Introduction

One of the most basic kinematic chains is the four bar mechanism and it's inversions.This report introduces a program that can do kinematic analysis of some of the basic inversions of the four bar mechanism like Slider Crank and Crank Rocker with the use of constraint equations.This program will be able to generate constraint equations(C(q,t)=0),Ct which is the first derivative of C(q,t) with respect to time explicitly,Qd which appears in the equation of second derivative of C(q,t) with respect to time,Jacobian(Cq) and also finding position,velocity and acceleration of the links.Though this code is only capable of making equations for Ground joint,Revolute Joint and a simple Prismatic joint.After obtaining the constraint equations for the system the program uses Newton-Raphson algorithm to solve a system of non-linear equations.The kinematic chain is simulated over time where the input (driving constraint) is the rotation of the crank.The simulated answers are then compared to answers found from solving loop-closure equations.

Constrained Kinematics

One can describe the motion of different links in a kinematic chain with the set of algebraic equations called constraint equations.This set of algebraic equations describes the joints between those links.Each constraint equation removes or constraints the motion of links and removes a degree of freedom in certain direction.This program will be dealing with planar mechanism only, so the maximum degree of freedom one body can have is 3.If there are n bodies in the system,the maximum number of freedom the system can have is 3*n.Each constraint equation removes one degree of freedom from the system.If the number of constraint equation(n_c) is equal to 3*n then the system is called kinematic system.Usually there are two types of constraint equations

1. Joint Constraint
2. Driving Constraint

Joint Constraint are constraint equation which describes the joints between the links and the motion constraint because of those joints.While the Driving Constraint will describe the motion which drives the kinematic chain.Joint Constraint and Driving Constraint will together make the system kinematic.This program deals with three kinds of joints

i. Ground
ii. Revolute
iii. Simple Prismatic

Ground

Ground is a body who's degree of freedom is zero.Essentially it's a fixed link.The constraint equation for ground is,

$$q - c = 0 \qquad (1)$$

Where, $q=[R_x^i ; R_y^i ; \theta^i]$ and $c=[c_1 ; c_2 ; c_3]$.

Revolute

When two bodies are connected with a revolute joint,only relative rotation between the two is allowed.There can not be any relative translation between the two.Hence,it generates two constrained equation and gives only one degree of freedom.

If the bodies i and j are connected with revolute joint at point p,

$$R^j + A^j * U^j_p - R^i - A^i * U^i_p = 0 \qquad (2)$$

Where, $R^j = [R_x^j ; R_y^j]$ and U^j_p is the coordinates of point p on body j in local coordinate system.A^j is a transformation matrix for body j.

Simple Prismatic

Simple Prismatic joint only allows relative translation in one direction while constraining translation in one direction and relative rotation.It allows only one degree of freedom.

In this case,we will be using the following equations,

$$\begin{aligned} R_y^i - c_1 &= 0 \\ \Theta^i - c_2 &= 0 \end{aligned} \qquad (3)$$

We can form the constraint equations C (q,t) = 0 with above mentioned equations where, $q = [R^n \theta^n]$ (n=1,2,3..number of bodies) and t indicates time.

Velocity Analysis

Velocity Analysis is done to find the translation and angular velocities of the bodies in the kinematic chain.Differentiating the vector of constraint equation we get,

$$Cq*(dq/dt) = -Ct \qquad (4)$$

Where, Cq is a jacobian matrix and Ct is the matrix we get while differentiating C (q,t) partially with respect to time. The vector Ct is essentially zero if C(q,t) does not contain a equation which is explicitly a function of time.

Acceleration Analysis

We can differentiate the equation 4 again with respect to time to get the equation,

$$Cq*(d^2q/dt^2) = Qd \qquad (5)$$

The equation 5 is a much simplified version,and is useful for the problem we are determined to solve.

Newton Raphson Algorithm

Newton Raphson method is used to solve the system of non-linear equations.The constraint equations can be non-linear and hence we will be using this method to solve the constraint equation.This method can only be used if the system is kinematically driven.

$$Cqi*dqi = -C(qi,t) \qquad (6)$$

$$q_{i+1} = q_i + dq_i \qquad (7)$$

Here,we assume the time is constant and start with a initial guess.Each time we can update the initial guess as shown in equation 7 and can work until,

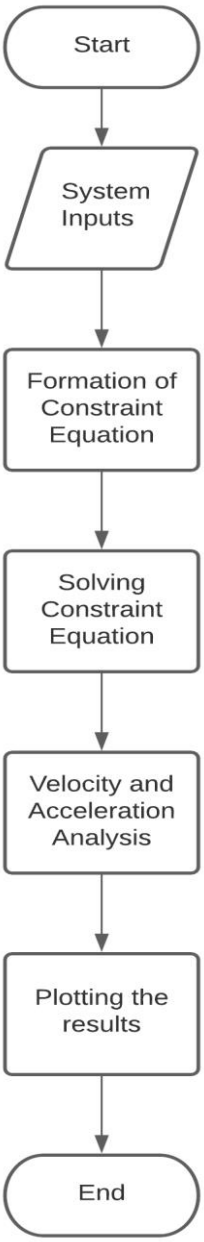
$$| dq_k | < e \quad \text{or} \quad | C (q_k,t) | < e \qquad (8)$$

For this project the value of e is 5*10^-8.

This method to solve non-linear equation does not always work,sometimes the Newton-Raphson fails to converge and the condition 8 doesn't get satisfied.We will talk more about this in the later section.

Computer Implementation

For this project we will be using MATLAB as a means to implement kinematic analysis in the computer.The general flow of the process is shown below

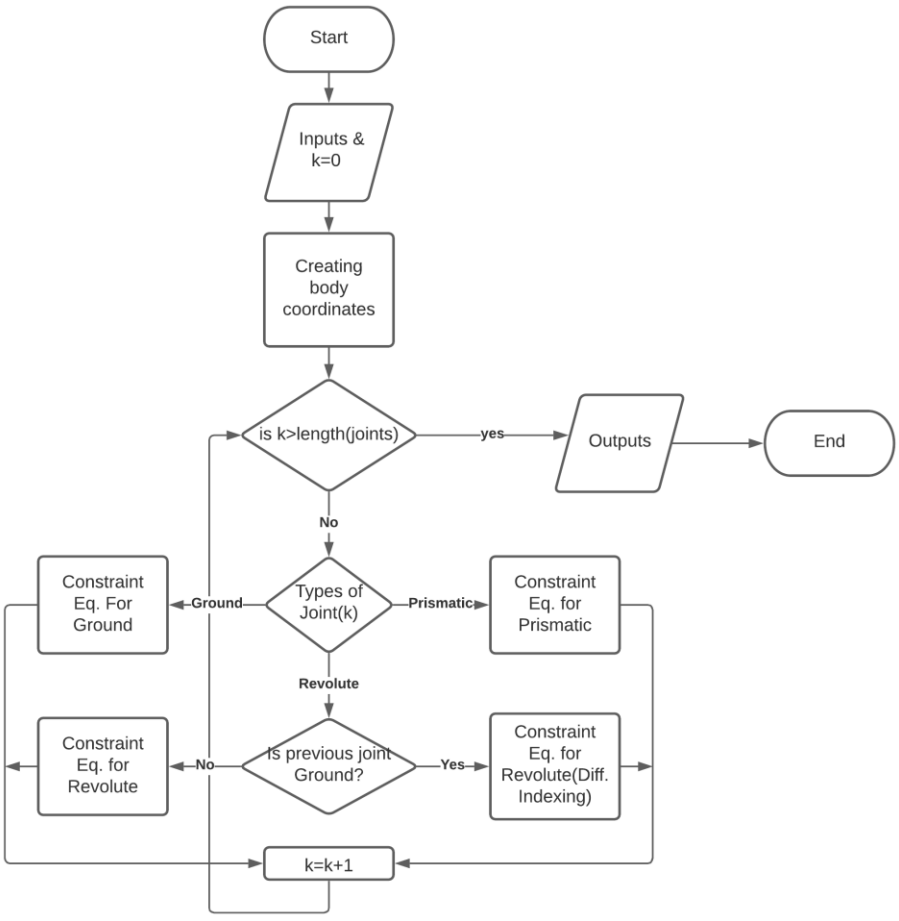


Here the system input contains,

- Type of Kinematic Chain
- Link Lengths
- Number of bodies
- Constants for ground and prismatic joint
- Number of input body (driving link)
- Initial Angle of driving link at t=0
- Angular velocity of driving link
- Simulation time
- Number of points between 0-1 s to solve for
- Initial guess

Making Constraint Equations

The program for making constraint equation follows the below shown flow,



There are several inputs for this function which can be seen in the appendix of this report.The output for this function are Constraint Equations,Jacobian and body coordinates vector.

After we have gained the constraint equation the next function the program calls to is for solving the constraint equation for different time given by the user as an input by using Newton-Raphson method and after solving the equation it goes on to find velocity and acceleration of different points at that point in time.There is also a function to find out the vector Qd for the given system.It separates the terms with d^2q/dt^2 in the second derivative of constraint equation with respect to time and gives the output.The script to those programs can be found in the appendix.

The function to find solution of constraint equation and doing velocity and acceleration analysis is made in a way to avoid singularities,and if it is not avoidable NaN (Not a Number) is assigned to those velocities and acceleration.

Slider Crank

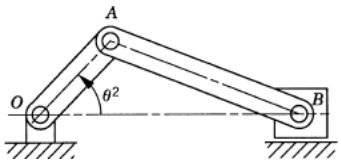


Figure 1. Slider Crank

Figure 1 shows a slider crank mechanism where,

OA = 0.2m

AB = 0.4m

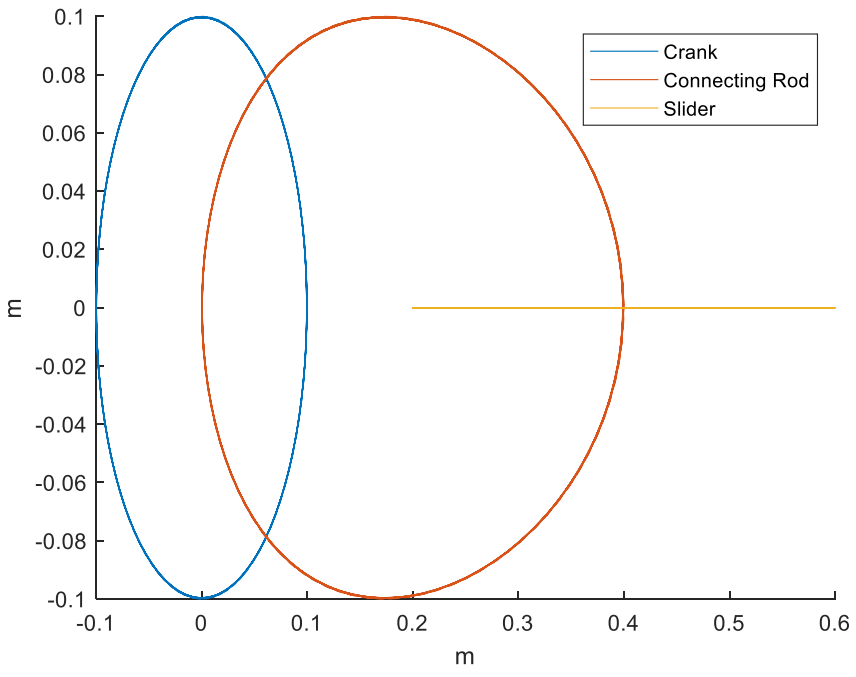
The initial angle at t=0, $\theta^2_0 = 30$ degrees. The angular velocity of link 2 is 30 rad/s.

Along with these inputs,the simulation for this mechanism will be from 0 to 2 seconds. The program will solve for 150 points between 0 - 1 second so in total there will be 300 points at which the program has to find the solution.The outputs for this simulation are shown in the next page.

ConstraintEquation =

$$\begin{pmatrix} r_{x1} \\ r_{y1} \\ \Theta_1 \\ r_{x2} - \frac{\cos(\Theta_2)}{10} \\ r_{y2} - \frac{\sin(\Theta_2)}{10} \\ r_{x3} - r_{x2} - \frac{\cos(\Theta_2)}{10} - \frac{\cos(\Theta_3)}{5} \\ r_{y3} - r_{y2} - \frac{\sin(\Theta_2)}{10} - \frac{\sin(\Theta_3)}{5} \\ r_{x4} - r_{x3} - \frac{\cos(\Theta_3)}{5} \\ r_{y4} - r_{y3} - \frac{\sin(\Theta_3)}{5} \\ r_{y4} \\ \Theta_4 \\ \Theta_2 - 30\,t - \frac{\pi}{6} \end{pmatrix}$$

Below shown plot is the traces of paths for Crank,Connecting Rod and Slider during the simulation



The jacobian

Cq =

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{\sin(\Theta_2)}{10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \sigma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & \frac{\sin(\Theta_2)}{10} & 1 & 0 & \frac{\sin(\Theta_3)}{5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & \sigma_1 & 0 & 1 & \sigma_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & \frac{\sin(\Theta_3)}{5} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \sigma_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = -\frac{\cos(\Theta_2)}{10}$$

$$\sigma_2 = -\frac{\cos(\Theta_3)}{5}$$

Ct =

0
0
0
0
0
0
0
0
0
-30

Qd =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ -\sigma_3 \\ -\sigma_1 \\ -\sigma_3 - \sigma_4 \\ -\sigma_1 - \sigma_2 \\ -\sigma_4 \\ -\sigma_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

