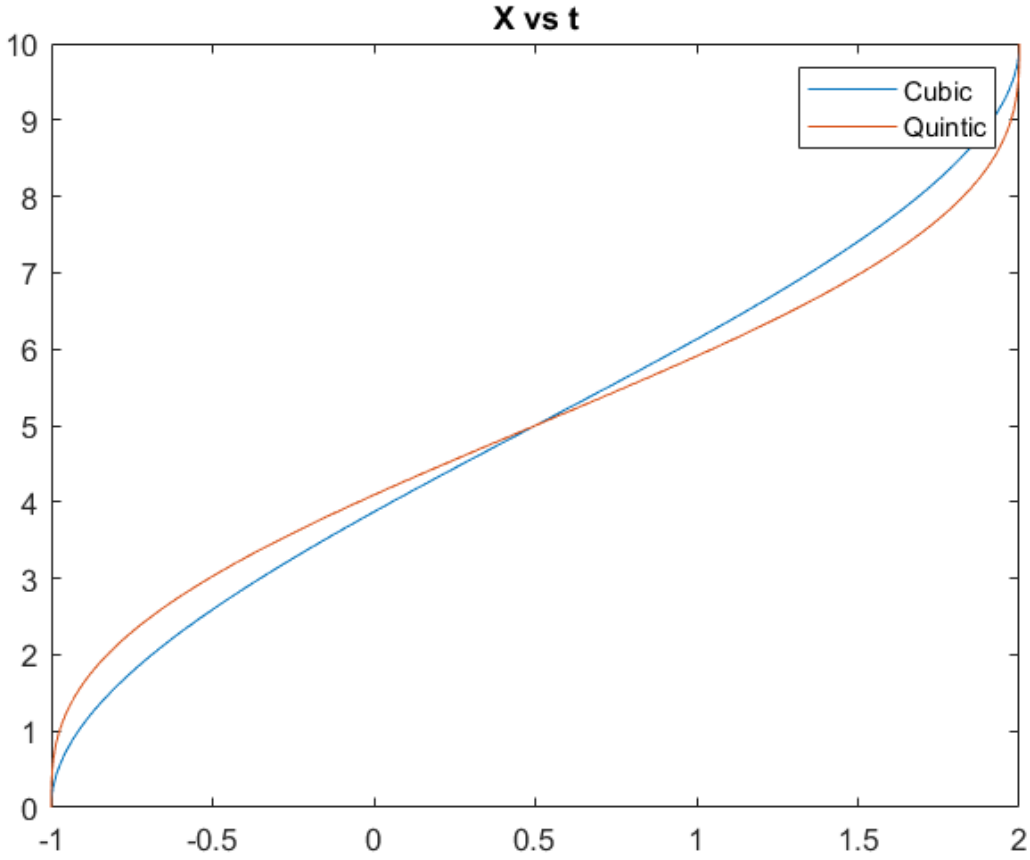
We know the start and end velocities as zero and the same with acceleration. Hence, we get six equations and six unknowns which can be solved.

