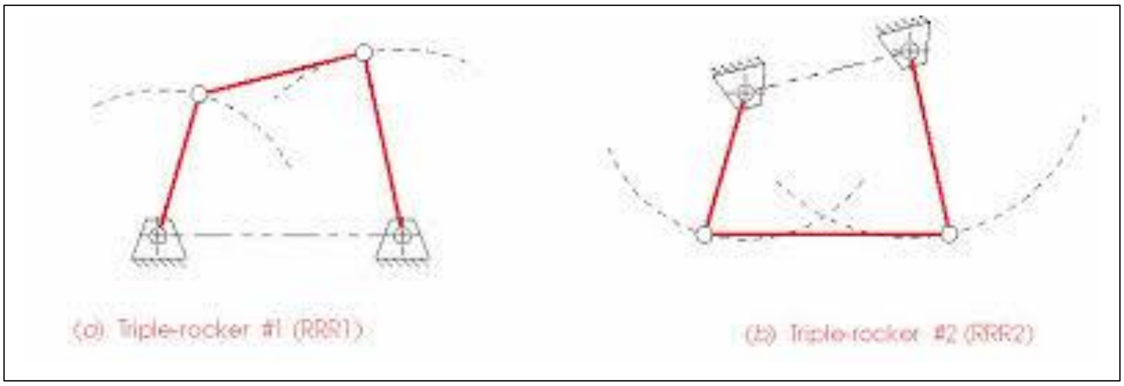


COMPUTATIONAL ENGINEERING MECHANICS-2

TRIPLE ROCKER MECHANISM

AMRITA VISHWA VIDYAPEETHAM

BY:

Adhithan P	CB.EN.U4AIE19003
Dinesh S	CB.EN.U4AIE19025
Kabilan N	CB.EN.U4AIE19033
Ch. Sai Harsha Reddy	CB.EN.U4AIE19018
SivaMaran M.A.C	CB.EN.U4AIE19061
Vijay Simmon S	CB.EN.U4AIE19068

ACKNOWLEDGEMENT

For the completion of this project, we have taken the guidance and trust of many respected faculty who have our utmost gratitude. As the completion of this project gave me much pleasure and more knowledge, we would like to show our gratitude for **Dr Gopalakrishnan EA**, Computational Engineering Mechanics teacher, in Amrita university for giving us good guidelines for the project throughout numerous consultations. We would also like to expand our gratitude to all those who have directly and indirectly guided us in helping us with this project.

Many people, especially my team mates have been a very vital part of this project and their opinions have been of the utmost importance for the completion of this project.

DECLARATION

We hereby declare that the project entitled **"Triple Rocker Mechanism"** submitted to Amrita Vishwa Vidyapeetham, Coimbatore is the record of original work done by us under the guidance of **Dr .Gopalakrishnan EA**, Department of Physics, Amrita Vishwa Vidyapeetham and this project work has not been submitted for any degree, diploma or other similar titles elsewhere. However, extracts of any content which has been used for this project have been duly acknowledged by providing the details in the references.

Adhithan P

Dinesh S

Kabilan N

Harsha Reddy

Sivamaran M.A.C

Vijay Simmon S

TABLE OF CONTENTS

ABSTRACT

A four-bar mechanism consists mainly of four planar links connected with four revolute joints. The input is usually given as rotary motion of a link and output can be obtained from the motion of another link or a coupler point. A numerical procedure will be studied and computer codes that generate the motion curves, angular velocity, and angular acceleration with respect to time graph will be presented.

WHAT IS TRIPLE ROCKER MECHANISM?

In Triple rocker mechanism, one link is fixed while the other three oscillates.

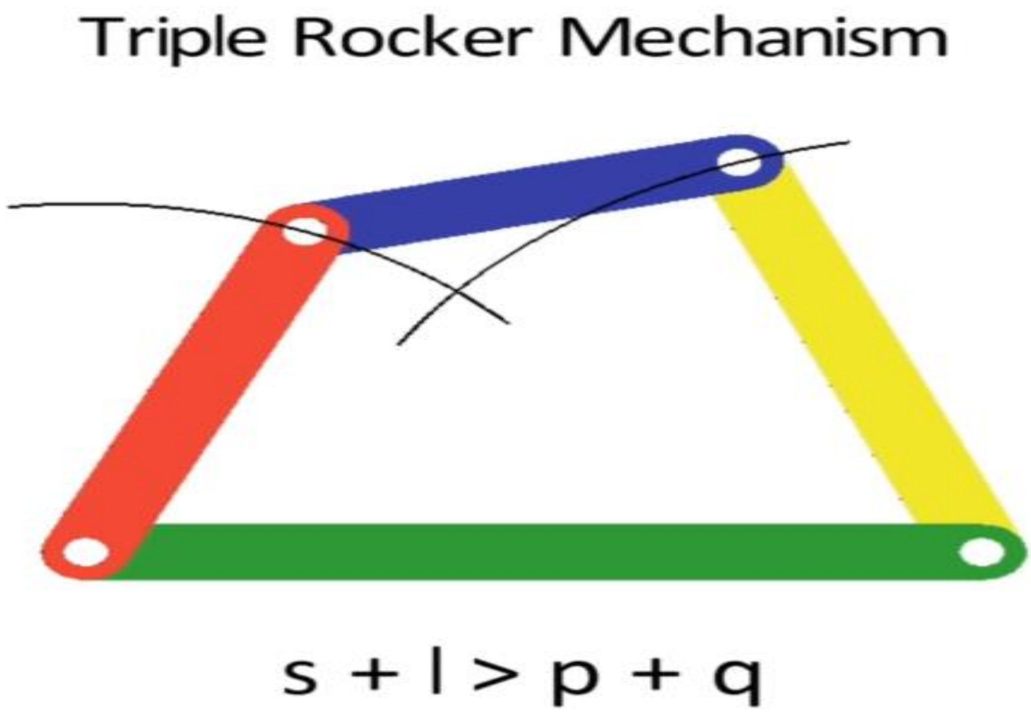
It is a **non Grashof** linkage for which the sum of the lengths of the shortest link and the longest link should be greater than the sum of the lengths of the other two links.

$s + l > p + q$

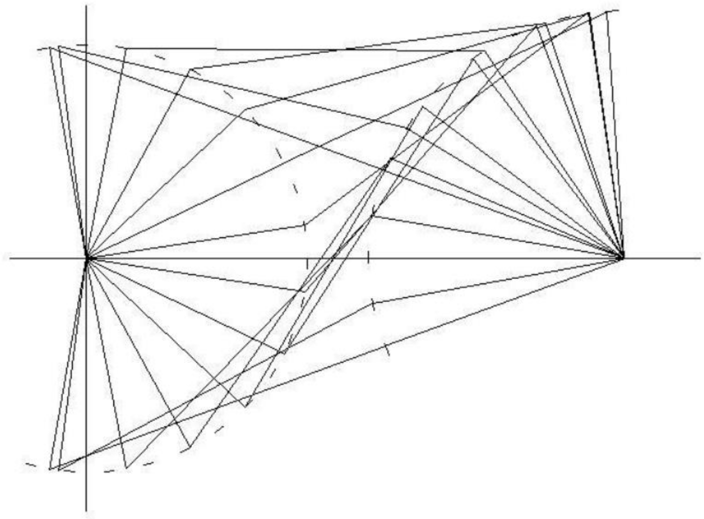
s = length of the shortest link

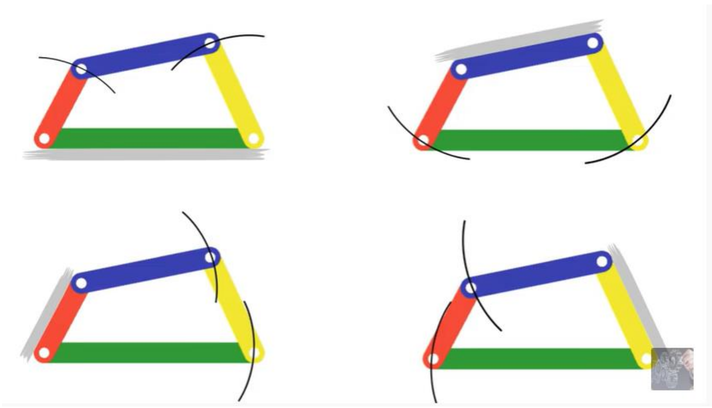
l = length of the largest link

p and q = lengths of the other two links



POSSIBLE TYPES IN TRIPLE ROCKER MECHANISM





COMPONENTS

Link

A link is one of the rigid bodies or members joined together. It does not consider small deflections due to strains in machine members. The word link is used in a general sense to include cams, gears, and other machine members in addition to cranks.

Frame

The fixed or stationary link in a mechanism is called the frame. When there is no link that is actually fixed, one link may be considered as being fixed and determine the motion of the other links relative to it.

Joint

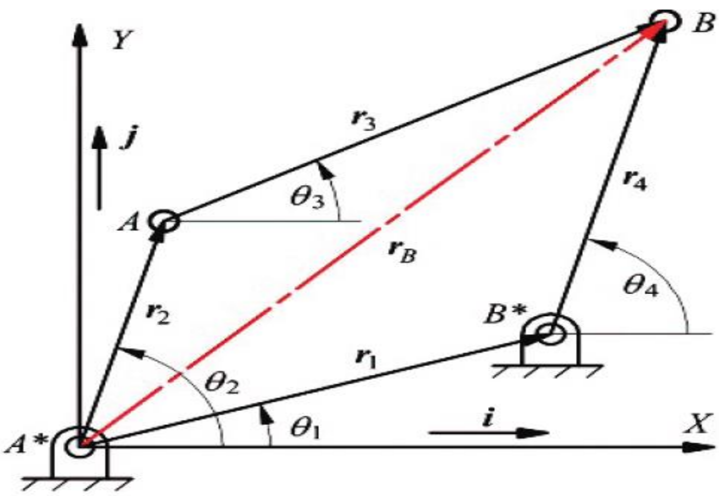
The connections between links that permit relative motion are called joints. An unconstrained rigid body has a mobility of six degrees of freedom. Each joint reduces the mobility of a system. The joint between a crank and connecting rod called a revolute joint or pin joint. The revolute joint has one degree-of-freedom in that if one element is fixed, the revolute joint allows the other only to rotate in a plane. A sphere joint has three degrees-of-freedom.

Lower and Higher Pairs

Connections between rigid bodies can be categorized as lower and higher pairs of elements. The two elements of a lower pair have theoretical surface contact with one another, while the two elements in the higher pair have theoretical point or line contact.
Lower pairs include revolute or pin connections.
higher pair include a pair of gears or a disk cam and a follower.