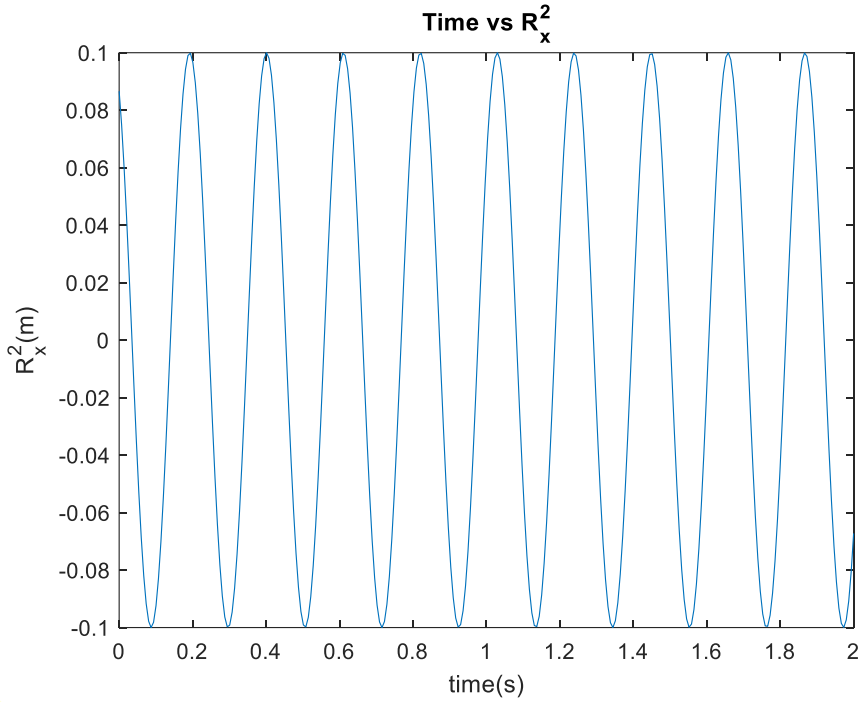
where

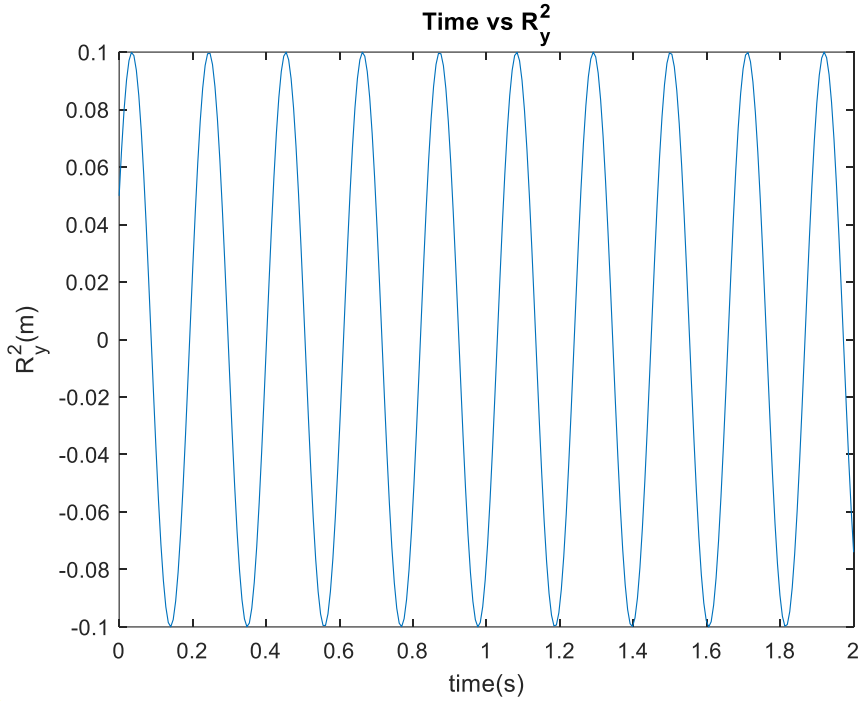
$$\sigma_1 = \frac{\sin(\theta_2(t)) \left(\frac{\partial}{\partial t}\, \theta_2(t)\right)^2}{10}$$

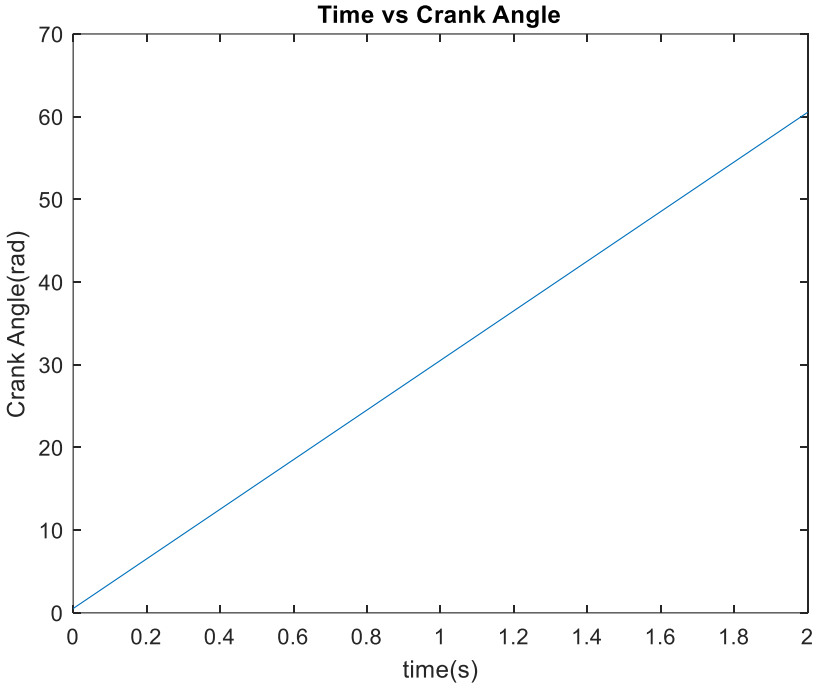
$$\sigma_2 = \frac{\sin(\theta_3(t)) \left(\frac{\partial}{\partial t}\, \theta_3(t)\right)^2}{5}$$

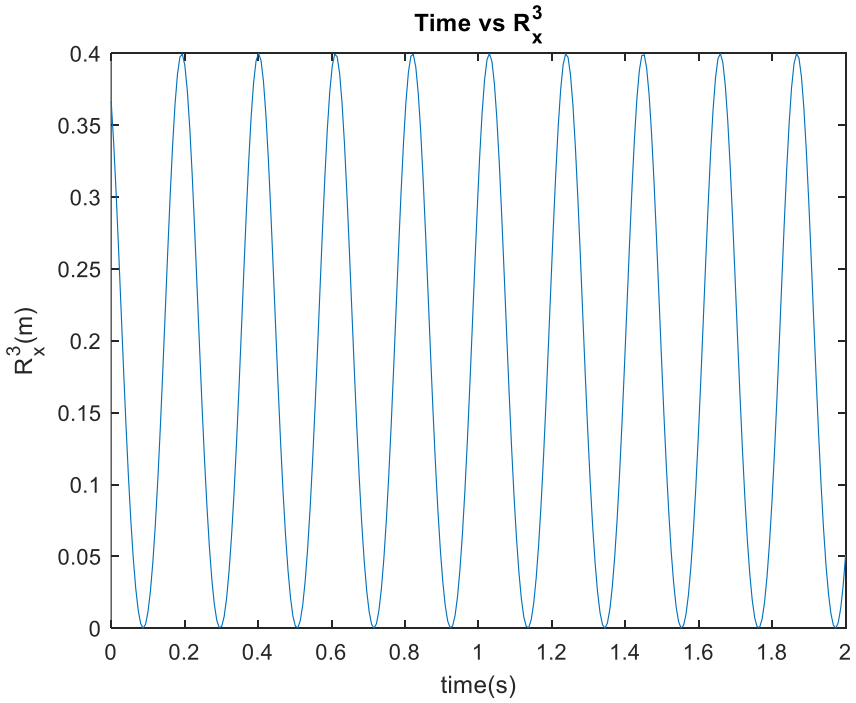
$$\sigma_3 = \frac{\cos(\theta_2(t)) \left(\frac{\partial}{\partial t}\, \theta_2(t)\right)^2}{10}$$

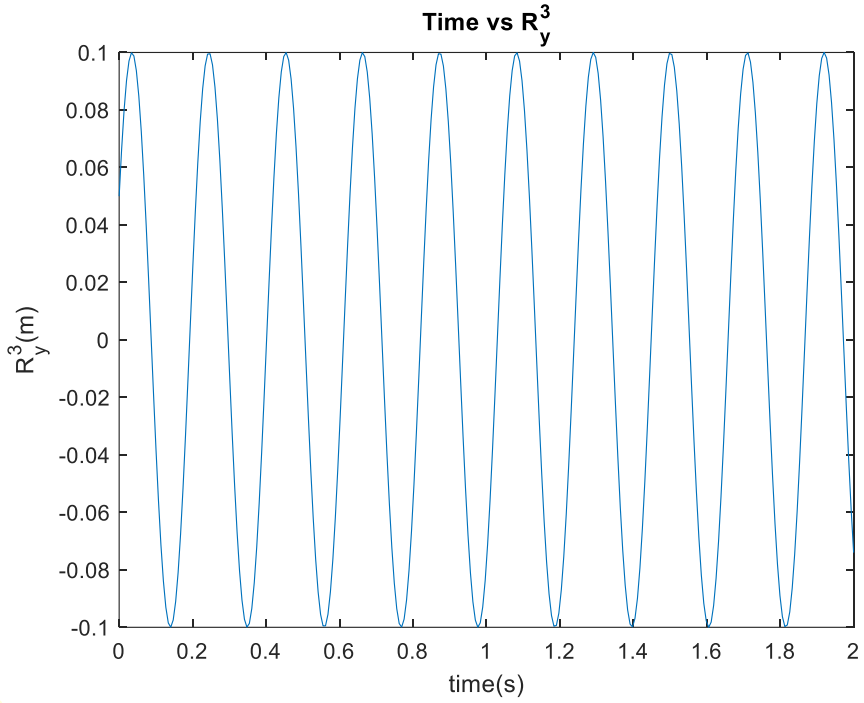
$$\sigma_4 = \frac{\cos(\theta_3(t)) \left(\frac{\partial}{\partial t}\, \theta_3(t)\right)^2}{5}$$

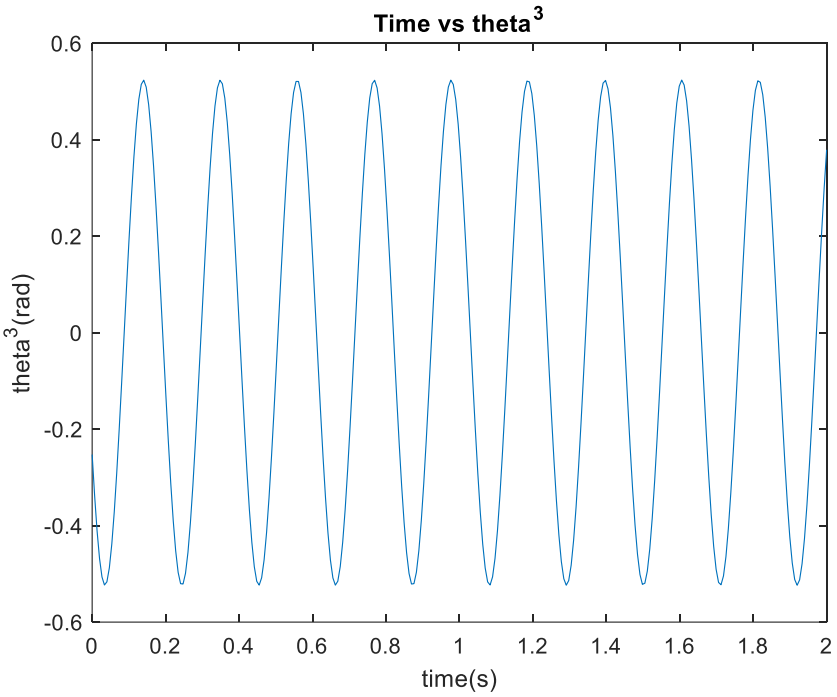

Time vs R_x^2
R_x^2(m)
time(s)
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

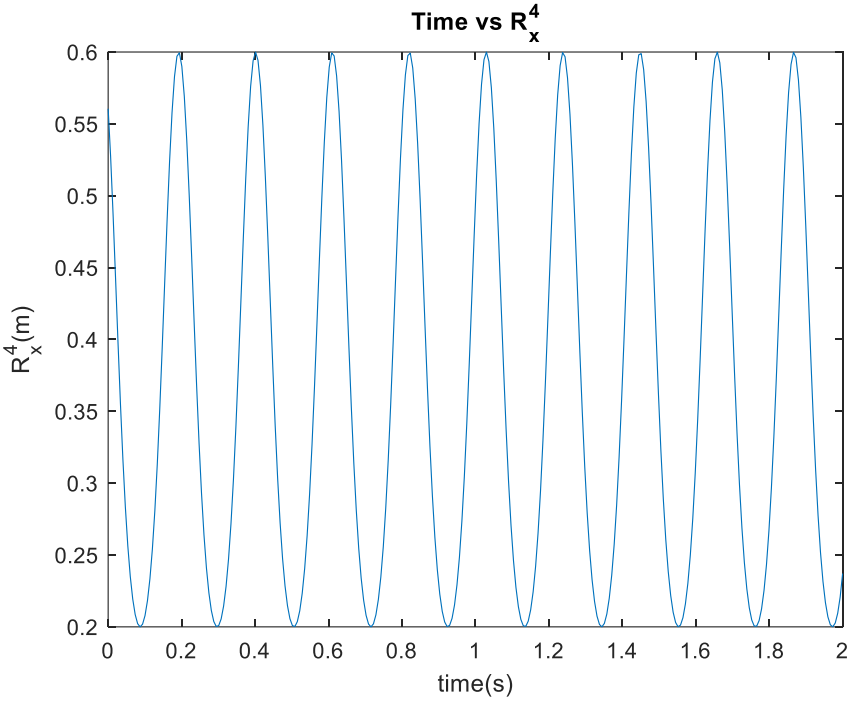

Time vs R_y^2
R_y^2(m)
time(s)
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

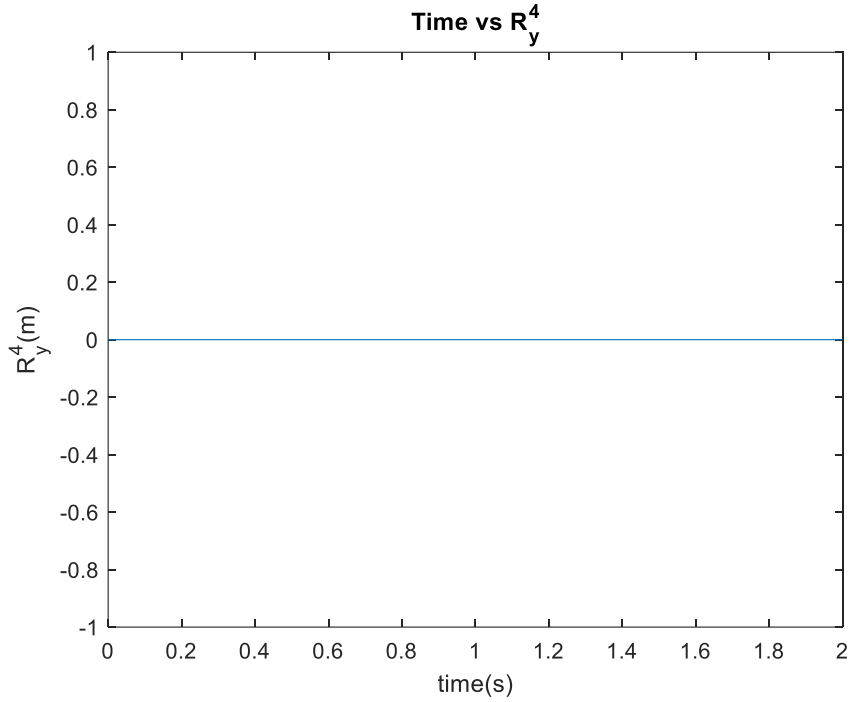

Time vs Crank Angle
Crank Angle(rad)
time(s)
0
10
20
30
40
50
60
70
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