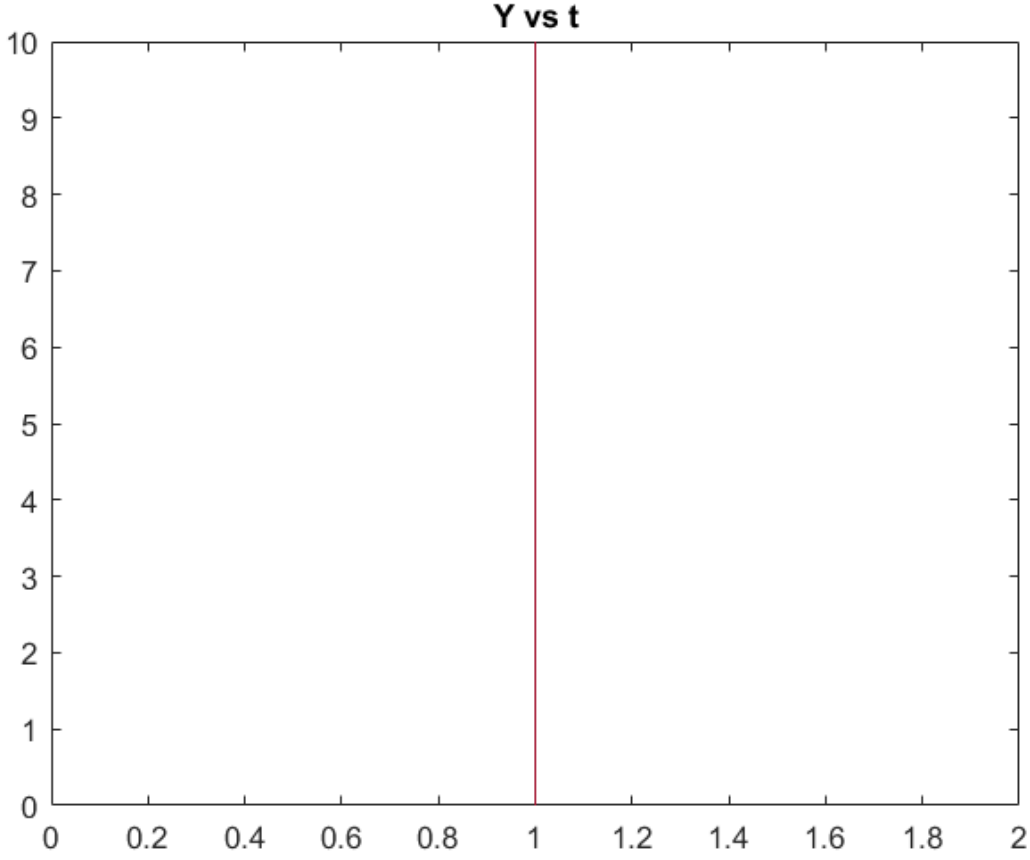
$$\begin{pmatrix} s_0 \\ s_T \\ \dot{s}_0 \\ \dot{s}_T \\ \ddot{s}_0 \\ \ddot{s}_T \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix}$$

Since the matrix is square we can solve for the coefficient vector (A, B, C, D, E, F) using standard linear algebra methods such as the MATLAB® \-operator. For a quintic polynomial acceleration will be a smooth cubic polynomial, and jerk will be a parabola.