ANALYSIS

I. Position Analysis



1) When Crank Is Driver:

$$\vec{r}_B = \vec{r}_1 + \vec{r}_4 = \vec{r}_2 + \vec{r}_3$$

$$r_1(\cos\theta_1 i + \sin\theta_1 j) + r_4(\cos\theta_4 i + \sin\theta_4 j) = r_2(\cos\theta_2 i + \sin\theta_2 j) + r_3(\cos\theta_3 i + \sin\theta_3 j)$$

$$\Rightarrow r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2 - r_3\cos\theta_3 = 0$$

$$\Rightarrow r_1 s\,in\,\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2 - r_3\sin\theta_3 = 0$$

The base vector $\vec{r}_1$ *will be a constant.*

If the crank is the driver then θ_2 *will be given. We need to find other angles.*

$$r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2 = r_3\cos\theta_3 ---Eq(1)$$

$$r_1 s\,in\,\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2 = r_3\sin\theta_3 ----Eq(2)$$

$$r_3^{\,2} = r_1^{\,2} + r_2^{\,2} + r_4^{\,2} + 2r_1r_4(\cos\theta_1\cos\theta_4 + s\,in\,\theta_1\sin\theta_4)$$

$$-2r_1r_2(\cos\theta_1\cos\theta_2 + s\,in\,\theta_1\sin\theta_2) - 2r_2r_4(\cos\theta_2\cos\theta_4 + s\,in\,\theta_2\sin\theta_4)$$

$$A\cos\theta_4 + B\sin\theta_4 + C = 0$$

$$Here, A = 2r_1r_4\cos\theta_1 - 2r_2r_4\cos\theta_2$$

$$B = 2r_1r_4\sin\theta_1 - 2r_2r_4\sin\theta_2$$

$$C = r_1^{\,2} + r_2^{\,2} + r_4^{\,2} - r_3^{\,2} - 2r_1r_2(\cos\theta_1\cos\theta_2 + s\,in\,\theta_1\sin\theta_2)$$

$$A\cos\theta_4 + B\sin\theta_4 + C = 0$$

$$t = \tan\left(\frac{\theta_4}{2}\right); \sin\theta_4 = \left(\frac{2t}{1+t^2}\right); \cos\theta_4 = \left(\frac{1-t^2}{1+t^2}\right)$$

$$(C-A)t^2 + 2Bt + (A+C) = 0$$

On Solving this quadratic equation,

$$t = \frac{-2B \pm \sqrt{4B^2 - 4(C-A)(A+C)}}{2(C-A)}$$

$$t = \frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}$$

if $A^2 + B^2 < C^2 \Rightarrow$ *The mechanism open mechanism.*

$$\theta_4 = 2\tan^{-1}(t)$$

$$\theta_4 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_3 = \left(\frac{r_1 s\,in\,\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2}\right)$$

$$\theta_3 = \tan^{-1}\left(\frac{r_1 s\,in\,\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2}\right)$$

1) When Coupler Is Driver:

$$\vec{r}_B = \vec{r}_1 + \vec{r}_4 = \vec{r}_2 + \vec{r}_3$$

$$r_1(\cos\theta_1 i + \sin\theta_1 j) + r_4(\cos\theta_4 i + \sin\theta_4 j) = r_2(\cos\theta_2 i + \sin\theta_2 j) + r_3(\cos\theta_3 i + \sin\theta_3 j)$$

$$\Rightarrow r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2 - r_3\cos\theta_3 = 0$$

$$\Rightarrow r_1 \sin\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2 - r_3\sin\theta_3 = 0$$

The base vector $\vec{r}_1$ *will be a constant.*

If the coupler is the driver then θ_3 *will be given. We need to find other angles.*

$$r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3 = r_2\cos\theta_2 \ --- \ Eq(1)$$

$$r_1 \sin\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3 = r_2\sin\theta_2 \ ---- Eq(2)$$

$$r_2^{\,2} = r_1^{\,2} + r_3^{\,2} + r_4^{\,2} + 2r_1r_4(\cos\theta_1\cos\theta_4 + \sin\theta_1\sin\theta_4)$$

$$-2r_1r_3(\cos\theta_1\cos\theta_3 + \sin\theta_1\sin\theta_3) - 2r_3r_4(\cos\theta_3\cos\theta_4 + \sin\theta_3\sin\theta_4)$$

$$A\cos\theta_4 + B\sin\theta_4 + C = 0$$

$$A = 2r_1r_4\cos\theta_1 - 2r_3r_4\cos\theta_3$$

$$B = 2r_1r_4\sin\theta_1 - 2r_3r_4\sin\theta_3$$

$$C = r_1^{\,2} + r_3^{\,2} + r_4^{\,2} - r_2^{\,2} - 2r_1r_3(\cos\theta_1\cos\theta_3 + \sin\theta_1\sin\theta_3)$$

$$A\cos\theta_4 + B\sin\theta_4 + C = 0$$

$$t = \tan\left(\frac{\theta_4}{2}\right); \sin\theta_4 = \left(\frac{2t}{1+t^2}\right); \cos\theta_4 = \left(\frac{1-t^2}{1+t^2}\right)$$

$$(C-A)t^2 + 2Bt + (A+C) = 0$$

On Solving this quadratic equation,

$$t = \frac{-2B \pm \sqrt{4B^2 - 4(C-A)(A+C)}}{2(C-A)}$$

$$t = \frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}$$

if $A^2 + B^2 < C^2 \Rightarrow$ *The mechanism is open mechanism.*

$$\theta_4 = 2\tan^{-1}(t)$$

$$\theta_4 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_2 = \left(\frac{r_1 \sin\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{r_1 \sin\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

2) When Follower Is Driver:

$$\vec{r}_B = \vec{r}_1 + \vec{r}_4 = \vec{r}_2 + \vec{r}_3$$

$$r_1(\cos\theta_1 i + \sin\theta_1 j) + r_4(\cos\theta_4 i + \sin\theta_4 j) = r_2(\cos\theta_2 i + \sin\theta_2 j) + r_3(\cos\theta_3 i + \sin\theta_3 j)$$

$$\Rightarrow r_1\cos\theta_1 + r_4\cos\theta_4 - r_2\cos\theta_2 - r_3\cos\theta_3 = 0$$

$$\Rightarrow r_1 s\text{in}\,\theta_1 + r_4\sin\theta_4 - r_2\sin\theta_2 - r_3\sin\theta_3 = 0$$

The base vector $\vec{r}_1$ *will be a constant.*

If the rocker *is the driver then* θ_4 *will be given. We need to find other angles.*

$$r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3 = r_2\cos\theta_2 --- Eq(1)$$

$$r_1 s\text{in}\,\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3 = r_2\sin\theta_2 ---- Eq(2)$$

$$r_2{}^2 = r_1{}^2 + r_3{}^2 + r_4{}^2 + 2r_1r_4(\cos\theta_1\cos\theta_4 + s\text{in}\,\theta_1\sin\theta_4)$$

$$-2r_1r_3(\cos\theta_1\cos\theta_3 + s\text{in}\,\theta_1\sin\theta_3) - 2r_3r_4(\cos\theta_3\cos\theta_4 + s\text{in}\,\theta_3\sin\theta_4)$$

$$A\cos\theta_3 + B\sin\theta_3 + C = 0$$

$$Here, A = -2r_1r_3\cos\theta_1 - 2r_3r_4\cos\theta_4$$

$$B = -2r_1r_3\sin\theta_1 - 2r_3r_4\sin\theta_4$$

$$C = r_1{}^2 + r_3{}^2 + r_4{}^2 - r_2{}^2 + 2r_1r_4(\cos\theta_1\cos\theta_4 + s\text{in}\,\theta_1\sin\theta_4)$$

$$A\cos\theta_3 + B\sin\theta_3 + C = 0$$

$$t = \tan\left(\frac{\theta_3}{2}\right); \sin\theta_3 = \left(\frac{2t}{1+t^2}\right); \cos\theta_3 = \left(\frac{1-t^2}{1+t^2}\right)$$

$$(C-A)t^2 + 2Bt + (A+C) = 0$$

On Solving this quadratic equation,

$$t = \frac{-2B \pm \sqrt{4B^2 - 4(C-A)(A+C)}}{2(C-A)}$$

$$t = \frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}$$

if $A^2 + B^2 < C^2 \Rightarrow$ *The mechanism is open mechanism.*

$$\theta_3 = 2\tan^{-1}(t)$$

$$\theta_3 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C-A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_2 = \left(\frac{r_1 s\text{in}\,\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{r_1 s\text{in}\,\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

II) Angular Velocity

$\vec{r}_B = \vec{r}_1 + \vec{r}_4 = \vec{r}_2 + \vec{r}_3$

By diff. with respect to time

$\dot{\vec{r}}_1 + \dot{\vec{r}}_4 = \dot{\vec{r}}_2 + \dot{\vec{r}}_3$

As $\vec{r}_1$ is fixed link so it doesn't vary with time. $\vec{r}_1$ =0

$\dot{\vec{r}}_4 = \dot{\vec{r}}_2 + \dot{\vec{r}}_3$

For Angular Velocity:

Taking i terms:

$$r_4\dot{\theta}_4 \cos\theta_4 = r_2\dot{\theta}_2 \cos\theta_2 + r_3\dot{\theta}_3 \cos\theta_3$$

Taking j terms:

$$-r_4\dot{\theta}_4 \sin\theta_4 = -r_2\dot{\theta}_2 \sin\theta_2 - r_3\dot{\theta}_3 \sin\theta_3$$

When r_2 (crank) is driver:

$$\begin{bmatrix} r_4 \cos\theta_4 & -r_3 \cos\theta_3 \\ -r_4 \sin\theta_4 & r_3 \sin\theta_3 \end{bmatrix} \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} r_2\dot{\theta}_2 \cos\theta_2 \\ -r_2\dot{\theta}_2 \sin\theta_2 \end{bmatrix}$$

When r_3 (coupler) is driver:

$$\begin{bmatrix} r_4 \cos\theta_4 & -r_2 \cos\theta_2 \\ -r_4 \sin\theta_4 & r_2 \sin\theta_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} r_3\dot{\theta}_3 \cos\theta_3 \\ -r_3\dot{\theta}_3 \sin\theta_3 \end{bmatrix}$$

When r_4 (rocker) is driver:

$$\begin{bmatrix} r_2 \cos\theta_2 & r_3 \cos\theta_3 \\ -r_2 \sin\theta_2 & -r_3 \sin\theta_3 \end{bmatrix} \begin{bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} r_4\dot{\theta}_4 \cos\theta_4 \\ -r_4\dot{\theta}_4 \sin\theta_4 \end{bmatrix}$$

A x = b

x = inv(A)*b

III) Angular Acceleration

$$\ddot{\vec{r}}_4 = \ddot{\vec{r}}_2 + \ddot{\vec{r}}_3$$

For Angular Acceleration:

Taking i terms:

$$-r_4\dot{\theta}_4^2 \sin\theta_4 + r_4\ddot{\theta}_4 \cos\theta_4 = -r_2\dot{\theta}_2^2 \sin\theta_2 + r_2\ddot{\theta}_2 \cos\theta_2 - r_3\dot{\theta}_3^2 \sin\theta_3 + r_3\ddot{\theta}_3 \cos\theta_3$$