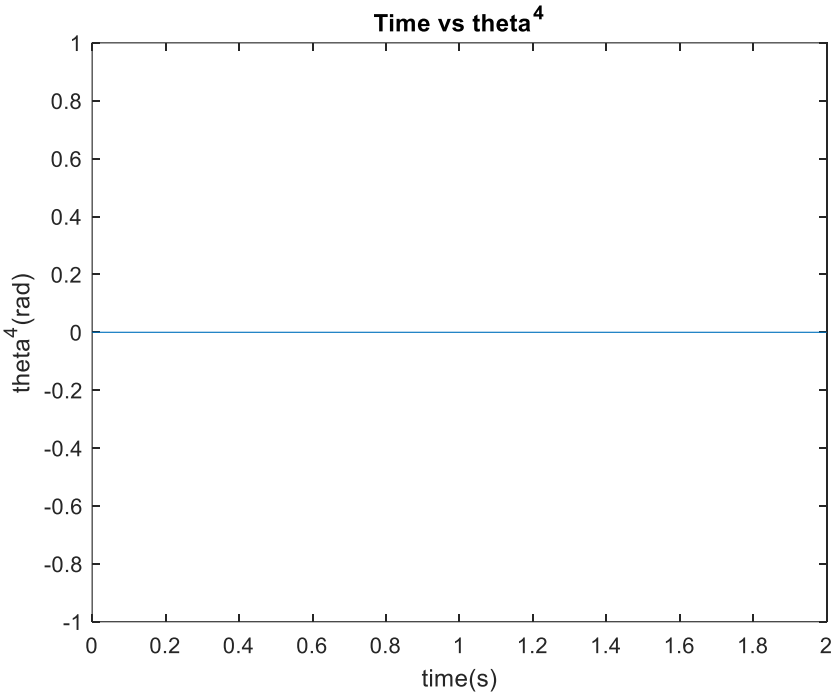

Time vs R_x^3
R_x^3(m)
time(s)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Time vs R_y^3
R_y^3(m)
time(s)
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

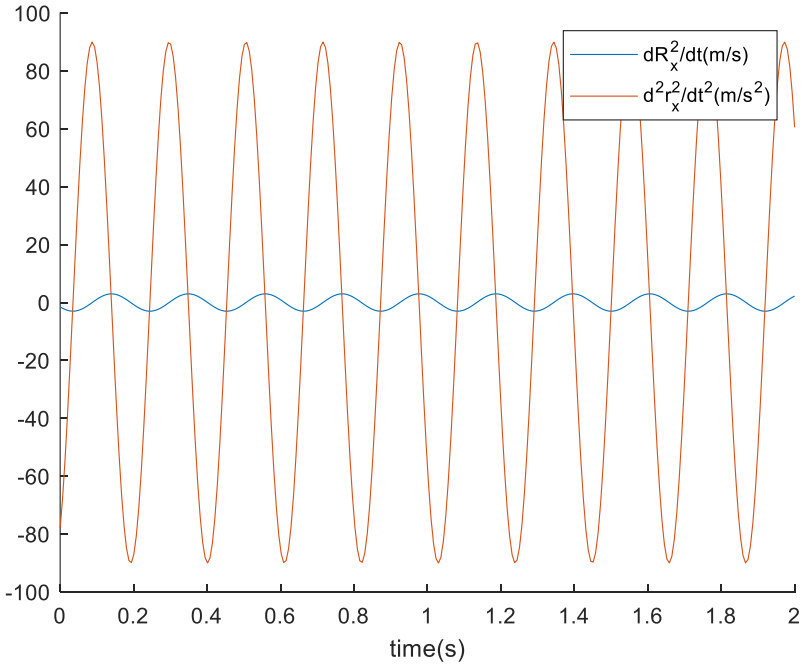

Time vs theta^3
theta^3(rad)
time(s)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

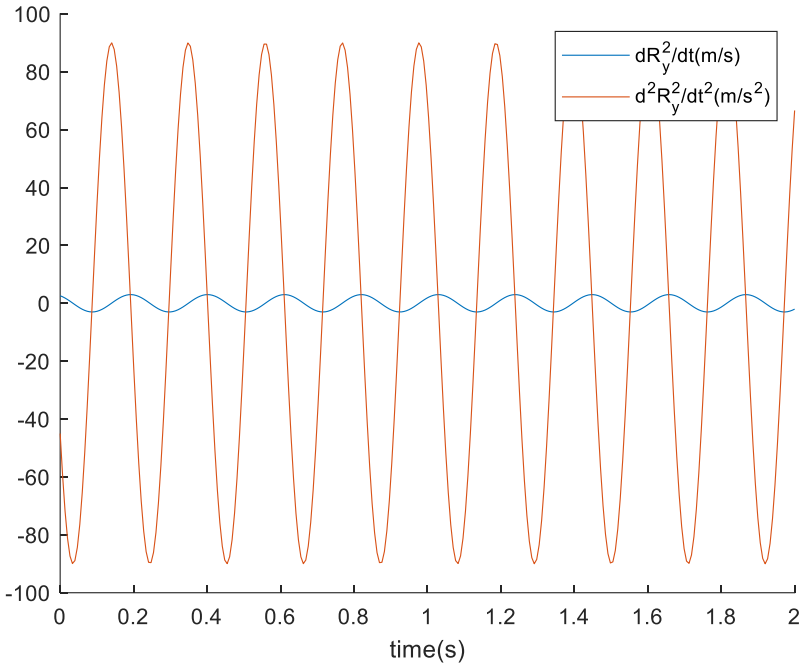

Time vs R_x^4
R_x^4(m)
time(s)
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

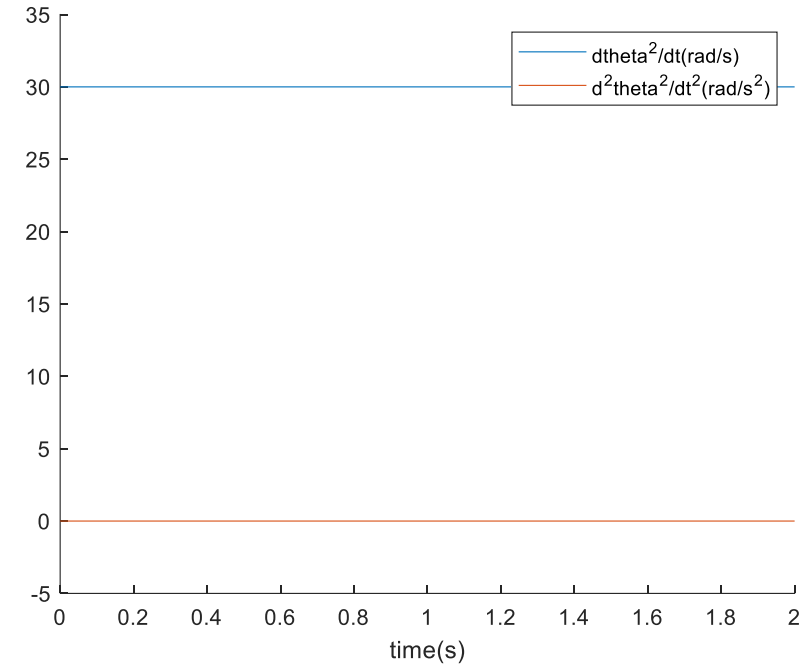

Time vs R_y^4
R_y^4(m)
time(s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

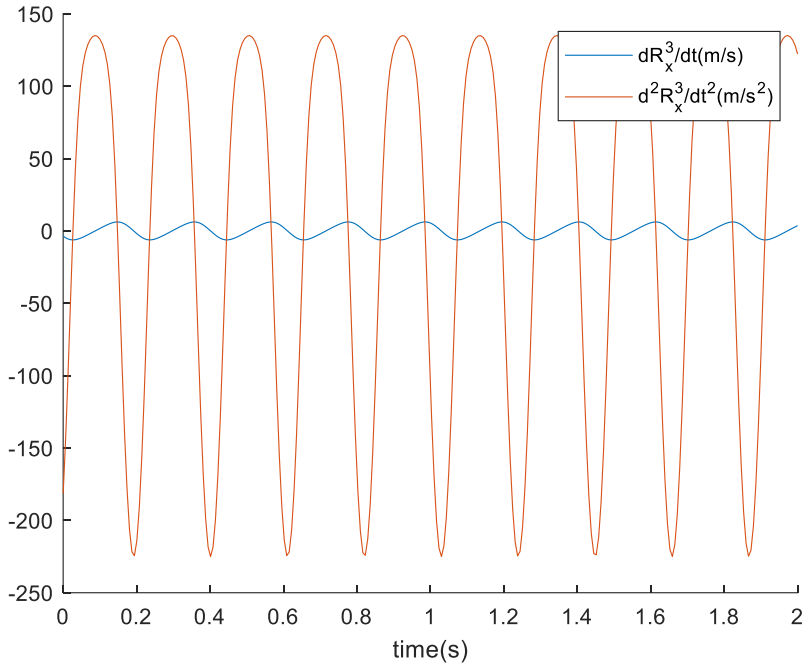


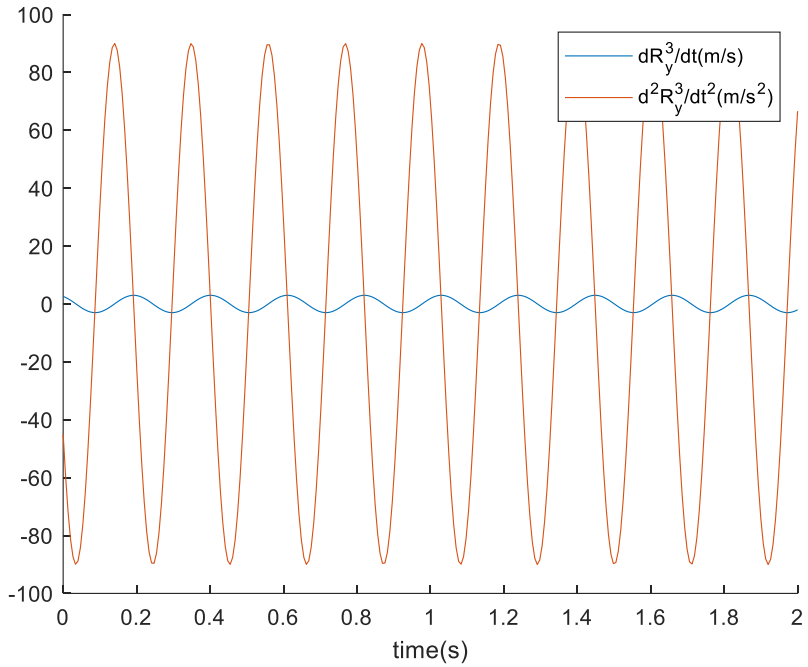
The above ones were all position plots vs time.Now the plots will show the velocity and acceleration vs time

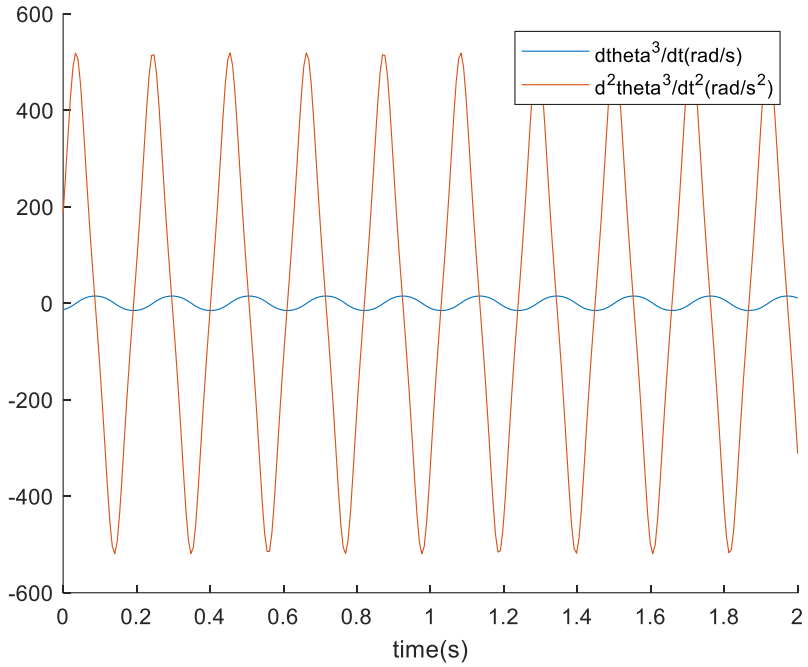


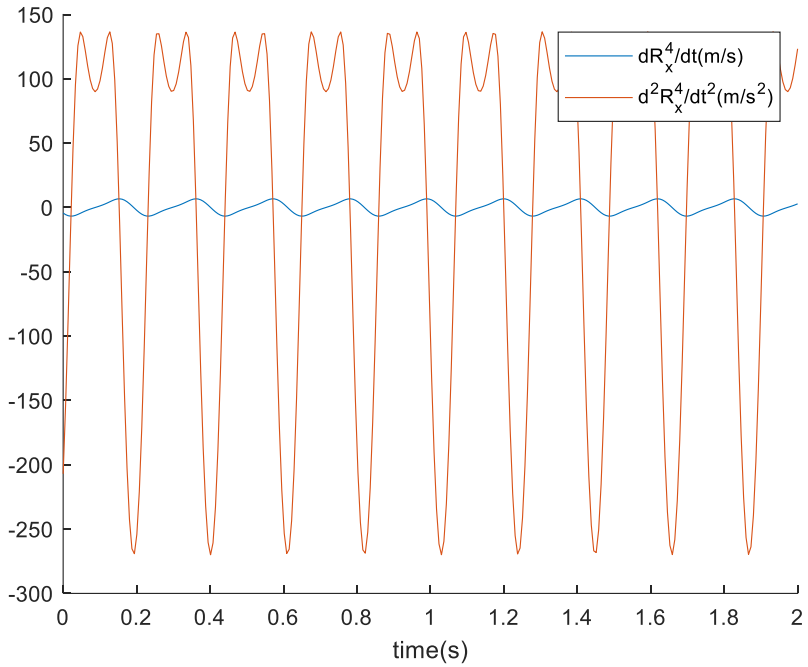

dR$_y^2$/dt(m/s)
d^{2}R$_y^2$/dt^2(m/s^2)
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


dtheta^2/dt(rad/s)
d^{2}theta^2/dt^2(rad/s^2)
35
30
25
20
15
10
5
0
-5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