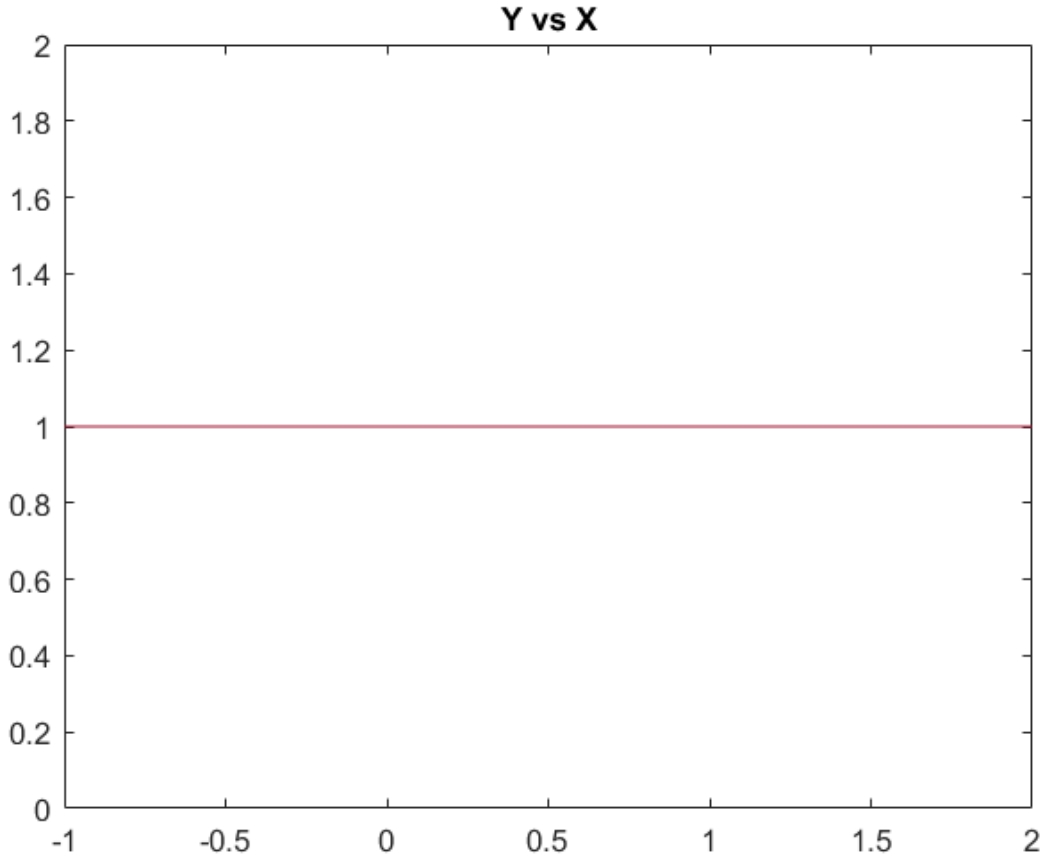

Theta1 vs t
Quintic - Manual
Quintic - Tpoly
10
9
8
7
6
5
4
3
2
1
0
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3


Theta2 vs t
Quintic - Manual
Quintic - Tpoly
10
9
8
7
6
5
4
3
2
1
0
1.5
1.6
1.7
1.8
1.9
2
2.1
2.2


X vs t
Cubic
Quintic
10
9
8
7
6
5
4
3
2
1
0
-1
-0.5
0
0.5
1
1.5
2


Y vs t
10
9
8
7
6
5
4
3
2
1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

MaM



Matlab Code:

```
P = [0 0 0 1
     1000 100 10 1
     0 0 1 0
     300 20 1 0];
B = [2.15;1.5708;0;0];
A = inv(P)*B;

P2 = [0 0 0 0 0 1
      10^5 10^4 10^3 10^2 10 1
      0 0 0 0 1 0
      5*10^4 4*10^3 3*10^2 2*10 1 0
      0 0 0 2 0 0
      20*10^3 2*10^2 6*10 2 0 0];
B2 = [2.84;0.97;0;0;0;0];
B3 = [2.15;1.5708;0;0;0;0];
A2 = inv(P2)*B2;
A3 = inv(P2)*B3;
t = 0:0.01:10;
y = ones(length(t));
a1 = 2;
a2 = 1;
```

```
for i = 1:length(t)
    X(i) = -0.0060*t(i)^3 + 0.09*t(i)^2 -1;
    theta2_2a(i) = acos((X(i)^2+y(i)^2-a1^2-a2^2)/(2*a1*a2));
    theta1_2a(i) = atan2(y(i),X(i)) -
atan2((a2*sin(theta2_2a(i))),(a1+a2*cos(theta2_2a(i))));
    t1(i) = A2(1)*t(i)^5 + A2(2)*t(i)^4 + A2(3)*t(i)^3 + A2(4)*t(i)^2 + A2(5)*t(i) +
A2(6);
    t2(i) = A3(1)*t(i)^5 + A3(2)*t(i)^4 + A3(3)*t(i)^3 + A3(4)*t(i)^2 + A3(5)*t(i) +
A3(6);
    X2(i) = a1*cos(t1(i)) + a2*cos(t1(i) + t2(i));
end
s = tpoly(-1,2,1001);
s2 = tpoly(2.84,0.97,1001);
s3 = tpoly(2.15,1.5708,1001);
figure
plot(t,X);
hold on
plot(t,theta1_2a);
hold on
plot(t, theta2_2a);
title('X and Theta vs t - Cubic');
legend('X', 'theta1', 'theta2');
figure
plot(y,t);
title('Y vs t');
figure
plot(X,y);
title('Y vs X');
figure
plot(t1,t);
hold on
plot(s2,t);
title('Theta_1 vs t')
legend('Quintic - Manual', 'Quintic - Tpoly')
figure
plot(t2,t);
hold on
plot(s3,t);
title('Theta_2 vs t');
legend('Quintic - Manual', 'Quintic - Tpoly');
```