Taking j terms:

$$-r_4\ddot{\theta}_4 \sin\theta_4 - r_4\dot{\theta}_4^2 \cos\theta_4 = -r_2\dot{\theta}_2^2 \cos\theta_2 - r_2\ddot{\theta}_2 \sin\theta_2 - r_3\dot{\theta}_3^2 \cos\theta_3 - r_3\ddot{\theta}_3 \sin\theta_3$$

When r_2 (crank) is driver:

$$\begin{bmatrix} r_4 \cos\theta_4 & -r_3 \cos\theta_3 \\ -r_4 \sin\theta_4 & r_3 \sin\theta_3 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_4 \\ \ddot{\theta}_3 \end{bmatrix} = \begin{bmatrix} r_4\dot{\theta}_4^2 \sin\theta_4 - r_2\dot{\theta}_2^2 \sin\theta_2 + r_2\ddot{\theta}_2 \cos\theta_2 - r_3\dot{\theta}_3^2 \sin\theta_3 \\ r_4\dot{\theta}_4^2 \cos\theta_4 - r_2\dot{\theta}_2^2 \cos\theta_2 - r_2\ddot{\theta}_2 \sin\theta_2 - r_3\dot{\theta}_3^2 \cos\theta_3 \end{bmatrix}$$

When r_3 (coupler) is driver:

$$\begin{bmatrix} r_4 \cos\theta_4 & -r_2 \cos\theta_2 \\ -r_4 \sin\theta_4 & r_2 \sin\theta_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_4 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} r_4\dot{\theta}_4^2 \sin\theta_4 - r_3\dot{\theta}_3^2 \sin\theta_3 + r_3\ddot{\theta}_3 \cos\theta_3 - r_2\dot{\theta}_2^2 \sin\theta_2 \\ r_4\dot{\theta}_4^2 \cos\theta_4 - r_3\dot{\theta}_3^2 \cos\theta_2 - r_3\ddot{\theta}_3 \sin\theta_3 - r_2\dot{\theta}_2^2 \cos\theta_2 \end{bmatrix}$$

When r_4 (rocker) is driver:

$$\begin{bmatrix} r_2 \cos\theta_2 & r_3 \cos\theta_3 \\ -r_2 \sin\theta_2 & -r_3 \sin\theta_3 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = \begin{bmatrix} -r_4\dot{\theta}_4^2 \sin\theta_4 + r_4\ddot{\theta}_4 \cos\theta_4 + r_2\dot{\theta}_2^2 \cos\theta_2 + r_3\dot{\theta}_3^2 \cos\theta_3 \\ -r_4\ddot{\theta}_4 \sin\theta_4 - r_4\dot{\theta}_4^2 \cos\theta_4 + r_2\dot{\theta}_2^2 \cos\theta_2 + r_3\dot{\theta}_3^2 \cos\theta_3 \end{bmatrix}$$

A x = b

x = inv(A)*b

DETAILED IMPLEMENTATION OF MATLAB CODE

Crank is the Driver Link:

```
function [o4_1, o4_2,o3_1,o3_2,o4v_1,o3v_1,o4v_2,o3v_2,o4a,o3a]=crank(r1,r2,r3,r4,o1,o2, o2v,o2a
o=pi/180;
A=(2*r1*r4*cos((pi/180)*o1))-(2*r2*r4*cos((pi/180)*o2));
B=(2*r1*r4*sin((pi/180)*o1))-(2*r2*r4*sin((pi/180)*o2));
C=(r1^2)+(r2^2)-(r3^2)+(r4^2)-(2*r1*r2*((cos((pi/180)*(o1-o2)))));

X1=(-B+sqrt((A^2)+(B^2)-(C^2)))/(C-A);
X2=(-B-sqrt((A^2)+(B^2)-(C^2)))/(C-A);

 % Below  for getting the value of theta 4 there will be 2 values that is
 % denoted as o4_1  o4_2
o4_1=(180/pi)*2*atan(X1);
o4_2=(180/pi)*2*atan(X2);

U1=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4_1))-(r2*sin((pi/180)*o2));
U2=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4_2))-(r2*sin((pi/180)*o2));
D1=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4_1))-(r2*cos((pi/180)*o2));
D2=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4_2))-(r2*cos((pi/180)*o2));

 % Below  for getting the value of theta 3 .
 %as theta 4 is of 2 values there will be 2 values that is denoted as o3_1  o3_2
o3_1=(180/pi)*atan(U1/D1);
o3_2=(180/pi)*atan(U2/D2);

Av1=[r4*sin((pi/180)*o4_1) -r3*sin((pi/180)*o3_1);r4*cos((pi/180)*o4_1) -r3*cos((pi/180)*o3_1)];

Av2=[r4*sin((pi/180)*o4_2) -r3*sin((pi/180)*o3_2);r4*cos((pi/180)*o4_2) -r3*cos((pi/180)*o3_2)];

Bv=[r2*sin((pi/180)*o2)*o2v ;r2*cos((pi/180)*o2)*o2v];

X1=inv(Av1)*Bv;
X2=inv(Av2)*Bv;

%Here I  have denoted the corresponding values to my own considerations
o4v_1=X1(1); % Link 4 velocity with one value of theta 4,3
o3v_1=X1(2); % Link 3 velocity with one value of theta 4,3
o4v_2=X2(1);% Link 4 velocity with another value of theta 4,3
o3v_2=X2(2);% Link 3 velocity with another value of theta 4,3

Aa=[r4*cos(o*o4_1) -r3*cos(o*o3_1);-r4*sin(o*o4_1) r3*sin(o*o3_1)];
Ba1=[r4*(o4v_1^2)*sin(o*o4_1)-r2*(o2v^2)*sin(o*o2)-r3*(o3v_1^2)*sin(o*o3_1)+r2*(o2a)*cos(o*o2)];
Ba2=[r4*(o4v_1^2)*cos(o*o4_1)-r2*(o2v^2)*cos(o*o2)-r3*(o3v_1^2)*cos(o*o3_1)-r2*(o2a)*sin(o*o2)];

Ba=[Ba1;Ba2];

X=inv(Aa)*Ba;

o4a=X(1);
o3a=X(2);
end
```

This is this function to give the value of theta 3(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity, angular acceleration.

Input Values: r1,r2,r3,r4,theta1,theta2,theta2 angular velocity, theta2 angular acceleration

Output Values: theta 3,theta 4, theta 3 angular velocity, theta 4 angular velocity ,theta 3 angular acceleration ,theta 4 angular acceleration

Coupler is the Driver Link:

```
function[o4_1, o4_2,o2_1,o2_2,o4v_1,o2v_1,o4v_2,o2v_2,o4a,o2a]=coupler(r1,r2,r3,r4,o1,o3, o3v,o3a)
o=pi/180;
A=(2*r1*r4*cos((pi/180)*o1))-(2*r3*r4*cos((pi/180)*o3));
B=(2*r1*r4*sin((pi/180)*o1))-(2*r3*r4*sin((pi/180)*o3));
C=(r1^2)-(r2^2)+(r3^2)+(r4^2)-(2*r1*r3*((cos((pi/180)*(o1-o3)))));

X1=(-B+sqrt((A^2)+(B^2)-(C^2)))/(C-A);
X2=(-B-sqrt((A^2)+(B^2)-(C^2)))/(C-A);

% Below  for getting the value of theta 4 there will be 2 values that is
 % denoted as o4_1  o4_2
o4_1=(180/pi)*2*atan(X1);
o4_2=(180/pi)*2*atan(X2);


U1=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4_1))-(r3*sin((pi/180)*o3));
U2=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4_2))-(r3*sin((pi/180)*o3));
D1=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4_1))-(r3*cos((pi/180)*o3));
D2=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4_2))-(r3*cos((pi/180)*o3));

% Below  for getting the value of theta 2 .
 %as theta 4 is of 2 values there will be 2 values that is denoted as o2_1  o2_2
o2_1=(180/pi)*atan(U1/D1);
o2_2=(180/pi)*atan(U2/D2);

%This is the AX=B form for getting the velocity Av1 is A matrix with one
%value of theta 4 and theta 2
% Av2 is A matrix with another value of theta 4 and theta 2
%Bv is corresponding B matrix and we need to find X which consist of link
%2,4 velocity
Av1=[r4*sin((pi/180)*o4_1) -r2*sin((pi/180)*o2_1);r4*cos((pi/180)*o4_1) -r2*cos((pi/180)*o2_1)];
Av2=[r4*sin((pi/180)*o4_2) -r2*sin((pi/180)*o2_2);r4*cos((pi/180)*o4_2) -r2*cos((pi/180)*o2_2)];
Bv=[r3*sin((pi/180)*o3)*o3v ;r3*cos((pi/180)*o3)*o3v];

X1=inv(Av1)*Bv;
X2=inv(Av2)*Bv;

%Here I  have denoted the corresponding values to my own considerations
o4v_1=X1(1); %Link 4 velocity with one value of theta 4,2
o2v_1=X1(2); % Link 2 velocity with one value of theta 4,2
o4v_2=X2(1);% Link 4 velocity with another value of theta 4,2
o2v_2=X2(2); % Link 2 velocity with another value of theta 4,2

Aa=[r4*cos(o*o4_1) -r2*cos(o*o2_1);-r4*sin(o*o4_1) r2*sin(o*o2_1)];
Ba1=[r4*(o4v_1^2)*sin(o*o4_1)-r3*(o3v^2)*sin(o*o3)-r2*(o2v_1^2)*sin(o*o2_1)+r3*(o3a)*cos(o*o3)];
Ba2=[r4*(o4v_1^2)*cos(o*o4_1)-r3*(o3v^2)*cos(o*o3)-r2*(o2v_1^2)*cos(o*o2_1)-r3*(o3a)*sin(o*o3)];

Ba=[Ba1;Ba2];

X=inv(Aa)*Ba;

o4a=X(1);
o2a=X(2);
end
```

This is this function to give the value of theta 2(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity, angular acceleration.