150
100
50
0
-50
-100
-150
-200
-250
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)
dR_x^3/dt(m/s)
d^2R_x^3/dt^2(m/s^2)


100
80
60
40
20
0
-20
-40
-60
-80
-100
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)
dR_y^3/dt(m/s)
d^2R_y^3/dt^2(m/s^2)


dtheta3/dt(rad/s)
d2theta3/dt2(rad/s2)
600
400
200
0
-200
-400
-600
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


dR4x/dt(m/s)
d2R4x/dt2(m/s2)
150
100
50
0
-50
-100
-150
-200
-250
-300
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

The Table Below compares position,velocity and acceleration solved with different methods

1. Newton - Raphson (NR)
2. Loop - Closure Equations (LC)

At time t=0s

Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.0866	0.0866	-1.5000	-1.5000	-77.9423	-77.9423
0.0500	0.0500	2.5981	2.5981	-45.0000	-45.0000
0.5236	0.5236	30.0000	30.0000	-0.0000	0
0.3669	0.3669	-3.6708	-3.6708	-181.4463	-181.4463
0.0500	0.0500	2.5981	2.5981	-45.0000	-45.0000
-0.2527	-0.2527	-13.4164	-13.4164	185.9032	185.9032
0.5605	0.5605	-4.3416	-4.3416	-207.0080	-207.0080
0	0	-0.0000	0	-0.0000	0
0	0	0	0	0	0

At time = 1s

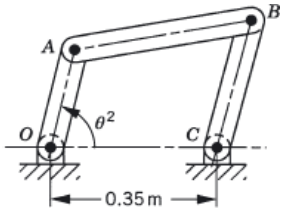
Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.0702	0.0628	2.1353	2.3356	-63.2185	56.4841
-0.0712	-0.0779	2.1073	1.8828	64.0579	70.0681
30.6239	30.5236	30.0000	30.0000	0.0000	0
0.3274	0.3097	5.0730	5.4669	-129.2473	-106.0365
-0.0712	-0.0779	2.1073	1.8828	64.0579	70.0681
0.3639	0.3998	-11.2745	-10.2201	-294.3207	-336.1990
0.5143	0.4940	5.8755	6.2626	-132.0575	-99.1048
0	0	-0.0000	0	0.0000	0
0	0	0	0	0	0

At time = 2s

Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
-0.0672	-0.0672	2.2205	2.2205	60.5168	60.5168
-0.0740	-0.0740	-2.0172	-2.0172	66.6162	66.6162
60.5236	60.5236	30.0000	30.0000	-0.0000	0
0.0513	0.0513	3.6375	3.6375	122.1950	122.1950
-0.0740	-0.0740	-2.0172	-2.0172	66.6162	66.6162
0.3791	0.3791	10.8570	10.8570	-311.5803	-311.5803
0.2371	0.2371	2.8339	2.8339	123.3565	123.3565
0	0	0.0000	0	-0.0000	0
0	0	0	0	0	0

As you can see from the tables above the answers are nearly the same for every position ,velocity and acceleration except at time t=1 s,the reason behind that is the answers for newton raphson are for time t=1.0033s instead of 1.0000s.

Crank Rocker Mechanism



The above figure shows a four-bar mechanism. The lengths of the crankshaft, coupler, and rocker are 0.2 m, 0.4 m, and 0.3 m, respectively. The crankshaft of the mechanism is assumed to rotate with a constant angular velocity 15 rad/s. The initial orientation of the crankshaft is assumed to be 45.

Constraint Equation =

$$\begin{pmatrix} r_{x1} \\ r_{y1} \\ \Theta_1 \\ r_{x2} - \frac{\cos(\Theta_2)}{10} \\ r_{y2} - \frac{\sin(\Theta_2)}{10} \\ r_{x3} - r_{x2} - \frac{\cos(\Theta_2)}{10} - \frac{\cos(\Theta_3)}{5} \\ r_{y3} - r_{y2} - \frac{\sin(\Theta_2)}{10} - \frac{\sin(\Theta_3)}{5} \\ r_{x4} - r_{x3} - \frac{\cos(\Theta_3)}{5} - \sigma_2 \\ r_{y4} - r_{y3} - \frac{\sin(\Theta_3)}{5} - \sigma_1 \\ r_{x4} - r_{x1} - \frac{7\cos(\Theta_1)}{20} + \sigma_2 \\ r_{y4} - r_{y1} - \frac{7\sin(\Theta_1)}{20} + \sigma_1 \\ \Theta_2 - 15\,t - \frac{\pi}{4} \end{pmatrix}$$

where

$$\sigma_1 = \frac{3\sin(\Theta_4)}{20}$$

$$\sigma_2 = \frac{3\cos(\Theta_4)}{20}$$

Jacobian (Cq) =

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{\sin(\Theta_2)}{10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \sigma_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & \frac{\sin(\Theta_2)}{10} & 1 & 0 & \frac{\sin(\Theta_3)}{5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & \sigma_3 & 0 & 1 & \sigma_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & \frac{\sin(\Theta_3)}{5} & 1 & 0 & \sigma_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \sigma_4 & 0 & 1 & -\sigma_2 \\ -1 & 0 & \frac{7\sin(\Theta_1)}{20} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\sigma_1 \\ 0 & -1 & -\frac{7\cos(\Theta_1)}{20} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \sigma_2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \frac{3\sin(\Theta_4)}{20}$$

$$\sigma_2 = \frac{3\cos(\Theta_4)}{20}$$

$$\sigma_3 = -\frac{\cos(\Theta_2)}{10}$$

$$\sigma_4 = -\frac{\cos(\Theta_3)}{5}$$

Ct =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -15 \end{pmatrix}$$

Qd =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ -\sigma_5 \\ -\sigma_3 \\ -\sigma_5 - \sigma_6 \\ -\sigma_3 - \sigma_4 \\ -\sigma_6 - \sigma_2 \\ -\sigma_4 - \sigma_1 \\ \sigma_2 - \dfrac{7\cos(\theta_1(t))\left(\frac{\partial}{\partial t}\,\theta_1(t)\right)^2}{20} \\ \sigma_1 - \dfrac{7\sin(\theta_1(t))\left(\frac{\partial}{\partial t}\,\theta_1(t)\right)^2}{20} \\ 0 \end{pmatrix}$$

where

$$\sigma_1 = \frac{3\sin(\theta_4(t))\left(\frac{\partial}{\partial t}\,\theta_4(t)\right)^2}{20}$$

$$\sigma_2 = \frac{3\cos(\theta_4(t))\left(\frac{\partial}{\partial t}\,\theta_4(t)\right)^2}{20}$$

$$\sigma_3 = \frac{\sin(\theta_2(t))\left(\frac{\partial}{\partial t}\,\theta_2(t)\right)^2}{10}$$

$$\sigma_4 = \frac{\sin(\theta_3(t))\left(\frac{\partial}{\partial t}\,\theta_3(t)\right)^2}{5}$$

$$\sigma_5 = \frac{\cos(\theta_2(t))\left(\frac{\partial}{\partial t}\,\theta_2(t)\right)^2}{10}$$

$$\sigma_6 = \frac{\cos(\theta_3(t))\left(\frac{\partial}{\partial t}\,\theta_3(t)\right)^2}{5}$$

This inversion of four bar mechanism,given this initial conditions can give us 2 different configurations.

Configuration 1

The Table Below compares position,velocity and acceleration solved with different methods

1. Newton - Raphson (NR)
2. Loop - Closure Equations (LC)

At time = 0 s

Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0.0000	-0.0000	-0.0000	0.0000
0	0	-0.0000	-0.0000	0.0000	-0.0000
0	0	0.0000	0.0000	-0.0000	-0.0000
0.0707	0.0707	-1.0607	-1.0607	-15.9099	-15.9099
0.0707	0.0707	1.0607	1.0607	-15.9099	-15.9099
0.7854	0.7854	15.0000	15.0000	-0.0000	0.0000
0.1669	0.1669	-2.6660	-2.6660	38.0235	38.0235
-0.0569	-0.0569	2.0512	2.0512	-21.3133	-21.3133
4.8403	-1.4428	-2.7460	-2.7460	353.0653	353.0653
0.2712	0.2712	-1.6054	-1.6054	53.9334	53.9334
-0.1277	-0.1277	0.9906	0.9906	-5.4034	-5.4034
7.3011	1.0179	-12.5759	-12.5759	324.9089	324.9089

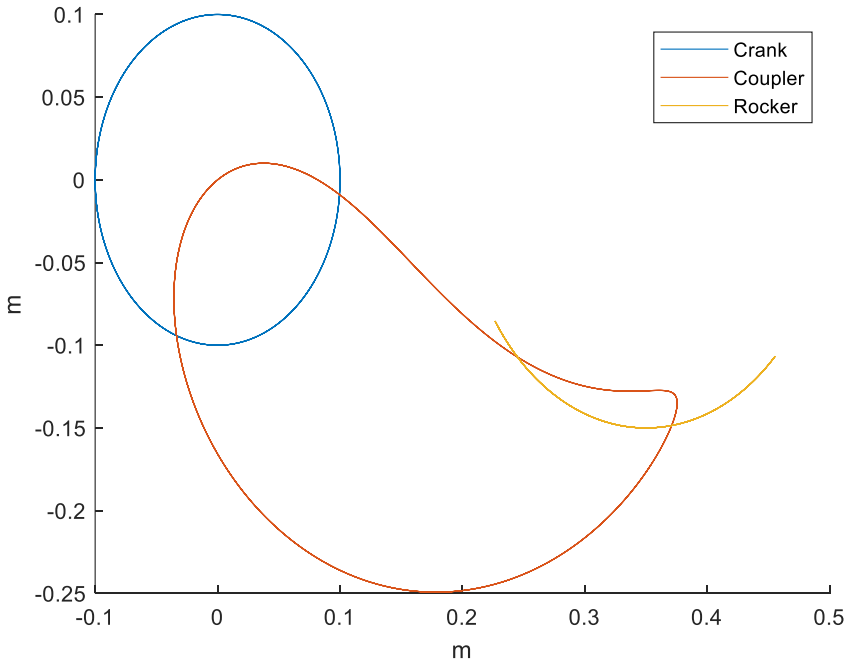
At time = 1 s

Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0.0000	0.0000	0.0000	-0.0000
0	0	0.0000	0.0000	0.0000	-0.0000
0	0	0.0000	0.0000	0.0000	-0.0000
-0.0998	-0.0997	0.0936	0.1160	22.4562	22.4326
-0.0062	-0.0077	-1.4971	-1.4955	1.4035	1.7405
15.7704	15.7854	15.0000	15.0000	0.0000	-0.0000
-0.0280	-0.0332	0.7248	-0.4043	34.8246	33.8331
-0.1151	0.0958	-2.0951	-2.0408	-3.3699	8.2025
5.7442	0.5901	5.2379	5.7181	-52.3954	50.3081
0.2468	0.2415	0.6313	-0.5203	12.3684	11.4005
-0.1089	0.1036	-0.5980	-0.5453	-4.7735	6.4620
7.0956	-0.7620	5.7971	5.0247	81.7457	-83.6360

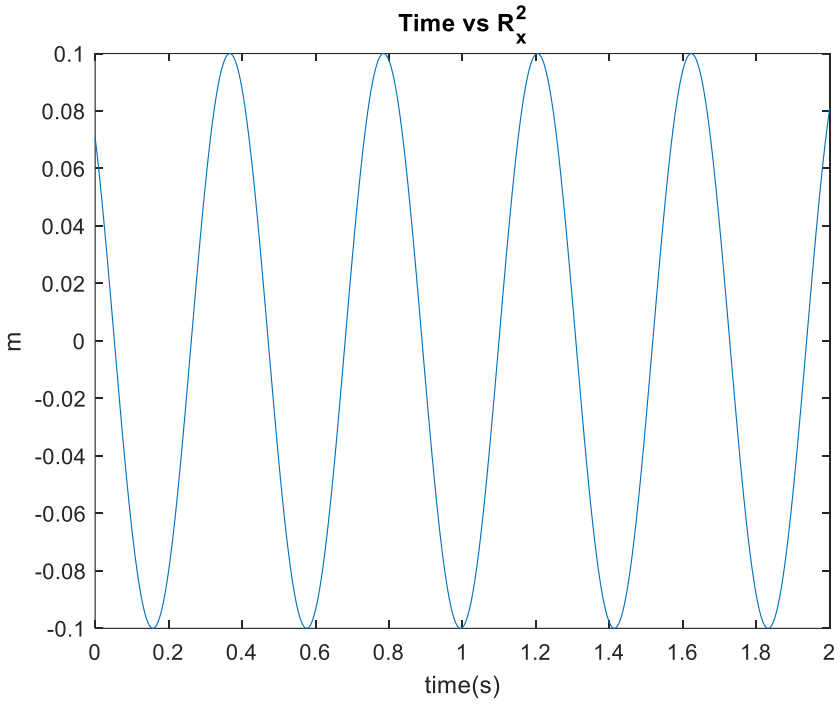
At time = 2 s

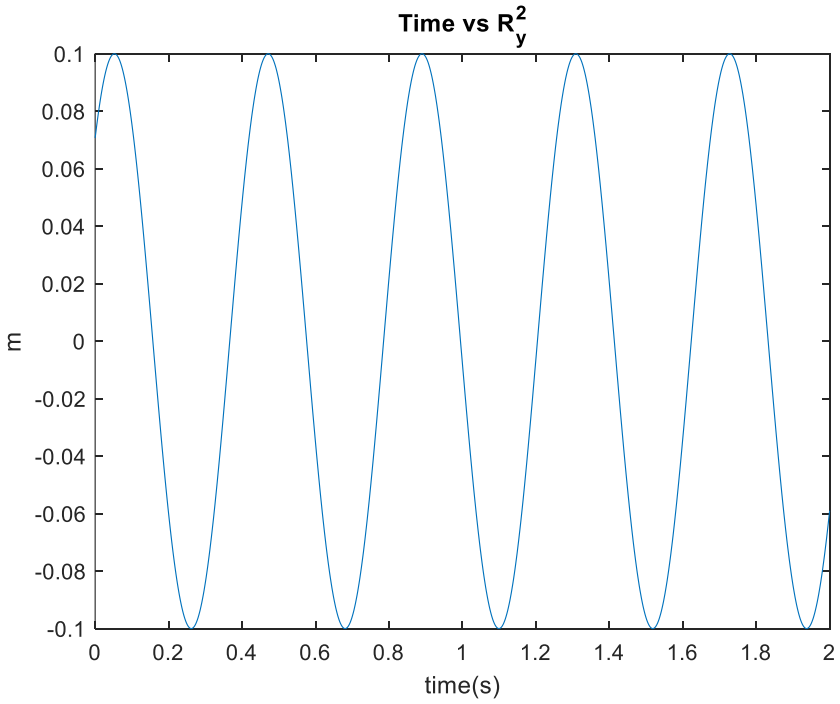
Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0.0000	-0.0000	0.0000	0.0000
0	0	0.0000	0.0000	-0.0000	0.0000
0	0	-0.0000	0.0000	-0.0000	-0.0000
-0.0816	0.0808	0.8676	0.8844	18.3543	-18.1736
-0.0578	-0.0590	-1.2236	1.2116	13.0142	13.2654
3.7584	30.7854	15.0000	15.0000	0.0000	-0.0000
0.0225	0.1968	1.9851	1.6020	31.1799	54.0988
-0.1900	0.0790	-1.8231	1.8334	18.3010	0.4524
5.9023	1.3937	3.3617	-3.0641	-46.1442	-468.1296
0.2791	0.2910	1.1175	0.7177	12.8256	72.2724
-0.1322	0.1379	-0.5995	0.6219	5.2868	-12.8130
7.3616	-1.1666	8.4544	6.3306	58.6876	-409.4020

You can see from these tables that the answers for the angles do not exactly match,One should keep in mind that the angles are periodical with a period of 2*pi.So the answers are identical only they are in different cycle.You can also see that the answers at the time t=2s do not match at all,those kind of answers can be due to singularities or sometimes the newton-raphson fails to converge at certain points.