Input Values: r1,r2,r3 ,r4,theta1,theta3,theta3 angular velocity, theta3 angular acceleration

Output Values: theta 2,theta 4, theta 2 angular velocity, theta 4 angular velocity , theta 2 angular acceleration ,theta 4 angular acceleration

Follower is the Driver Link:

```
function [o2_1, o2_2,o3_1,o3_2,o2v_1,o3v_1,o2v_2,o3v_2,o2a,o3a]=follower(r1,r2,r3,r4,o1,o4, o4v,o4a)
o=pi/180;
A=(-2*r1*r3*cos((pi/180)*o1))-(2*r3*r4*cos((pi/180)*o4));
B=(-2*r1*r3*sin((pi/180)*o1))-(2*r3*r4*sin((pi/180)*o4));
C=(r1^2)-(r2^2)+(r3^2)+(r4^2)+(2*r1*r4*((cos((pi/180)*(o1-o4)))));

X1=(-B+sqrt((A^2)+(B^2)-(C^2)))/(C-A);
X2=(-B-sqrt((A^2)+(B^2)-(C^2)))/(C-A);

 % Below  for getting the value of theta 4 there will be 2 values that is
 % denoted as o4_1  o4_2
o3_1=(180/pi)*2*atan(X1);
o3_2=(180/pi)*2*atan(X2);

U1=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4))-(r3*sin((pi/180)*o3_1));
U2=(r1*sin((pi/180)*o1))+(r4*sin((pi/180)*o4))-(r3*sin((pi/180)*o3_2));
D1=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4))-(r3*cos((pi/180)*o3_1));
D2=(r1*cos((pi/180)*o1))+(r4*cos((pi/180)*o4))-(r3*cos((pi/180)*o3_2));

 % Below  for getting the value of theta 3 .
 %as theta 4 is of 2 values there will be 2 values that is denoted as o3_1  o3_2
o2_1=(180/pi)*atan(U1/D1);
o2_2=(180/pi)*atan(U2/D2);

Av1=[r2*sin((pi/180)*o2_1) -r3*sin((pi/180)*o3_1);r2*cos((pi/180)*o2_1) -r3*cos((pi/180)*o3_1)];

Av2=[r2*sin((pi/180)*o2_2) -r3*sin((pi/180)*o3_2);r2*cos((pi/180)*o2_2) -r3*cos((pi/180)*o3_2)];

Bv=[r4*sin((pi/180)*o4)*o4v ;r4*cos((pi/180)*o4)*o4v];

X1=inv(Av1)*Bv;
X2=inv(Av2)*Bv;

%Here I  have denoted the corresponding values to my own considerations
o2v_1=X1(1); % Link 4 velocity with one value of theta 4,3
o3v_1=X1(2); % Link 3 velocity with one value of theta 4,3
o2v_2=X2(1);% Link 4 velocity with another value of theta 4,3
o3v_2=X2(2);% Link 3 velocity with another value of theta 4,3

Aa=[r2*cos(o*o2_1) -r3*cos(o*o3_1);-r2*sin(o*o2_1) r3*sin(o*o3_1)];
Ba1=[r2*(o2v_1^2)*sin(o*o2_1)-r4*(o4v^2)*sin(o*o4)-r3*(o3v_1^2)*sin(o*o3_1)+r2*(o4a)*cos(o*o4)];
Ba2=[r2*(o2v_1^2)*cos(o*o2_1)-r4*(o4v^2)*cos(o*o4)-r3*(o3v_1^2)*cos(o*o3_1)-r2*(o4a)*sin(o*o4)];

Ba=[Ba1;Ba2];

X=inv(Aa)*Ba;

o2a=X(1);
o3a=X(2);
end
```

This is this function to give the value of theta 2(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity, angular acceleration.

Input Values: r1,r2,r3,r4,theta1,theta4,theta4 angular velocity , theta 4 angular acceleration

Output Values: theta 2,theta 3, theta 2 angular velocity, theta 3 angular velocity, theta 2 angular acceleration ,theta 3 angular acceleration

Range:

```
function [ i ,i1 ,i2]=range(r1,r2,r3)
l=[r1 r2 r3];

disp("r4 is longest ")
i=(sum(l)-min(l))-min(l);
disp("r4 should be greater than:  "+i)
disp("r4 is shortest")
i1=(sum(l)-max(l))-max(l);
if(i1<=0)

    disp("Not valid for negative value:  "+i1)
else
    disp("r4 should be less than:  "+i1)
end

disp("r4 as medium length")
i2=(max(l)+min(l))-(sum(l)-(max(l)+min(l)));
disp("r4 should be less than:  "+i2)

disp("r4"+">"+i);
disp("r4"+"<"+i2);

if(i1<=0)

    disp("Not valid for negative value:  "+i1)
else
    disp("r4"+">"+i1);
end

end
```

This is this function to give the range for r4(length of follower)

Input Values: r1,r2,r3

Output Values: Prints the range for values

Pseudo Code:

```
Input: length of links, driver link
If(conditions for length of  last link )
        If((s+l>p+q) & l<s+p+q)
                If(driver link(crank))
                        -------------------------------------------(1)
                        Animation, angular velocity and acceleration
                Else if(driver link(coupler))
                        -------------------------------------------(2)
                        Animation, angular velocity and acceleration
                Else if(driver link(follower))
                        -------------------------------------------(3)
                        Animation, angular velocity and acceleration
        else("non grashoff")
else ("not in range")
```

Explanation of Main Code:

Our Basic representation:

r1 : Fixed Link ; r2 : Crank Link ; r3 : Coupler Link ; r4 = Follower Link

o=pi/180 to convert degree to radian

o1=theta 1(angle b/w x-axis and fixed link)

o2=theta 2(angle b/w x-axis and crank)

o3=theta 3(angle b/w x-axis and coupler)

o4=theta 4(angle b/w x-axis and follower)

o2v= angular velocity of crank ; o3v= angular velocity of coupler

o4v= angular velocity of follower.

o2a= angular acceleration of crank ; o3a= angular acceleration of coupler

o4a= angular acceleration of follower.

```
r1=input("Length of r1:  ");     % fixed
r2=input("Length of r2:  ");       %crank
r3=input("Length of r3:  ");       %coupler
```

Here we are taking the input of links length (fixed link, crank, coupler).

```
y='Type your Driver is \n     1.crank \n     2.coupler \n     3. follower \n';
 x=lower(input(y,'s'));
```

Here we are taking the input which is gonna be the driver link.

```
 [i ,i1 ,i2]=range(r1,r2,r3);
r4=input("length of r4 :   ");
```

With the help of range function, we can decide the length of r4.

At (1): If driver link is crank

```
        if (x=="crank")
            MaxT=180;
            y=r3^2+r4^2-2*r3*r4*cos(MaxT*o) ;
            y=(1/o)*acos((r2^2+r1^2-y)/(2*r1*r2)) ;
            disp("theta 2(angle b/w fixed link and crank ) range  is"+(-y)+"<="+"theta 2"+"<="+(y) )
            y=input("Enter the range(y) (-y <= theta 2 <=y) :   ");
            o2=-y:y;
            o2v=input("theta 2 angular velocity:  ");
            o2a=input("theta 2 angular acceleration:  ");
```

For Triple Rocker Mechanism the maximum transmission angle is 180 so with respect to we find the range of theta 2 (angle b/w fixed link and crank). And here we give the input for o2v, o2a.

```
for i=1:length(o2)

[o4_1, o4_2,o3_1,o3_2,o4v_1,o3v_1,o4v_2,o3v_2,o4a,o3a]=crank(r1,r2,r3,r4,o1,o2(i), o2v,o2a);
A=[ 0 0];
B=[ r2*cos(o*o2(i)) r2*sin(o*o2(i))];
C1=[ r2*cos(o*o2(i))+r3*cos(o*o3_1) r2*sin(o*o2(i))+r3*sin(o*o3_1)];
C2=[r1+r4*cos(o*o4_1) r4*sin(o*o4_1)];
D=[r1 0];

 plot([real(A(1)) real(B(1))],[real(A(2)) real(B(2))])
 hold on
 plot([real(B(1)) real(C2(1))] ,[real(B(2)) real(C2(2))],'ro--')
 hold on
 plot([real(C2(1)) real(D(1))],[real(C2(2)) real(D(2))])
 hold on
 plot([0 real(D(1))],[0 real(D(2))],'ro--')
 hold off
 axis([-10 20 -10 10])
pause(0.01)
end
```

Here we call the crank function, with the varying of theta 2 we will get the values of theta 3,4 and angular velocities too. So, we will get a movement of mechanism.

```
  t=linspace(1,length(o2)/50,length(o2));

 for i=1:length(o2)
 [o4_1(i), o4_2,o3_1(i),o3_2,o4v_1(i),o3v_1(i),o4v_2,o3v_2,o4a(i),o3a(i)]=crank(r1,r2,r3,r4,o1,o2(i), o2v,o2a);
 end
 figure
subplot(2,1,1);
plot(t,real(o4v_1));
title("theta 4 angular velocity wrt to time")
ylabel("theta 4 angular velocity")
xlabel("time")
subplot(2,1,2);
plot(t,real(o3v_1))
 title("theta 3 angular velocity wrt to time")
ylabel("theta 3 angular velocity")
xlabel("time")
   figure
subplot(2,1,1);
plot(t,real(o4a));
title("theta 4 angular acceleration wrt to time")
ylabel("theta 4 angular acceleration")
xlabel("time")
subplot(2,1,2);
plot(t,real(o3a))
 title("theta 3 angular acceleration wrt to time")
ylabel("theta 3 angular acceleration")
xlabel("time")
```

This is for get the velocities values for particular values of theta 2 and angular velocities of coupler and follower. And we get plots for and angular velocities , angular acceleration of coupler and follower w.r.t to time.

At (2): If driver link is coupler

```
elseif (x=="coupler")
      o3=0:100;  o3v=input("theta 3 angular velocity:  ");o3a=input("theta 3 angular acceleration:  ");
    for i=1:length(o3)

    [o4_1, o4_2,o2_1,o2_2,o4v_1,o2v_1,o4v_2,o2v_2,o4a,o2a]=coupler(r1,r2,r3,r4,o1,o3(i), o3v,o3a)

    A=[ 0 0];
    B=[ r2*cos(o*o2_2) r2*sin(o*o2_2)];
    C1=[ r2*cos(o*o2_2)+r3*cos(o*o3(i)) r2*sin(o*o2_2)+r3*sin(o*o3(i))];
    C2=[r1+r4*cos(o*o4_1) r4*sin(o*o4_1)];
    D=[r1 0];

    r2c=sqrt((A(1)-B(1))^2 +((A(2)-B(2))^2))<=r2;
    r3c1=sqrt((B(1)-C1(1))^2 +((B(2)-C1(2))^2))<=r3;
    r3c2=sqrt((B(1)-C2(1))^2 +((B(2)-C2(2))^2))<=r3;
    r4c2=sqrt((D(1)-C2(1))^2 +((D(2)-C2(2))^2))<=r4;
    r4c1=sqrt((D(1)-C1(1))^2 +((D(2)-C1(2))^2))<=r4;


    if(r2c&&r3c1&&r3c2&&r4c2&&r4c1)
      plot([real(A(1)) real(B(1))],[real(A(2)) real(B(2))])
      hold on
      plot([real(B(1)) real(C1(1))] ,[real(B(2)) real(C1(2))],'ro--')
      hold on
      plot([real(C1(1)) real(D(1))],[real(C1(2)) real(D(2))])
      hold on
      plot([0 real(D(1))],[0 real(D(2))],'ro--')
      hold off
      axis([-10 20 -10 10])
     pause(0.1)
    end
     end
```

Here we call the crank function, movement of coupler is from 1:100 degrees we will get the values of theta 2,4 and angular velocities too. Here the lengths are fixed means the animations will only execute when the length are equal. And here we give the input for o3v, o3a.

```
  t=linspace(1,length(o3)/20,length(o3));A
  for i=1:length(o3)
  [o4_1(i), o4_2,o2_1(i),o2_2,o4v_1(i),o2v_1(i),o4v_2,o3v_2,o4a(i),o2a(i)]=coupler(r1,r2,r3,r4,o1,o3(i), o3v,o3a);
  end
  figure

 subplot(2,1,1);
 plot(t,real(o4v_1))
 title("theta 4 angular velocity wrt to time")
 ylabel("theta 4 angular velocity")
 xlabel("time")
 subplot(2,1,2);
 plot(t,real(o2v_1))
  title("theta 2 angular velocity wrt to time")
 ylabel("theta 2 angular velocity")
 xlabel("time")
     figure
 subplot(2,1,1);
 plot(t,real(o4a));
 title("theta 4 angular acceleration wrt to time")
 ylabel("theta 4 angular acceleration")
 xlabel("time")
 subplot(2,1,2);
 plot(t,real(o2a))
  title("theta 2 angular acceleration wrt to time")
 ylabel("theta 2 angular acceleration")
 xlabel("time")
```

This is for get the velocities values for particular values of theta 3 and angular velocities of crank and follower. And we get plots for and angular velocities, angular acceleration of crank and follower w.r.t to time.

At (3): If driver link is follower

```
 elseif (x=="follower")
       o4=0:360;  o4v=input("theta 4 angular velocity:  ");o4a=input("theta 4 angular acceleration:  ");

         for i=1:length(o4)

          [o2_1, o2_2,o3_1,o3_2,o2v_1,o3v_1,o2v_2,o3v_2,o2a,o3a]=follower(r1,r2,r3,r4,o1,o4(i), o4v,o4a);

         A=[ 0 0];
         B=[ r2*cos(o*o2_1) r2*sin(o*o2_1)];
         C1=[ r2*cos(o*o2_1)+r3*cos(o*o3_1) r2*sin(o*o2_1)+r3*sin(o*o3_1)];
         C2=[r1+r4*cos(o*o4(i)) r4*sin(o*o4(i))];
         D=[r1 0];

         r2c=sqrt((A(1)-B(1))^2 +((A(2)-B(2))^2))<=r2;
         r3c1=sqrt((B(1)-C1(1))^2 +((B(2)-C1(2))^2))<=r3;
         r3c2=sqrt((B(1)-C2(1))^2 +((B(2)-C2(2))^2))==r3;
         r4c2=sqrt((D(1)-C2(1))^2 +((D(2)-C2(2))^2))<=r4;
         r4c1=sqrt((D(1)-C1(1))^2 +((D(2)-C1(2))^2))<=r4;
         if(r2c&&r3c1&&r3c2&&r4c2&&r4c1)
          plot([real(A(1)) real(B(1))],[real(A(2)) real(B(2))])
          hold on
          plot([real(B(1)) real(C2(1))] ,[real(B(2)) real(C2(2))],'ro--')
          hold on
          plot([real(C2(1)) real(D(1))],[real(C2(2)) real(D(2))])
          hold on
          plot([0 real(D(1))],[0 real(D(2))],'ro--')
          hold off
          axis([-10 20 -10 10])
         pause(0.1)

         end

         end
```

Here we call the follower function, we will get the values of theta 2,3 and angular velocities too. Here the lengths are fixed means the animations will only execute when the length are equal.

```
t=linspace(1,length(o4)/20,length(o4));
 for i=1:length(o4)
[o2_1, o2_2,o3_1,o3_2,o2v_1(i),o3v_1(i),o2v_2(i),o3v_2(i),o2a(i),o3a(i)]=follower(r1,r2,r3,r4,o1,o4(i), o4v,o4a
 end
    figure
   subplot(2,1,1);
plot(t,real(o3v_1))
title("theta 3 angular velocity wrt to time")
ylabel("theta 3 angular velocity")
xlabel("time")
subplot(2,1,2);
plot(t,real(o2v_1))
 title("theta 2 angular velocity wrt to time")
ylabel("theta 2 angular velocity")
xlabel("time")
   figure
subplot(2,1,1);
plot(t,real(o3a));
title("theta 3 angular acceleration wrt to time")
ylabel("theta 3 angular acceleration")
xlabel("time")
subplot(2,1,2);
plot(t,real(o2a))
 title("theta 2 angular acceleration wrt to time")
ylabel("theta 2 angular acceleration")
xlabel("time")
```

This is for get the velocities values for particular values of theta 4 and angular velocities of crank and coupler. And we get plots for and angular velocities , angular accelerations of crank and follower w.r.t to time.

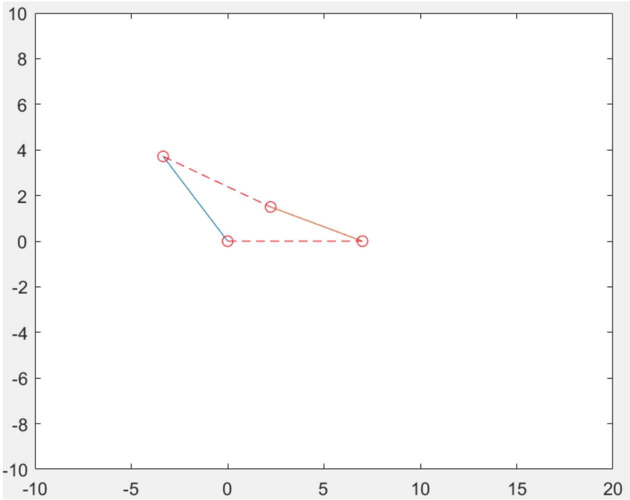
Example of Output:

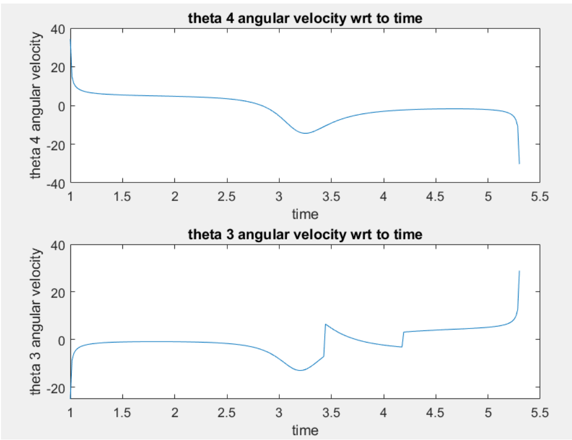
When Crank is the driver:

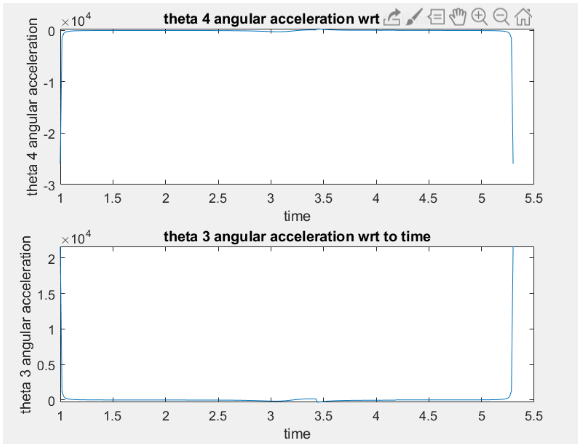
In command window:

```
Length of r1:  7
Length of r2:  5
Length of r3:  6
Type your Driver is
      1.crank
     2.coupler
     3. follower
crank
r4 is longest
r4 should be greater than:  8
r4 is shortest
r4 should be less than:  4
r4 as medium length
r4 should be less than:  6
r4>8
r4<6
r4>4
length of r4 :   5
can execte
theta 2(angle b/w fixed link and crank ) range  is-132.1774<=theta 2<=132.1774
Enter the range(y) (-y <= theta 2 <=y) :   132
theta 2 angular velocity:  5
theta 2 angular acceleration:  3
```

Plots







DETAILED IMPLEMENTATION OF PYTHON CODE:

Files we are importing

```
import numpy as np
import numpy.linalg as lin
import matplotlib.pyplot as plt
```

```
def crank():
    theta1=float(input("Enter \u03B8\u2081 : "))
    theta2=float(input("Enter \u03B8\u2082 : "))
    #converting theta values to radians
    theta1=theta1*np.pi/180
    theta2=theta2*np.pi/180

    #initialising variables
    theta3=np.zeros(2)
    theta4=np.zeros(2)

    t3dot=np.zeros(2)
    t4dot=np.zeros(2)

    t3ddot=np.zeros(2)
    t4ddot=np.zeros(2)

    #position analysis
    a=2*r1*r4*np.cos(theta1)-2*r2*r4*np.cos(theta2)
    b=2*r1*r4*np.sin(theta1)-2*r2*r4*np.sin(theta2)
    c=r1**2+r2**2+r4**2-r3**2-2*r1*r2*(np.cos(theta1)*np.cos(theta2)+np.sin(theta1)*np.sin(theta2))

    t41=2*np.arctan((-b+(a*a+b*b-c*c)**0.5)/(c-a))
    t42=2*np.arctan((-b-(a*a+b*b-c*c)**0.5)/(c-a))

    t31=np.arctan((r1*np.sin(theta1)+r4*np.sin(t41)-r2*np.sin(theta2))/(r1*np.cos(theta1)+r4*np.cos(t41)-r2*np.cos(theta2)))
    t32=np.arctan((r1*np.sin(theta1)+r4*np.sin(t42)-r2*np.sin(theta2))/(r1*np.cos(theta1)+r4*np.cos(t42)-r2*np.cos(theta2)))

    theta3[0]=t31;theta3[1]=t32
    theta4[0]=t41;theta4[1]=t42
    #print(theta3)
    #print(theta4)

    t2dot=float(input("enter \u03B8\u2082 dot : "))
    t2dot=t2dot*np.pi/180
    #for 1st value of theta3 and theta4
    A=np.array([[r4*np.cos(t41),-r3*np.cos(t31)],[-r4*np.sin(t41),r3*np.sin(t31)]])
    B=np.array([[r2*t2dot*np.cos(theta2)],[-r2*t2dot*np.sin(theta2)]])
    X=lin.solve(A,B)
    t4dot[0]=X[0,0]
    t3dot[0]=X[1,0]
```

```
#for 2st value of theta3 and theta4
A=np.array([[r4*np.cos(t42),-r3*np.cos(t32)],[-r4*np.sin(t42),r3*np.sin(t32)]])
X=lin.solve(A,B)
t4dot[1]=X[0,0]
t3dot[1]=X[1,0]

#print(t3dot,t4dot)

#Acceleration analysis
t2ddot=float(input("enter \u03B8\u2082 double dot : "))
t2ddot=t2ddot*np.pi/180

#first value of theta3 and theta4
A=np.array([[r4*np.cos(t41),-r3*np.cos(t31)],[-r4*np.sin(t41),r3*np.sin(t31)]])
B=np.array([[r4*t4dot[0]**2*np.sin(t41) - r2*t2dot**2*np.sin(theta2) + r2*t2ddot*np.cos(theta2)  - r3*t3dot[0]**2*np.sin(t31)],
[r4*t4ddot[0]**2*np.cos(t41) - r2*t2dot**2*np.cos(theta2) - r2*t2ddot*np.sin(theta2)  - r3*t3dot[0]**2*np.cos(t31)]])
X=lin.solve(A,B)

t4ddot[0]=X[0,0]
t3ddot[0]=X[1,0]

#second value of theta3 and theta 4
A=np.array([[r4*np.cos(t42),-r3*np.cos(t32)],[-r4*np.sin(t42),r3*np.sin(t32)]])
B=np.array([[r4*t4dot[1]**2*np.sin(t42) - r2*t2dot**2*np.sin(theta2) + r2*t2ddot*np.cos(theta2)  - r3*t3dot[1]**2*np.sin(t32)],
[r4*t4ddot[1]**2*np.cos(t42) - r2*t2dot**2*np.cos(theta2) - r2*t2ddot*np.sin(theta2)  - r3*t3dot[1]**2*np.cos(t32)]])
X=lin.solve(A,B)

t4ddot[1]=X[0,0]
t3ddot[1]=X[1,0]

#print(t3ddot,t4ddot)
listkey=["theta1","theta2","theta3","theta4","theta3dot","theta4dot","theta3ddot","theta4ddot"]
dct=dict(zip(listkey,[[theta1]*2,[theta2]*2,theta3,theta4,t3dot,t4dot,t3ddot,t4ddot]))


#show
print("\n\u0307\u03F4\u2083= {} or {}".format(t3dot[0],t3dot[1]))
print("\u0307\u03F4\u2084= {} or {}".format(t4dot[0],t4dot[1]))
print("\u0308\u03F4\u2083= {} or {}".format(t3ddot[0],t3ddot[1]))
print("\u0308\u03F4\u2084= {} or {}".format(t4ddot[0],t4ddot[1]))

return dct;
```