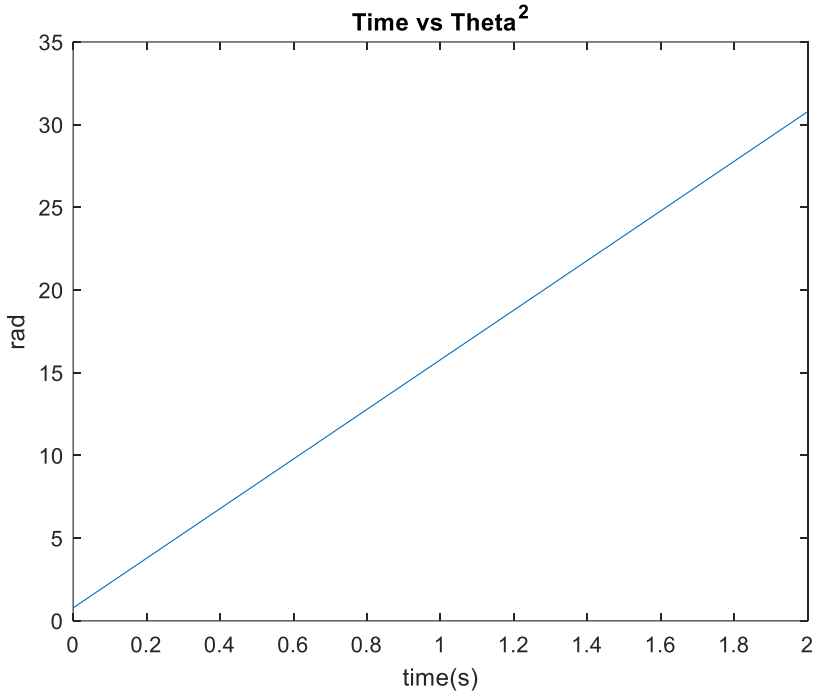
The above figure shows us the path traces of the different links.

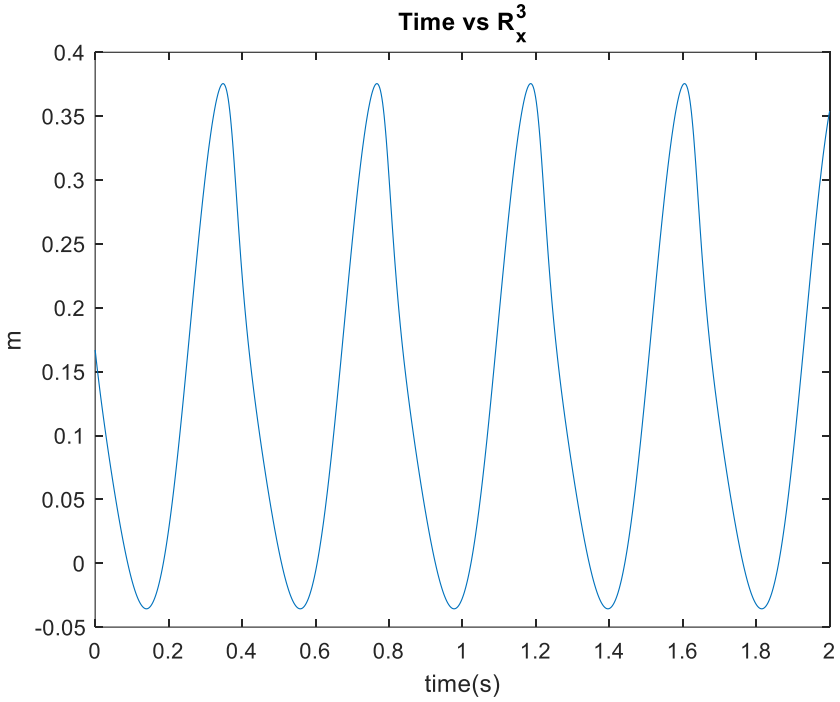

Time vs R_x^2
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
m
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

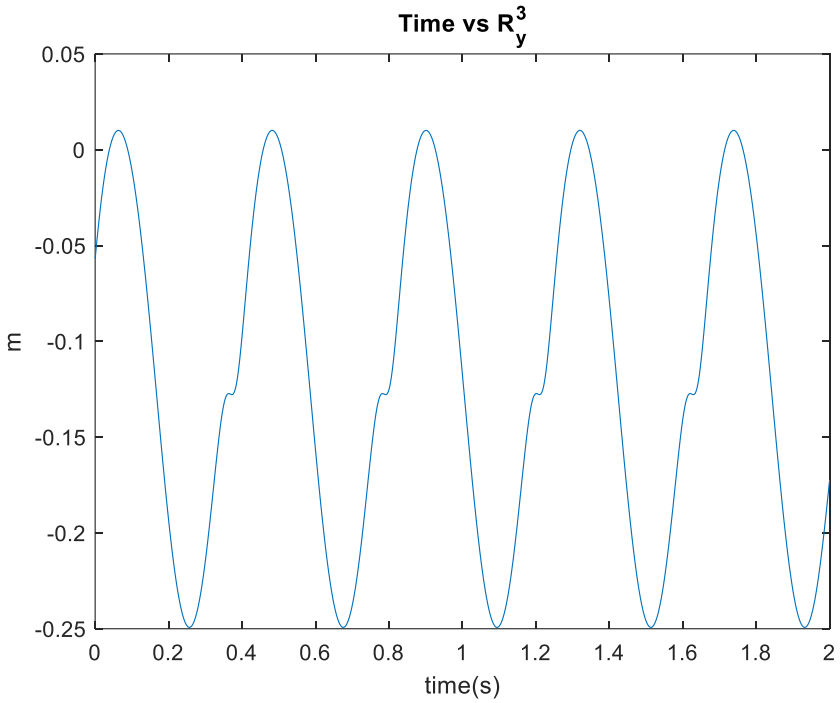

Time vs R_y^2
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
m
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

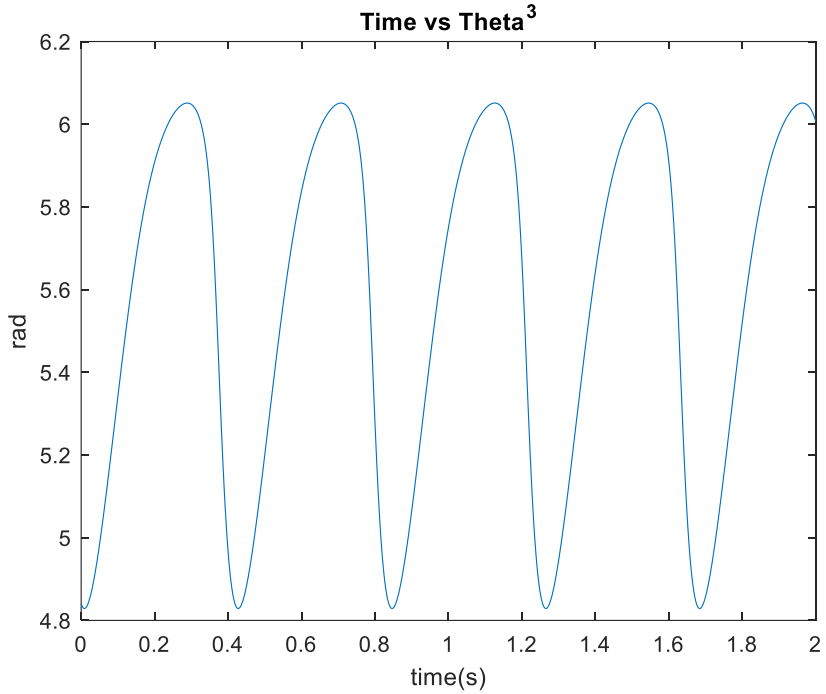
Project Report

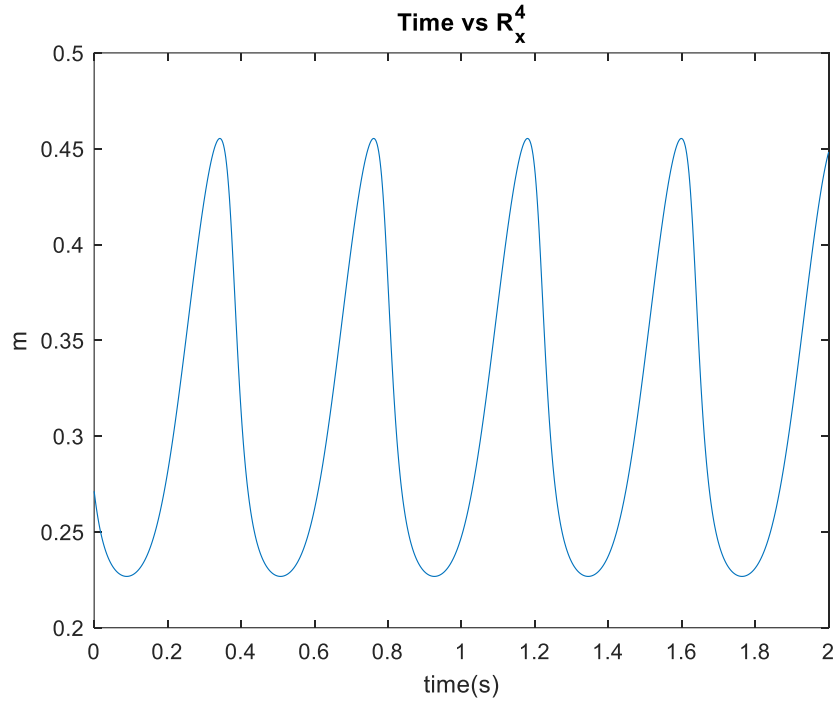
Meet N Mevada

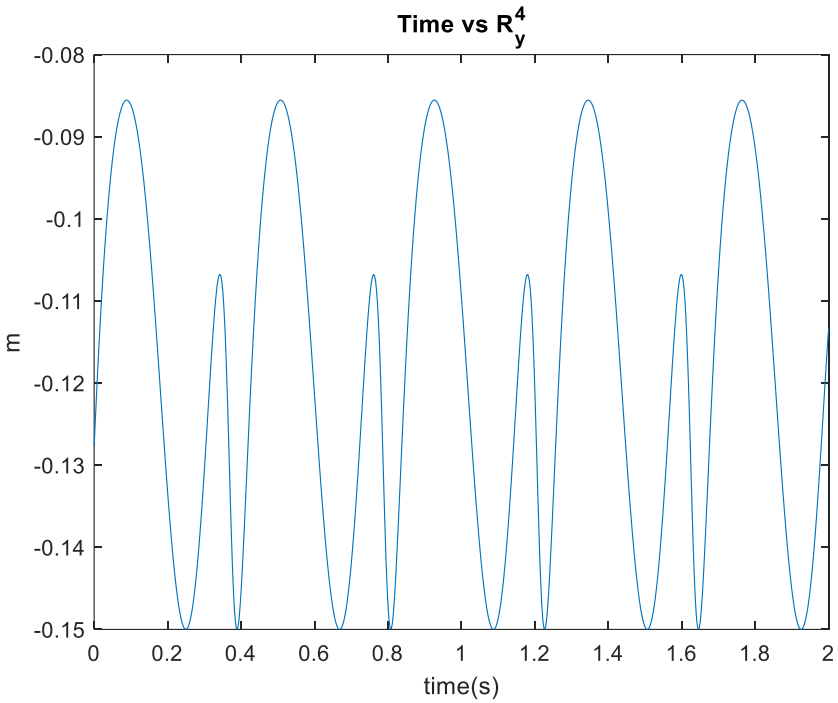


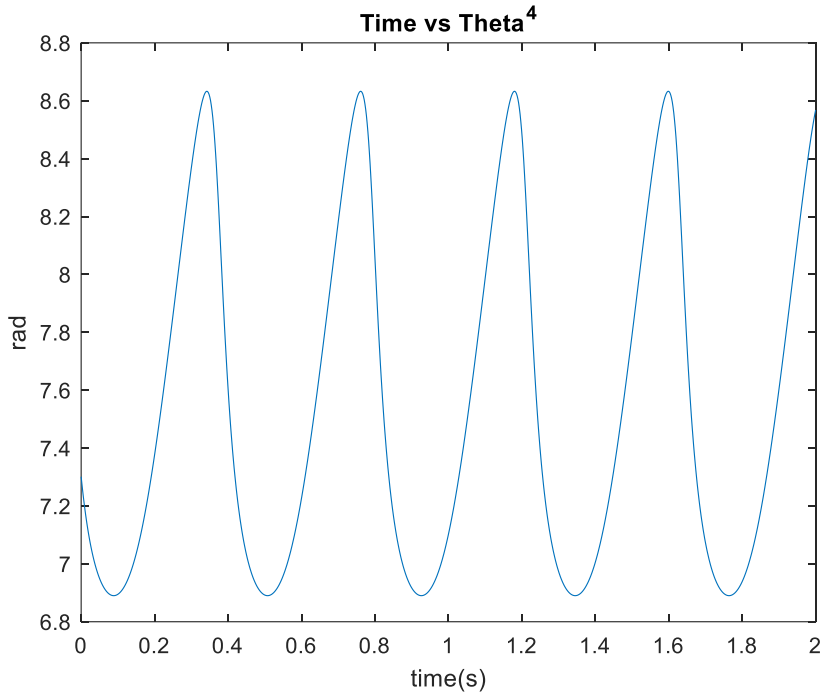



Time vs R_y^3
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
m
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