This is this function to give the value of theta 3(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity

$$\theta_4 = 2\tan^{-1}(t)$$

$$\theta_4 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C - A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_3 = \left(\frac{r_1 s\,\text{in}\,\theta_1 + r_4 \sin\theta_4 - r_2 \sin\theta_2}{r_1 \cos\theta_1 + r_4 \cos\theta_4 - r_2 \cos\theta_2}\right)$$

$$\theta_3 = \tan^{-1}\left(\frac{r_1 s\,\text{in}\,\theta_1 + r_4 \sin\theta_4 - r_2 \sin\theta_2}{r_1 \cos\theta_1 + r_4 \cos\theta_4 - r_2 \cos\theta_2}\right)$$

As we get two values for theta3 and theta4 so there will be two plots

Input Values: r1,r2,r3,r4,theta1,theta2,theta2 angular velocity, theta2 angular acceleration.

Output Values: it returns a dictionary which contains theta 3, theta 4, theta 3 angular velocity, theta4 angular velocity, theta 3 angular acceleration, theta4 angular acceleration and plots for the values of theta3 and theta4.

```python
def coupler():
    theta1=float(input("Enter \u03B8\u2081 : "))
    theta3=float(input("Enter \u03B8\u2083 : "))
    #converting theta values to radians
    theta1=theta1*np.pi/180
    theta3=theta3*np.pi/180

    #initialising variables
    theta2=np.zeros(2)
    theta4=np.zeros(2)

    t2dot=np.zeros(2)
    t4dot=np.zeros(2)

    t2ddot=np.zeros(2)
    t4ddot=np.zeros(2)

    #position analysis
    a=2*r1*r4*np.cos(theta1)-2*r2*r4*np.cos(theta3)
    b=2*r1*r4*np.sin(theta1)-2*r2*r4*np.sin(theta3)
    c=r1**2+r3**2+r4**2-r2**2-2*r1*r3*(np.cos(theta1)*np.cos(theta3)+np.sin(theta1)*np.sin(theta3))

    t41=2*np.arctan((-b+(a*a+b*b-c*c)**0.5)/(c-a))
    t42=2*np.arctan((-b-(a*a+b*b-c*c)**0.5)/(c-a))

    t21=np.arctan((r1*np.sin(theta1)+r4*np.sin(t41)-r3*np.sin(theta3))/(r1*np.cos(theta1)+r4*np.cos(t41)-r3*np.cos(theta3)))
    t22=np.arctan((r1*np.sin(theta1)+r4*np.sin(t42)-r2*np.sin(theta3))/(r1*np.cos(theta1)+r4*np.cos(t42)-r2*np.cos(theta3)))

    theta2[0]=t21;theta2[1]=t22
    theta4[0]=t41;theta4[1]=t42


    t3dot=float(input("enter \u03B8\u2083 dot : "))
    t3dot=t3dot*np.pi/180
    #for 1st value of theta3 and theta4
    A=np.array([[r4*np.cos(t41),-r2*np.cos(t21)],[-r4*np.sin(t41),r2*np.sin(t21)]])
    B=np.array([[r3*t3dot*np.cos(theta3)],[-r3*t3dot*np.sin(theta3)]])
    X=lin.solve(A,B)
    t4dot[0]=X[0,0]
    t2dot[0]=X[1,0]
```

```python
#for 2st value of theta3 and theta4
A=np.array([[r4*np.cos(t42),-r2*np.cos(t22)],[-r4*np.sin(t42),r2*np.sin(t22)]])
X=lin.solve(A,B)

t4dot[1]=X[0,0]
t2dot[1]=X[1,0]

#print(t3dot,t4dot)

#Acceleration analysis
t3ddot=float(input("enter \u03B8\u2083 double dot : "))
t3ddot=t3ddot*np.pi/180

#first value of theta3 and theta4
A=np.array([[r4*np.cos(t41),-r2*np.cos(t21)],[-r4*np.sin(t41),r2*np.sin(t21)]])
B=np.array([[r4*t4dot[0]**2*np.sin(t41) - r3*t3dot**2*np.sin(theta3) + r3*t3ddot*np.cos(theta3)  - r2*t2dot[0]**2*np.sin(t21)],
[r4*t4dot[0]**2*np.cos(t41) - r3*t3dot**2*np.cos(theta3) - r3*t3ddot*np.sin(theta3)  - r2*t2dot[0]**2*np.cos(t21)]])
X=lin.solve(A,B)

t4ddot[0]=X[0,0]
t2ddot[0]=X[1,0]

#second value of theta3 and theta 4
A=np.array([[r4*np.cos(t42),-r2*np.cos(t22)],[-r4*np.sin(t42),r2*np.sin(t22)]])
B=np.array([[r4*t4dot[1]**2*np.sin(t42) - r3*t3dot**2*np.sin(theta3) + r3*t3ddot*np.cos(theta3)  - r2*t2dot[1]**2*np.sin(t22)],
[r4*t4dot[1]**2*np.cos(t42) - r3*t3dot**2*np.cos(theta3) - r3*t3ddot*np.sin(theta3)  - r2*t2dot[1]**2*np.cos(t22)]])
X=lin.solve(A,B)

t4ddot[1]=X[0,0]
t2ddot[1]=X[1,0]

#print(t3ddot,t4ddot)
listkey=["theta1","theta2","theta3","theta4","theta2dot","theta4dot","theta2ddot","theta4ddot"]
dct=dict(zip(listkey,[[theta1]*2,theta2,[theta3]*2,theta4,t2dot,t4dot,t2ddot,t4ddot]))

#show
print("\n\u0307\u03F4\u2082= {} or {}".format(t2dot[0],t2dot[1]))
print("\u0307\u03F4\u2084= {} or {}".format(t4dot[0],t4dot[1]))
print("\u0308\u03F4\u2082= {} or {}".format(t2ddot[0],t2ddot[1]))
print("\u0308\u03F4\u2084= {} or {}".format(t4ddot[0],t4ddot[1]))

return dct;
```

This is this function to give the value of theta 2(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity

$$\theta_4 = 2\tan^{-1}(t)$$

$$\theta_4 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2+B^2-C^2}}{(C-A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_2 = \left(\frac{r_1 \sin\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{r_1 \sin\theta_1 + r_4\sin\theta_4 - r_3\sin\theta_3}{r_1\cos\theta_1 + r_4\cos\theta_4 - r_3\cos\theta_3}\right)$$

Input Values: r1,r2,r3 ,r4,theta1,theta3,theta3 angular velocity, theta3 angular acceleration.

Output Values: theta 2,theta 4, theta 2 angular velocity, theta 4 angular velocity, theta 2 angular acceleration, theta1 angular acceleration and plots for the values of theta2 and theta4

```
def rocker():
    theta1=float(input("Enter \u03B8\u2081 : "))
    theta4=float(input("Enter \u03B8\u2084 : "))
    #converting theta values to radians
    theta1=theta1*np.pi/180
    theta4=theta4*np.pi/180

    #print(theta1,theta4)

    #initialising variables
    theta2=np.zeros(2)
    theta3=np.zeros(2)

    t2dot=np.zeros(2)
    t3dot=np.zeros(2)

    t2ddot=np.zeros(2)
    t3ddot=np.zeros(2)

    a=-2*r1*r3*np.cos(theta1)-2*r3*r4*np.cos(theta4)
    b=-2*r1*r3*np.sin(theta1)-2*r3*r4*np.sin(theta4)
    c=r1**2+r3**2+r4**2-r2**2+2*r1*r4*(np.cos(theta1)*np.cos(theta4)+np.sin(theta1)*np.sin(theta4))

    t31=2*np.arctan((-b+(a*a+b*b-c*c)**0.5)/(c-a))
    t32=2*np.arctan((-b-(a*a+b*b-c*c)**0.5)/(c-a))

    t21=np.arctan((r1*np.sin(theta1)+r4*np.sin(theta4)-r3*np.sin(t31))/(r1*np.cos(theta1)+r4*np.cos(theta4)-r3*np.cos(t31)))
    t22=np.arctan((r1*np.sin(theta1)+r4*np.sin(theta4)-r3*np.sin(t32))/(r1*np.cos(theta1)+r4*np.cos(theta4)-r3*np.cos(t32)))

    theta2[0]=t21;theta2[1]=t22
    theta3[0]=t31;theta3[1]=t32
    #print(theta2,theta3)
    #print((-b+(a*a+b*b-c*c)**0.5)/(c-a))
```

```
t4dot=float(input("enter \u03B8\u2084 dot : "))
t4dot=t4dot*np.pi/180
#for 1st value of theta3 and theta4
A=np.array([[r2*np.cos(t21),r3*np.cos(t31)],[-r2*np.sin(t21),-r3*np.sin(t31)]])
B=np.array([[r4*t4dot*np.cos(theta4)],[-r4*t4dot*np.sin(theta4)]])
X=lin.solve(A,B)

t2dot[0]=X[0,0]
t3dot[0]=X[1,0]

#for 2st value of theta3 and theta4
A=np.array([[r2*np.cos(t22),r3*np.cos(t32)],[-r2*np.sin(t22),-r3*np.sin(t32)]])
X=lin.solve(A,B)

t2dot[1]=X[0,0]
t3dot[1]=X[1,0]

#print(t3dot,t4dot)

#Acceleration analysis
t4ddot=float(input("enter \u03B8\u2084 double dot : "))
t4ddot=t4ddot*np.pi/180

#first value of theta3 and theta4
A=np.array([[r2*np.cos(t21),r3*np.cos(t31)],[-r2*np.cos(t21),-r3*np.sin(t31)]])
B=np.array([[-r4*t4dot**2*np.sin(theta4) + r4*t4ddot*np.cos(theta4) + r2*t2dot[0]**2*np.cos(t21)  - r3*t3dot[0]**2*np.cos(t31)],
[-r4*t4ddot*np.sin(theta4) - r4*t4dot**2*np.cos(theta4) + r2*t2dot[0]**2*np.cos(t21)  + r3*t3dot[0]**2*np.cos(t31)]])
X=lin.solve(A,B)

t2ddot[0]=X[0,0]
t3ddot[0]=X[1,0]

#second value of theta3 and theta 4
A=np.array([[r2*np.cos(t22),r3*np.cos(t32)],[-r2*np.cos(t22),-r3*np.sin(t32)]])
B=np.array([[-r4*t4dot**2*np.sin(theta4) + r4*t4ddot*np.cos(theta4) + r2*t2dot[1]**2*np.cos(t22)  - r3*t3dot[1]**2*np.cos(t32)],
[-r4*t4ddot*np.sin(theta4) - r4*t4dot**2*np.cos(theta4) + r2*t2dot[1]**2*np.cos(t22)  + r3*t3dot[1]**2*np.cos(t32)]])
X=lin.solve(A,B)

t2ddot[1]=X[0,0]
t3ddot[1]=X[1,0]
```

```
#print(t3ddot,t4ddot)
listkey=["theta1","theta2","theta3","theta4","theta2dot","theta3dot","theta2dot","theta3ddot"]
dct=dict(zip(listkey,[[theta1]*2,theta2,theta3,[theta4]*2,t2dot,t3dot,t2ddot,t3ddot]))

#show
print("\n\u0307\u03F4\u2082= {} or {}".format(t2dot[0],t2dot[1]))
print("\u0307\u03F4\u2083= {} or {}".format(t3dot[0],t3dot[1]))
print("\u0308\u03F4\u2082= {} or {}".format(t2ddot[0],t2ddot[1]))
print("\u0308\u03F4\u2083= {} or {}".format(t3ddot[0],t3ddot[1]))

return dct;
```