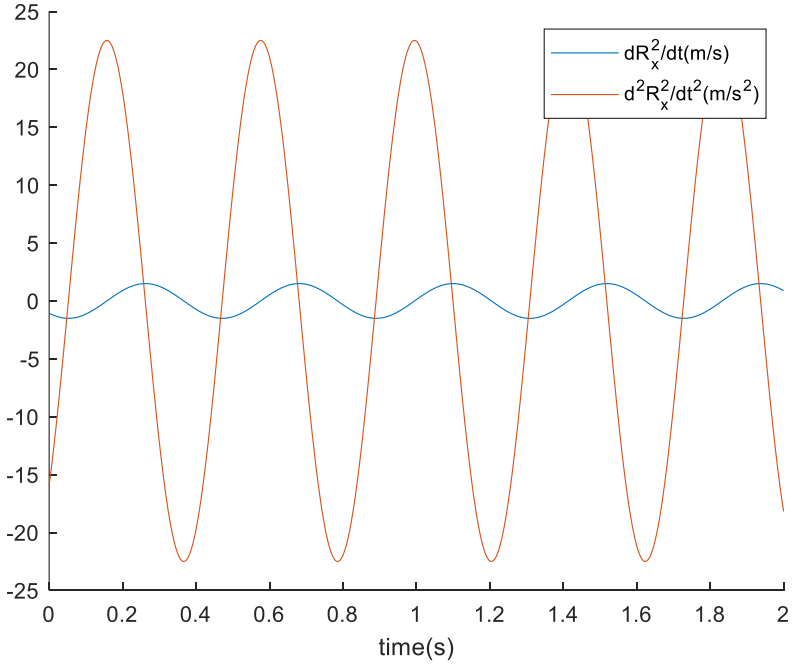

Time vs $Theta^3$
6.2
6
5.8
5.6
5.4
5.2
5
4.8
rad
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


Time vs R_x^4
0.5
0.45
0.4
0.35
0.3
0.25
0.2
m
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


Time vs R_y^4
-0.08
-0.09
-0.1
-0.11
-0.12
-0.13
-0.14
-0.15
m
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

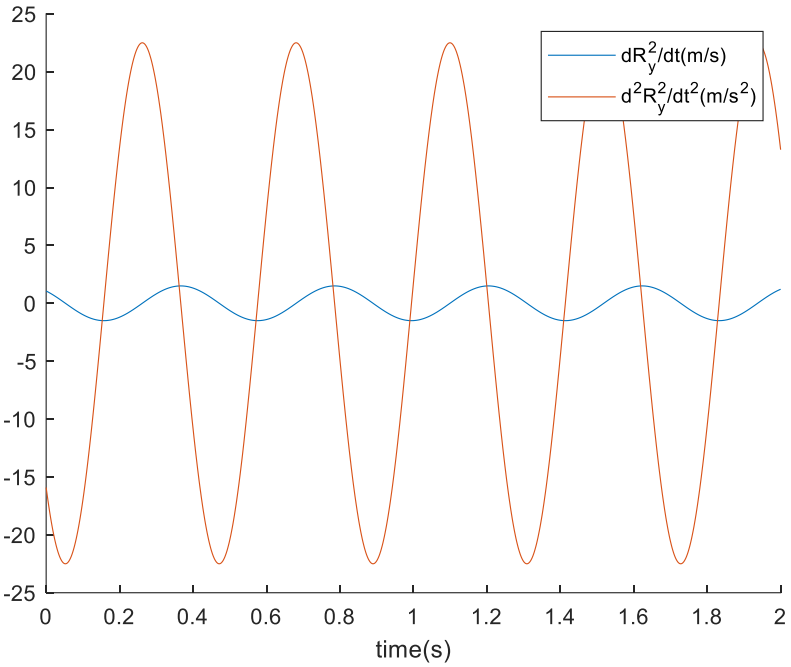


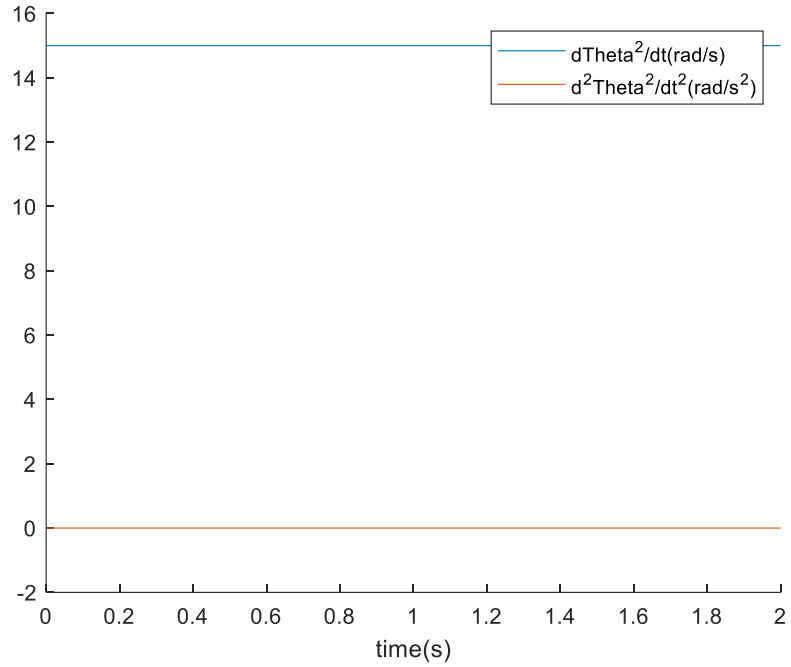
Above ones were the plots of different position vs time.Now the following ones will be Velocity and Acceleration vs time.

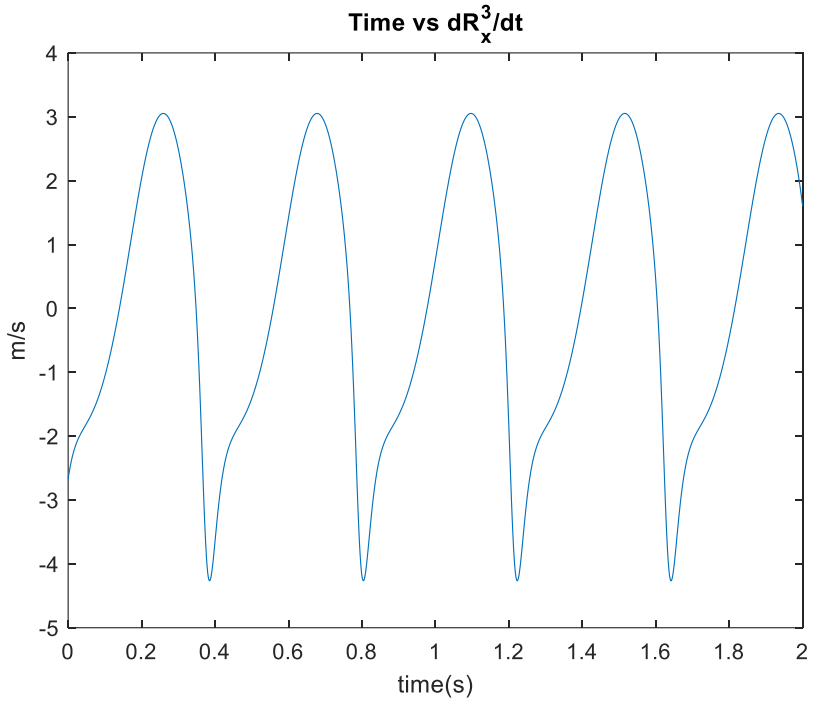


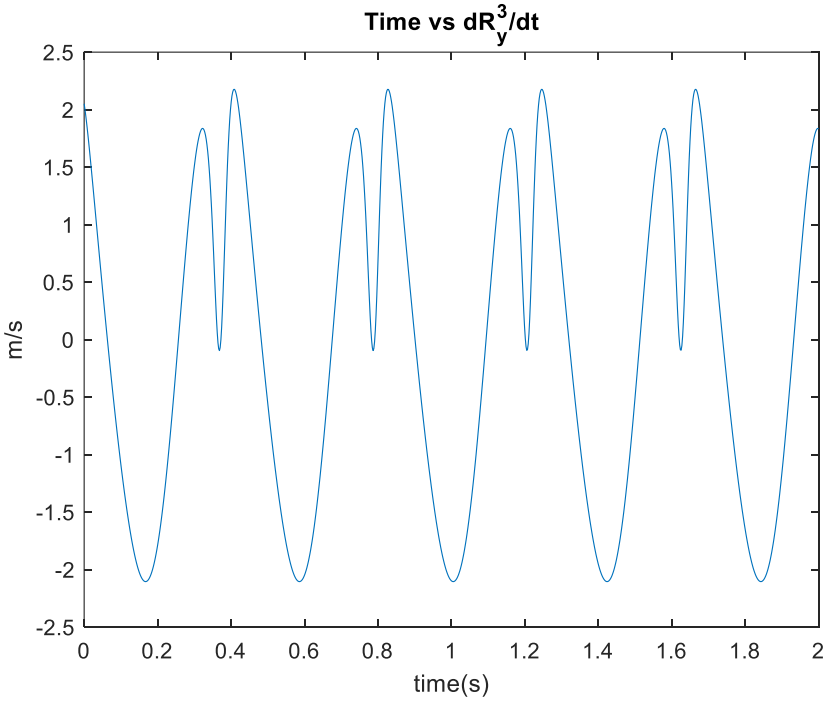
Project Report

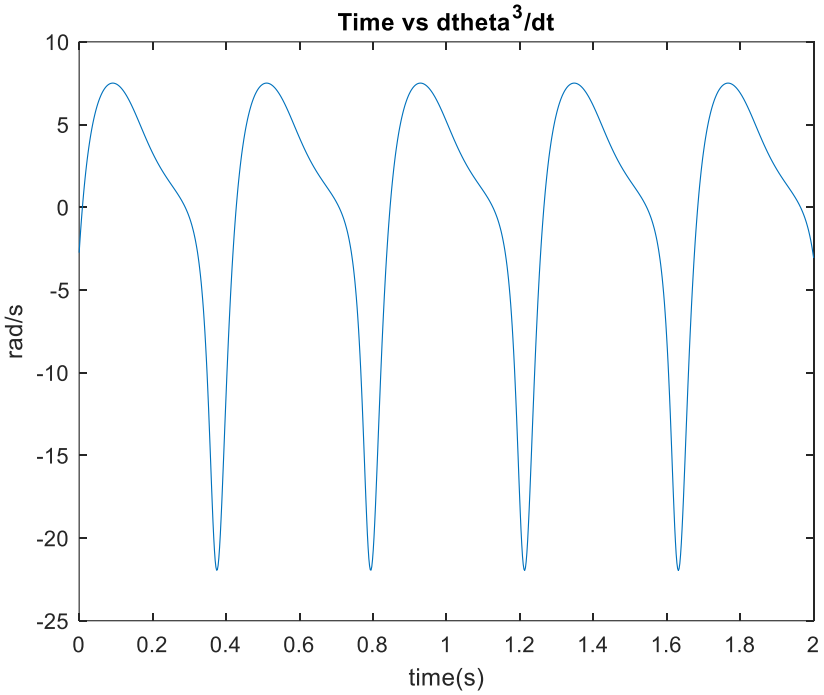
Meet N Mevada

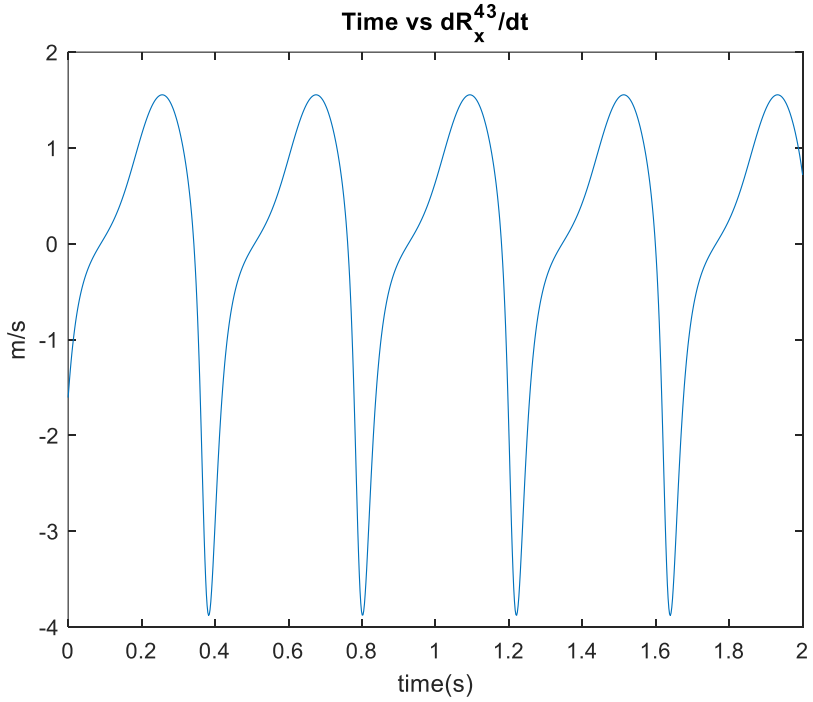


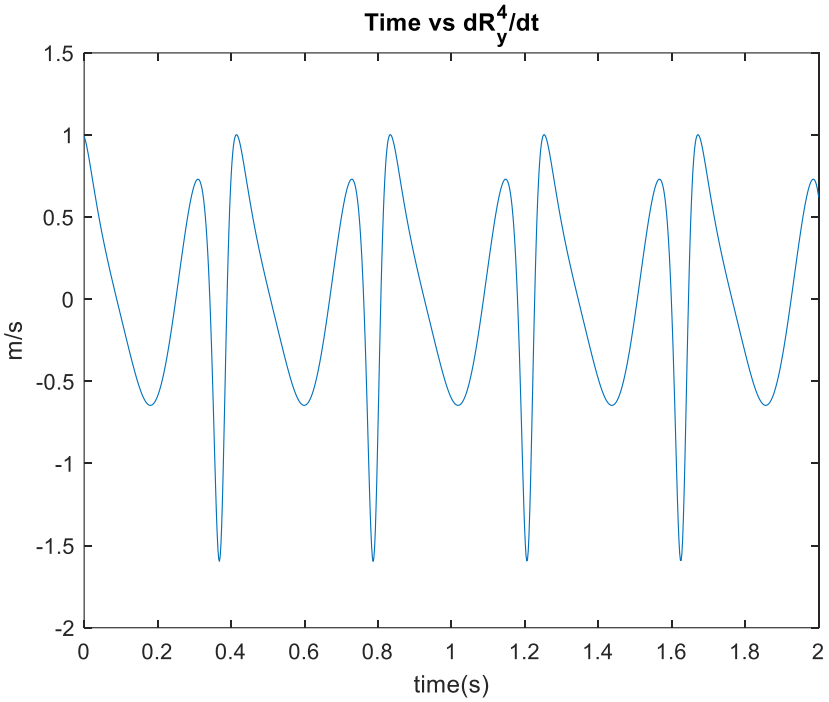


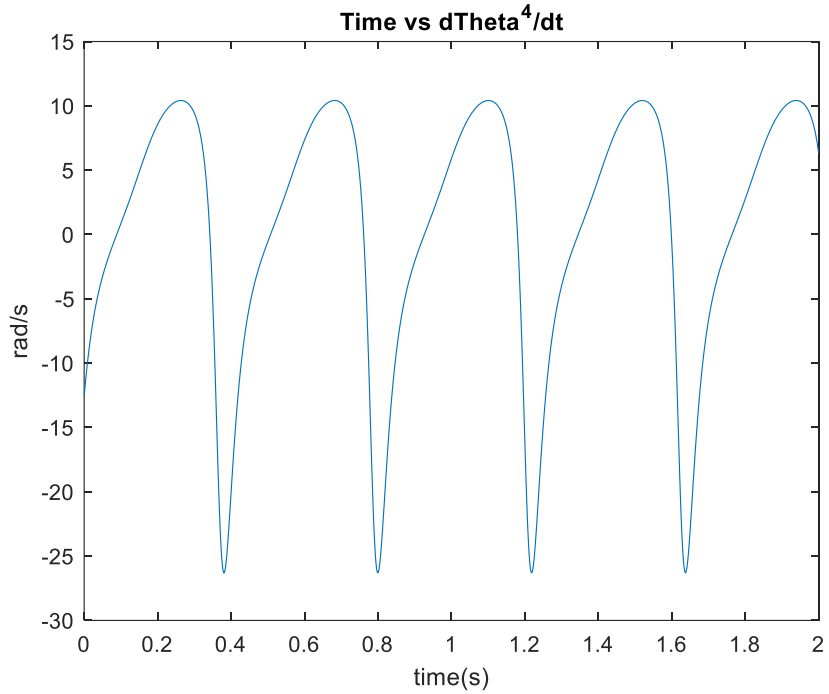

Time vs dR$_x^3$/dt
m/s
time(s)
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Time vs dR$_y^3$/dt
m/s
time(s)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Time vs dtheta3/dt
rad/s
time(s)
10
5
0
-5
-10
-15
-20
-25
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

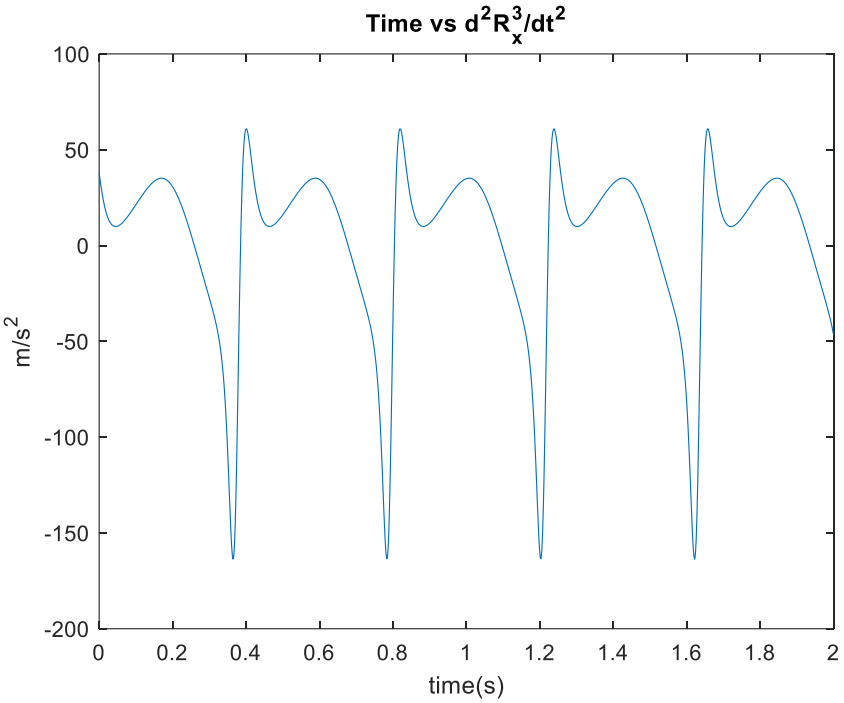

Time vs dRx43/dt
m/s
time(s)
2
1
0
-1
-2
-3
-4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

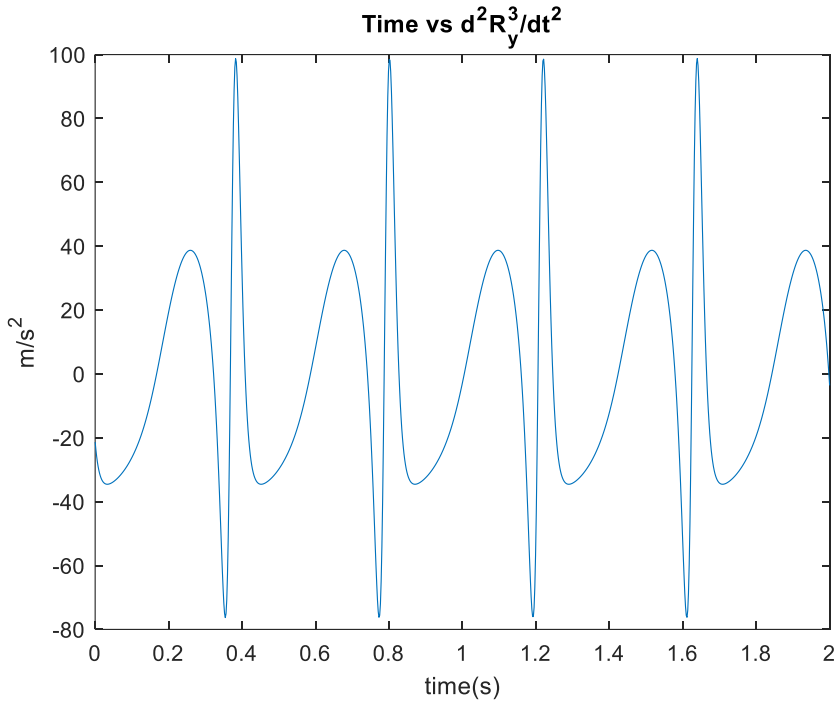

Time vs dR$_y^4$/dt
m/s
time(s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

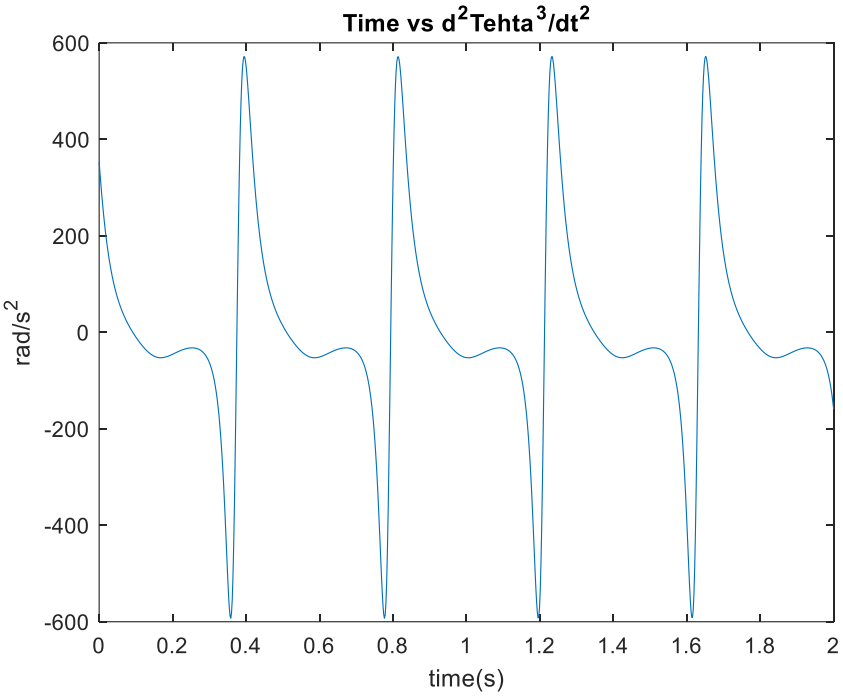

Time vs dTheta^4/dt
rad/s
time(s)
15
10
5
0
-5
-10
-15
-20
-25
-30
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

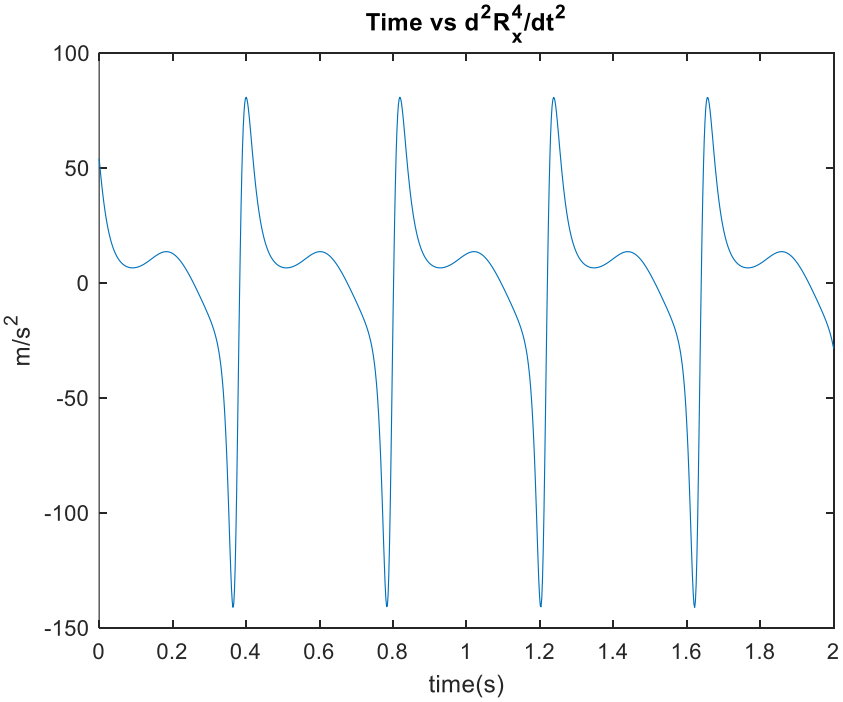
Project Report

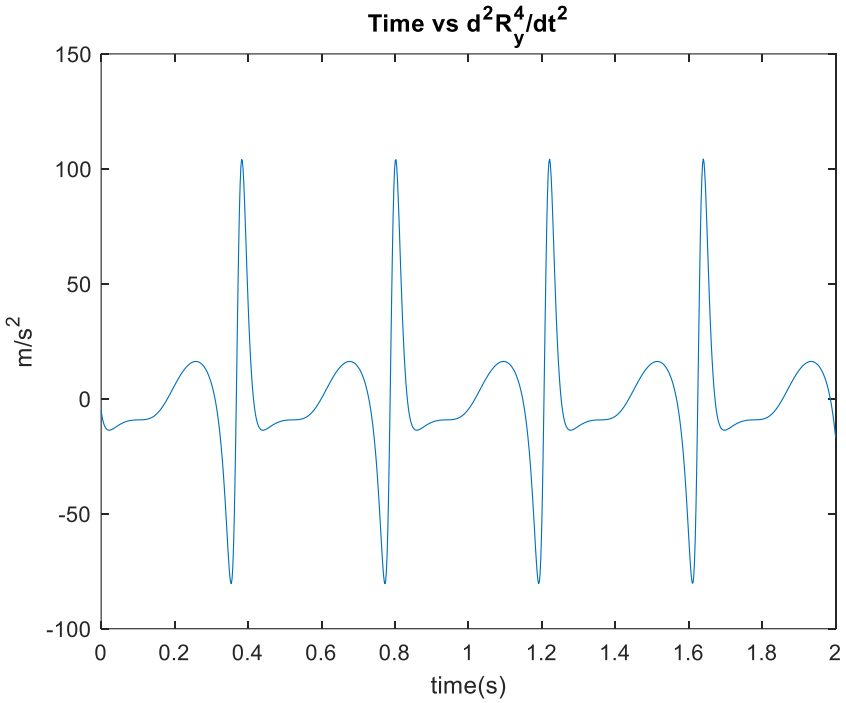
Meet N Mevada

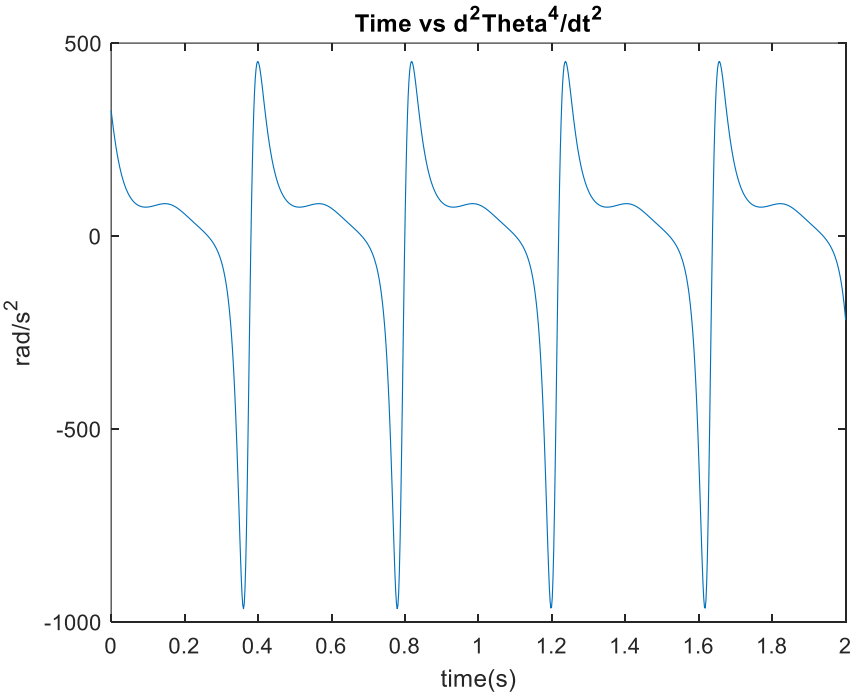





Time vs d2Tehta3/dt2
600
400
200
0
-200
-400
-600
rad/s2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