This is this function to give the value of theta 2(angle formed by coupler with x-axis), theta 4(angle formed by follower with x-axis) and its angular velocity

$$\theta_3 = 2\tan^{-1}(t)$$

$$\theta_3 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{A^2 + B^2 - C^2}}{(C - A)}\right)$$

$$\frac{Eq2}{Eq1} = tan\,\theta_2 = \left(\frac{r_1 s\,\text{in}\,\theta_1 + r_4 \sin\theta_4 - r_3 \sin\theta_3}{r_1 \cos\theta_1 + r_4 \cos\theta_4 - r_3 \cos\theta_3}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{r_1 s\,\text{in}\,\theta_1 + r_4 \sin\theta_4 - r_3 \sin\theta_3}{r_1 \cos\theta_1 + r_4 \cos\theta_4 - r_3 \cos\theta_3}\right)$$

Input Values: r1,r2,r3,r4,theta1,theta4,theta4 angular velocity,theta4 angular acceleration.

Output Values: theta 2,theta 3, theta 2 angular velocity, theta 3 angular velocity, , theta 3 angular acceleration, theta2 angular acceleration and plots for the values of theta3 and theta2.

Pseudo Code:

Input: length of links, driver link

If(conditions for length of last link)

    If(s+l>p+q)

        If(driver link(1))

            ---(crank)

            Output graph, Angular velocity and acceleration

        Else if(driver link(2))

            ---(coupler)

            Output graph, Angular velocity and acceleration

        Else if(driver link(3))

            ---(follower)

            Output graph, Angular velocity and acceleration

    else("it is not a in valid entry")

else ("the given configuration is not a triple rocker mechanism")

Explanation of Main Code:

Our Basic representation:

r1 : Fixed Link ; r2 : Crank Link ; r3 : Coupler Link ; r4 = Follower Link

$\theta_1 =$ theta 1(angle b/w x-axis and fixed link)

$\theta_2 =$ theta 2(angle b/w x-axis and crank)

$\theta_3 =$ theta 3(angle b/w x-axis and coupler)

$\theta_4 =$ theta 4(angle b/w x-axis and follower)

```
r1=float(input("enter length of r1 : "))
r2=float(input("enter length of r2 : "))
r3=float(input("enter length of r3 : "))
r4=float(input("enter length of r4 : "))
l=[r1,r2,r3,r4]
l.sort()
```

Here we are taking the input of links length (fixed link, crank, coupler).

We are storing it in array. Then we are sorting it.

```
if l[0]+l[3]>l[1]+l[2]:

    print("\nEnter the given numbers to choose the following case:")
    choice=int(input("1- crank as driver\n2- coupler as driver\n3- rocker as driver\n"))

    if choice == 1:
        dct=crank()
        pic(dct)
    elif choice == 2:
        dct=coupler()
        pic(dct)
    elif choice == 3:
        dct=rocker()
        pic(dct)
    else:
        print("{} is not a valid entry".format(choice))
else:
    print("The given configuration is not a triple rocker mechanism")
```

This is explained in our pseudocode.

Simulation:

Python:

```
import numpy as np
import numpy.linalg as lin
import matplotlib.pyplot as plt


from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from celluloid import Camera
```

```python
def crank(theta1,theta2,t2dot,t2ddot):
    theta1
    theta2
    t2dot
    #converting theta values to radians
    theta1=theta1*np.pi/180
    theta2=theta2*np.pi/180
    t2dot=t2dot*np.pi/180
    t2ddot=t2ddot*np.pi/180


    #position analysis
    a=2*r1*r4*np.cos(theta1)-2*r2*r4*np.cos(theta2)
    b=2*r1*r4*np.sin(theta1)-2*r2*r4*np.sin(theta2)
    c=r1**2+r2**2+r4**2-r3**2-2*r1*r2*(np.cos(theta1)*np.cos(theta2)+np.sin(theta1)*np.sin(theta2))

    theta4=2*np.arctan((-b+(a*a+b*b-c*c)**0.5)/(c-a))


    theta3=np.arctan((r1*np.sin(theta1)+r4*np.sin(theta4)-r2*np.sin(theta2))/(r1*np.cos(theta1)+r4*np.cos(theta4)-r2*np.cos(theta


    #calculating angular velocity
    A=np.array([[r4*np.cos(theta4),-r3*np.cos(theta3)],[-r4*np.sin(theta4),r3*np.sin(theta3)]])
    B=np.array([[r2*t2dot*np.cos(theta2)],[-r2*t2dot*np.sin(theta2)]])
    X=lin.solve(A,B)
    t4dot=X[0,0]
    t3dot=X[1,0]

    #calculating angular acceleration
    A=np.array([[r4*np.cos(theta4),-r3*np.cos(theta3)],[-r4*np.sin(theta4),r3*np.sin(theta3)]])
    B=np.array([[r4*t4dot**2*np.sin(theta4) - r2*t2dot**2*np.sin(theta2) + r2*t2ddot*np.cos(theta2)  - r3*t3dot**2*np.sin(theta3)
    [r4*t4dot**2*np.cos(theta4) - r2*t2dot**2*np.cos(theta2) - r2*t2ddot*np.sin(theta2)  - r3*t3dot**2*np.cos(theta3)]])
    X=lin.solve(A,B)

    t4ddot=X[0,0]
    t3ddot=X[1,0]



    dct=dict(zip(["theta1","theta2","theta3","theta4","t2dot","t3dot","t4dot","t2ddot","t3ddot","t4ddot"],
               [theta1,theta2,theta3,theta4,t2dot,t3dot,t4dot,t2ddot,t3ddot,t4ddot]))
    return dct
```

Crank- based on the values of theta1, theta2, t2dot and t2ddot, the following values are calculated: theta3, theta4, theta3dot, theta4dot, theta3ddot, theta4ddot. The input values and the calculated values are then stored in dictionary and returned.

```python
def animation(dct):
    t=np.linspace(0,10,100)
    delt=t[1]-t[0]

    fig,axes = plt.subplots(4,constrained_layout=True,figsize=(15,15))


    camera = Camera(fig)

    global a2;
    a2=np.array([]);
    for ang in np.linspace(-np.pi/2,np.pi,500):
        temp=crank(dct["theta1"]*180/np.pi,ang*180/np.pi,dct["t2dot"]*180/np.pi,dct["t2ddot"]*180/np.pi)
        ax=0;ay=0
        bx=r1*np.cos(temp["theta1"]);by=r1*np.sin(temp["theta1"])
        dx=r2*np.cos(ang);dy=r2*np.sin(ang)
        cx=bx+r4*np.cos(temp["theta4"]);cy=by+r4*np.sin(temp["theta4"])
        if(round(abs(d([dx,cx],[dy,cy])),2)-r3<0.1):
            a2=np.append(a2,ang)

    #angle2=np.arange(min(a2),max(a2),delt*dct["t2dot"]+0.5*dct["t2ddot"]*delt**2)
    global omega
    ang=min(a2)
    angle2=[]
    w=dct["t2dot"]
    omega=[w]
    while(ang<max(a2)):
        angle2.append(ang)
        ang=ang+delt*w+0.5*dct["t2ddot"]*delt**2
        w=w+dct["t2ddot"]*delt
        omega.append(w)

    angle2=np.array(angle2)
    omega=np.array(omega)
    print("angle2:{}".format(angle2))
    print("omega2:{}".format(omega))
    #thetaf=thetai*t+0.5alpha*t^2
```

```python
global tet2,tet3,tet4,a4,v2,v3,v4
tet2=tet3=tet4=a2=a3=a4=v2=v3=v4=t1=np.array([])

angle2=np.concatenate((angle2,np.flip(angle2)),axis=0)
omega=np.concatenate((omega,np.flip(omega)),axis=0)
for i,ang in enumerate(angle2):

    print(omega[i]*180/3.14)

    dct=crank(dct["theta1"]*180/np.pi,ang*180/np.pi,omega[i]*180/np.pi,dct["t2ddot"]*180/np.pi)
    ax=0;ay=0
    bx=r1*np.cos(dct["theta1"]);by=r1*np.sin(dct["theta1"])
    dx=r2*np.cos(dct["theta2"]);dy=r2*np.sin(dct["theta2"])
    cx=bx+r4*np.cos(dct["theta4"]);cy=by+r4*np.sin(dct["theta4"])

    axes[0].plot([ax,bx],[ay,by],'b')
    axes[0].plot([ax,dx],[ay,dy],'b')
    axes[0].plot([dx,cx],[dy,cy],'b')
    axes[0].plot([bx,cx],[by,cy],'b')


    #axes[0].set_xlim([-10, 10])
   #6 axes[0].set_ylim([-5,5])

    tet2=np.append(tet2,dct["theta2"])
    tet3=np.append(tet3,dct["theta3"])
    tet4=np.append(tet4,dct["theta4"])
    v2=np.append(v2,omega[i])
    v3=np.append(v3,dct["t3dot"])
    v4=np.append(v4,dct["t4dot"])
    a2=np.append(a2,dct["t2ddot"])
    a3=np.append(a3,dct["t3ddot"])
    a4=np.append(a4,dct["t4ddot"])
    t1=np.append(t1,i*delt)

    axes[1].plot(t1,tet2*180/np.pi,'r');
    axes[1].plot(t1,tet3*180/np.pi,'b');
    axes[1].plot(t1,tet4*180/np.pi,'g');

    axes[2].plot(t1,v2*180/np.pi,'r');
    axes[2].plot(t1,v3*180/np.pi,'b');
    axes[2].plot(t1,v4*180/np.pi,'g');

    axes[3].plot(t1,a2*180/np.pi,'r');
    axes[3].plot(t1,a3*180/np.pi,'b');
    axes[3].plot(t1,a4*180/np.pi,'g');

    camera.snap()
```

```python
axes[1].legend(["\u03B8\u2082 ","\u03B8\u2083 ","\u03B8\u2084 "],prop={'size': 10})
axes[2].legend(["\u03B8\u2082 dot","\u03B8\u2083 dot","\u03B8\u2084 dot"],prop={'size': 10})
axes[3].legend(["\u03B8\u2082 ddot","\u03B8\u2083 ddot","\u03B8\u2084 ddot"],prop={'size': 10})

axes[1].set_title("\u03B8 vs time")
axes[2].set_title("\u03C9 vs time")
axes[3].set_title("\u03B1 vs time")
'''
axes[1].set_ylabel('\u03C9',fontsize=14)
axes[1].set_xlabel('Time',fontsize=14)

axes[2].set_ylabel('\u03B1',fontsize=14)
axes[2].set_xlabel('Time',fontsize=14)
'''
#axes[1].set_ylim([-180,180])
animation = camera.animate()
animation.save("animation.gif",writer='imagemagick')
```