Time vs d2Rx4/dt2
100
50
0
-50
-100
-150
m/s2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


Time vs d^{2}R$_y^4$/dt^2
150
100
50
0
-50
-100
m/s^2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)


Time vs d^{2}Theta^4/dt^2
500
0
-500
-1000
rad/s^2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(s)

Configuration 2

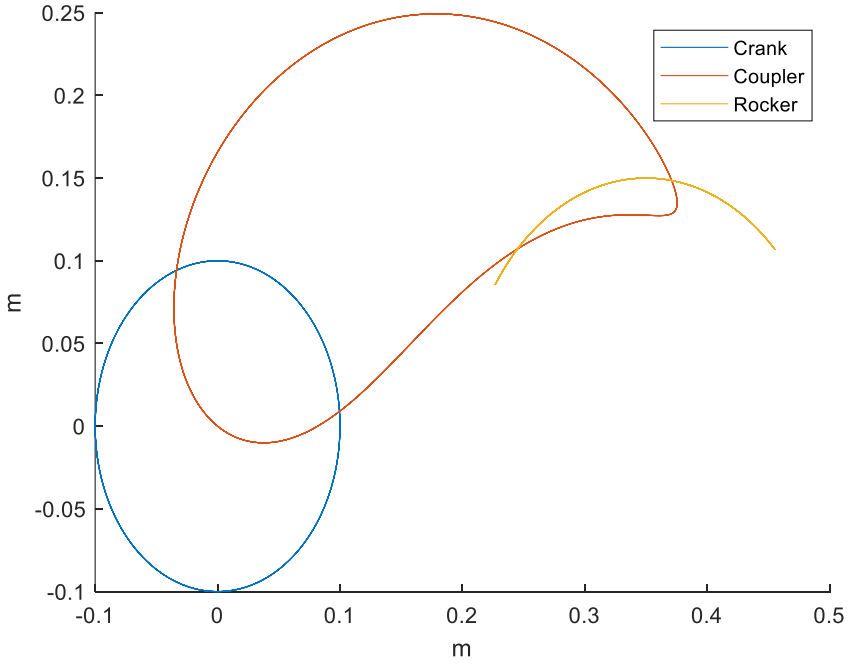
The Table Below compares position,velocity and acceleration solved with different methods
1. Newton - Raphson (NR)
2. Loop - Closure Equations (LC)

At time = 0 s

Position(NR)	Position(LC)	Velocity(NR)	Velocity(LC)	Acc.(NR)	Acc.(LC)
0	0	0.0000	-0.0000	0.0000	-0.0000
0	0	-0.0000	0.0000	-0.0000	0.0000
0	0	0.0000	0.0000	0.0000	-0.0000
0.0707	0.0707	-1.0607	-1.0607	-15.9099	-15.9099
0.0707	0.0707	1.0607	1.0607	-15.9099	-15.9099
0.7854	0.7854	15.0000	15.0000	0.0000	0.0000
0.3351	0.3351	-2.0348	-2.0348	-37.5065	-37.5065
0.1911	0.1911	1.7841	1.7841	-12.0971	-12.0971
-12.3152	0.2512	-1.7406	-1.7406	102.5863	102.5863
0.4394	0.4394	-0.9741	-0.9741	-21.5966	-21.5966
0.1204	0.1204	0.7235	0.7235	3.8129	3.8129
-8.4928	-2.2096	8.0893	8.0893	130.7428	130.7428

As mentioned earlier,This four bar inversion can have singularities.Singularities come when at that point in time,given the condition multiple geometrically configurations are possible .When we solve constraint equation for the vector q and substitute those values in the Jacobian matrix,the matrix becomes singular(Determinant=0) and at that point calculating velocity and acceleration is not possible.Given this reason I could not find velocities and acceleration for different points in time for this configuration.I'm not stating that calculating velocity and acceleration for all of those points in time is not possible but my program needs a little more tweaking to be able to do so.In order to avoid singularities I scripted my program to assign NaN values to those velocities and accelerations.I was still able to find the position vector q for all of these points and the trace path for the links is shown in the next page.

A conclusion can be drawn from this plot that this configuration is a mirror image of the previous configuration from y=0 line or x-axis.From this analogy one can extend the imagination and can infer that all of the plots for position,velocity and acceleration for link 3 and 4 will be mirror image of those in the first configuration.



Concluding Remarks

the initial guess we put in affects the outcome of the program,at first glance this does not seem right but for the mechanism with multiple configurations this is very true.As seen in the previous section,To get these different configuration the only change one has to make in the inputs is the initial guess.

This program tries to generalize the process of making constraint equations for multi-body systems.Though right now this program is only capable of simulating very basic of kinematic chains like Slider Crank and Crank Rocker,it shows us that with further improvement we can generalize it even more and can make a program that can truly simulate kinematic chains with higher complexities and different joints.

Appendix

A. Script for Basic Joint Constraints

1) Ground

```
function ConGrd=Ground(coordinates,Constants)


ConGrd(1,:)=coordinates(1)-Constants(1);
ConGrd(2,:)=coordinates(2)-Constants(2);
ConGrd(3,:)=coordinates(3)-Constants(3);


end
```

2) Revolute Joint

```
function ConRev=RevoluteJoint(prev_body,body,Point_Coords)


ConRev=[body(1);body(2)]+([cos(body(3)) -sin(body(3));sin(body(3))
cos(body(3))]*Point_Coords(:,1))-[prev_body(1);prev_body(2)]-[cos(prev_body(3))
-sin(prev_body(3));sin(prev_body(3)) cos(prev_body(3))]*Point_Coords(:,2);
end
```

3) Simple Prismatic Joint

```
function ConPrism=Prismatic(coordinates,Constants)


ConPrism(1,:)=coordinates(2)-Constants(1);
ConPrism(2,:)=coordinates(3)-Constants(2);


end
```

B. Script for formulating constraint equation and Jacobian

Jacobian

```
function Cq=Jacobian(ConEq,q)


for i=1:length(ConEq)
for j=1:length(q)
Cq(i,j)=diff(ConEq(i),q(j));
end
end


end
```

Constraint Equation

```
function
[q,qt,ConEqt,Eq,Cq]=ContraintEq(number_of_bodies,types_of_joints,point_coords,GndCon,PrismCo
n,theta_input_number,angular_velocity,initial_theta,t)




syms Theta r_x r_y [number_of_bodies 1]


syms body [number_of_bodies 3]


for j=1:number_of_bodies
body(j,:)=[r_x(j) r_y(j) Theta(j)];
end







for k=1:length(types_of_joints)
switch types_of_joints{k}
case 'Ground'
ConEq(k:k+2,:)=Ground(body(k,:),GndCon);
case 'Revolute'
if k<length(types_of_joints)
switch types_of_joints{k-1}
case 'Ground'
ConEq(2*k:2*k+1,:)=RevoluteJoint(zeros(3,1),body(k,:),point_coords{k});
otherwise
ConEq(2*k:2*k+1,:)=RevoluteJoint(body(k-1,:),body(k,:),point_coords{k});
end
else
ConEq(2*k:2*k+1,:)=RevoluteJoint(body(1,:),body(k-1,:),point_coords{k});
end
case 'Prismatic'
ConEq(2*k:2*k+1,:)=Prismatic(body(k-1,:),PrismCon);
end
end
ConEq(3*number_of_bodies,:)=body(theta_input_number,3)-angular_velocity*t-initial_theta;
for k=1:number_of_bodies
g(:,k)=[r_x(k);r_y(k);Theta(k)];
end
q=g(:);
Cq=Jacobian(ConEq,q);
syms R_x(t) R_y(t) theta(t) [number_of_bodies 1]
qt=subs(q,[r_x,r_y,Theta],[R_x(t),R_y(t),theta(t)]);
ConEqt=subs(ConEq,[r_x,r_y,Theta],[R_x(t),R_y(t),theta(t)]);
Eq=ConEq;
end
```

C. Script for formulating vector Qd

```
function Qd=accelerationd(qt,Con,t)
f=diff(qt,t,2);
g=diff(Con,t,2);
for i=1:length(Con)
Qd(i,:)=g(i);
for j=1:length(qt)
Qd(i,:)=Qd(i,:)-f(j)*diff(g(i),f(j),1);
end


end
Qd=-Qd;
end
```

D. Script for Newton - Raphson method

```
function [solvedq,solvedCon,solvedCq]=NewtonRaphson(Cq,ConEq,intial_guess,q)


dq=-inv(double(subs(Cq,q,intial_guess)))*double(subs(ConEq,q,intial_guess));




gq=intial_guess+dq;


i=50;


while isAlways(and(and(i>0,norm(dq)>5*10^-8),norm(double(subs(ConEq,q,gq)))>5*10^-8))
dq=-inv(double(subs(Cq,q,gq)))*double(subs(ConEq,q,gq));


gq=gq+dq;
i=i-1;


end
solvedCq=double(subs(Cq,q,gq));
solvedq=double(gq);
solvedCon=double(subs(ConEq,q,solvedq));
end
```

E. Script for Velocity Analysis

```
function dq=VelocityAnalysis(qt,q,SolvedCq,angular_velocity)
```

```
syms t
Ct=zeros(length(q), 1);


Ct(end)=-angular_velocity;


dq=-pinv(SolvedCq)*Ct;
```

F. Script for Acceleration Analysis

```
function acceleration=accelerationAnalysis(q,cq,qd,solvedq)


conEq=cq*qd;
cqq=Jacobian(conEq,q);
cqq=double(subs(cqq,q,solvedq));
qd=-cqq*qd;


acceleration=pinv(double(subs(cq,q,solvedq)))*qd;


end
```

G. Script for simulating the system over time

```
function
[time,position,velocity,acceleration]=PVelocity(q,qt,Cq,conEq,initial_guess,time_interval,th
eta_input,initial_theta,angular_velocity,point_multiplier)


time=linspace(time_interval(1),time_interval(2),(time_interval(2)-time_interval(1))*point_mu
ltiplier);
velocity=zeros(length(q),length(time));
position=zeros(length(q),length(time));
acceleration=zeros(length(q),length(time));
f=1000;
for k=time
j=round(k*((time_interval(2)-time_interval(1))*point_multiplier-1)/(time_interval(2)-time_in
terval(1))+1,0);
f=f-1
if j==1
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,initial_guess,q);
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
else
if isnan(velocity(:,j-1))==zeros(12,1)
if det(double(subs(Cq,q,position(:,j-1))))>10^-8
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,position(:,j-1),q);
```

```
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
end
else
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
if j>2
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,position(:,j-2),q);
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
end
else
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,initial_guess,q);
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
end
end
end
```

```
else
if det(double(subs(Cq,q,position(:,j-2))))>10^-8
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,position(:,j-2),q);
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
end
else
if j>3
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,position(:,j-3),q);
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
```

```
position(:,j)=solvedq;
velocity(:,j)=dq;
end
else
conEq(end)=q(3*theta_input)-angular_velocity*k-initial_theta;
[solvedq,~,solvedcq]=NewtonRaphson(Cq,conEq,initial_guess,q);
if det(solvedcq)<10^-8
position(:,j)=solvedq;
acceleration(:,j)=nan;
velocity(:,j)=nan;
else
dq=VelocityAnalysis(qt,q,solvedcq,angular_velocity);
acceleration(:,j)=accelerationAnalysis(q,Cq,dq,solvedq);
position(:,j)=solvedq;
velocity(:,j)=dq;
end
end
end
end


end
end
end
```

H. Script for generating local position vectors

```
function local_coordinates=Coordinates_generator(link_lengths,four_bar_inversion_type)


if four_bar_inversion_type=='Slider Crank'
local_coordinates={zeros(2,2),[-link_lengths(2)/2 0;0 0],[-link_lengths(3)/2
link_lengths(2)/2;0 0],[0 link_lengths(3)/2;0 0],zeros(2,2)};
elseif four_bar_inversion_type=='Crank Rocker'
local_coordinates={zeros(2,2),[-link_lengths(2)/2 0;0 0],[-link_lengths(3)/2
link_lengths(2)/2;0 0],[-link_lengths(4)/2 link_lengths(3)/2;0 0],[link_lengths(4)/2
link_lengths(1);0 0]};


end
end
```

I. Final Script

```
function
[q,qt,ConEq,Qd,Ct,Position,Velocity,Acceleration,time]=KinematicAnalysis(Inversion_Type,Link
_Lengths,number_of_bodies,GndCon,PrismCon,initial_theta,theta_input_number,angular_velocity,
time_interval,point_multiplier,initial_guess)
syms t
switch Inversion_Type
case 'Slider Crank'
Co=Coordinates_generator(Link_Lengths,'Slider Crank');
joint={'Ground','Revolute','Revolute','Revolute','Prismatic'};
case 'Crank Rocker'
Co=Coordinates_generator(Link_Lengths,'Crank Rocker');
joint={'Ground','Revolute','Revolute','Revolute','Revolute'};
end
[q,qt,con,ConEq,cq]=ContraintEq(number_of_bodies,joint,Co,GndCon,PrismCon,theta_input_number
,angular_velocity,initial_theta,t);
```

```
[time,Position,Velocity,Acceleration]=PVelocity(q,qt,cq,ConEq,initial_guess,time_interval,th
eta_input_number,initial_theta,angular_velocity,point_multiplier);
Ct=diff(ConEq,t,1);
Qd=accelerationd(qt,con,t);


hold on


plot(Position(4,:),Position(5,:))
plot(Position(7,:),Position(8,:))
plot(Position(10,:),Position(11,:))




plot(Position(6,:),Position(9,:))
plot(Position(6,:),Position(12,:))


legend({'Body_2','Body_3','Body_4','Crank_Angle Vs Connection_Rod_Angle','Crank_Angle Vs
Rocker_Angle'})


hold off



end
```