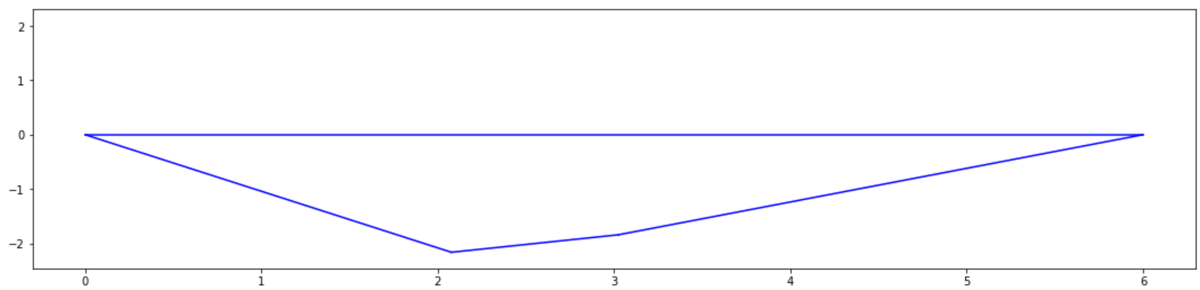
Animates- animates the triple rocker mechanism.

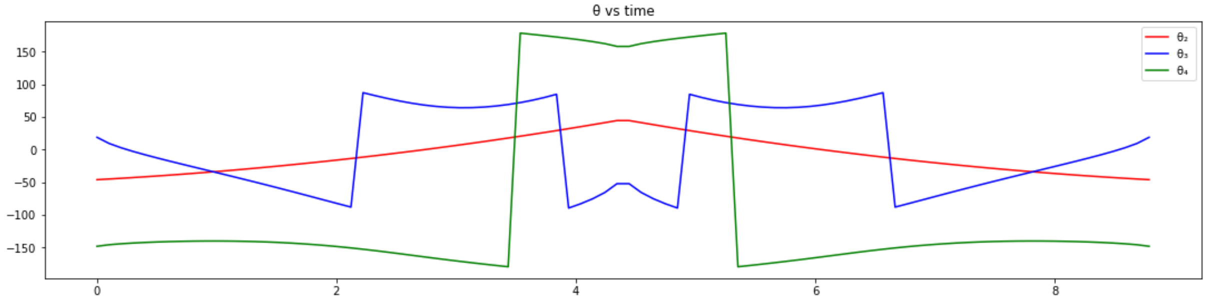
```
def d(x,y):
    return ((x[0]-x[1])**2+(y[0]-y[1])**2)**0.5


r1=float(input("enter length of r1 : "))
r2=float(input("enter length of r2 : "))
r3=float(input("enter length of r3 : "))
#theta1=0;theta2=0;theta3=0;theta4=0;
t1dot=0;t2dot=0;t3dot=0;t4dot=0;
l=[r1,r2,r3]
l.sort()
a=l[0]+l[1]-l[2]
b=-l[0]+l[1]+l[2]
ulimit=l[1]+l[2]+l[0]
lowLimit=a
if(lowLimit<0):
    lowLimit=(a+b)/2
r4=float(input(("enter length of r4 in range {}<r4<{} :").format(lowLimit,ulimit)))
if (r4>lowLimit and r4<ulimit):



    theta1=float(input("Enter \u03B8\u2081 : "))
    theta2=float(input("Enter \u03B8\u2082 : "))
    t2dot=float(input("enter \u03B8\u2082 dot : "))
    t2ddot=float(input("enter \u03B8\u2082 double dot : "))
    dct=crank(theta1,theta2,t2dot,t2ddot)
    animation(dct)


else:
    print("not in range")
```

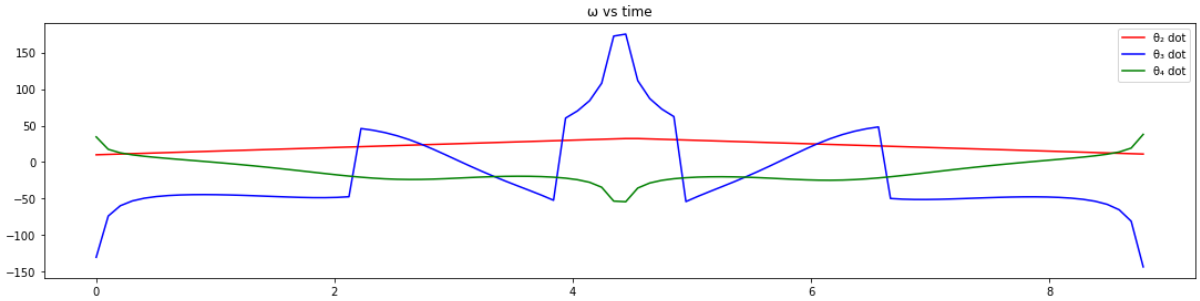
d-calculated the Euclidean distance between two points.

Firstly, the maximum and minimum value of theta2 in computed. We iterate the value of theta2 from minimum to maximum. At each iteration, we update the variables through crank method, a subplot is created based on the theta values and theta2 value is incremented as per t2dot and t2ddot. The theta values, angular velocity values as well as the angular acceleration values is plotted at each iteration as subplots. Finally, we use celluloid library to convert the series of plots to a gif file.

OUTPUT:







Transmission angle:

L_0=Fixed link

L_1=Crank

L_2=Coupler

L_3= Follower

Formula for min transmission angle:

$$\cos(\theta_{\min}) = \frac{R2^2 + R3^2 - (R0-1)^2}{2(R3)(R2)}$$

Equation Based on Grashoff's Criterion:

$$R_0 = \frac{L_0}{L_1} = \alpha \quad R_2 = \frac{L_2}{L_1} \quad R_3 = \frac{L_3}{L_1}$$

By keeping L0 as Longest:

$$L_0 + L_1 > L_2 + L_3$$
$$(\alpha)L_1 + L_1 > L_2 + L_3$$
$$(\alpha + 1)L_1 > L_2 + L_3$$
$$\text{Dividing with } L_1$$
$$R_2 + R_3 < (\alpha + 1)$$

By keeping L2 as Longest:

$$L_2 + L_1 > L_0 + L_3$$
$$L_2 - L_3 > 4L_1 - L_1$$
$$(\alpha - 1)L_1 < L_2 - L_3$$
$$\text{Dividing with } L_1$$
$$R_2 - R_3 > (\alpha - 1)$$

By keeping L3 as Longest:

$$L_3 + L_1 > L_0 + L_2$$
$$L_3 - L_2 > (\alpha)L_1 - L_1$$
$$(\alpha - 1)L_1 < L_3 - L_2$$
$$\text{Dividing with } L_1$$
$$R_3 - R_2 > (\alpha - 1)$$

Making a quadratic equation in terms of R2, R3 and $\theta_{\min}$

$$R2^2 + R3^2 - (R0 - 1)^2 - 2(R2)(R3)\cos(\theta_{\min}) = 0$$

$$R2^2 - 2(R2)(R3)\cos(\theta_{\min}) + R3^2 - (R0 - 1)^2 = 0$$

$$ax^2 + bx + c = 0$$
$$x = \frac{(-b) \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x = R2$$
$$a = 1$$
$$b = -2(R3)\cos(\theta_{\min})$$
$$c = R3^2 - (R0-1)^2$$

For Finding maximum Transmission Angle:

$$(\theta_{\max}) = \cos^{-1}\left(\frac{R2^2 + R3^2 - (R0+1)^2}{2(R2)(R3)}\right)$$

Explanation of Main Code:

```
disp("All lengths should be in same SI unit")
L0=input("What is fixed length:  ");
L1=input("What is crank length:  ");
L2=input("What is coupler length:  ");
L3=input("What is rocker length:  ");

l=[L0 L1 L2 L3];
```

Taking input of Links and arranging it in a array.

```
if(max(l)<(sum(l)-max(l)))&&(max(l)+min(l)>sum(l)-(max(l)+min(l)))
alpha=L0/L1 % Given Ratio L0/L1
R0=L0/L1; %Ratio L0/L1
R2=L2/L1; %Ratio L2/L1
R3=L3/L1;  %Ratio L3/L1
MinAngle=(180/pi)*acos(((R2^2)+(R3^2)-((R0-1)^2))/(2*R3*R2)) %minimum
MaxAnglei=(180/pi)*acos(((R2^2)+(R3^2)-((R0+1)^2))/(2*R3*R2))
R3=0:0.1:10; % Variying length of R4 (r4/r2)

x=0:0.1:10; % Assigning x as R4
```

Here we are checking for the condition **s+l>p+q** and **l<s+p+q**

Then making it into Ratios and finding alpha, and with the lengths we are finding minimum and maximum transmission angle. And by keeping the ratio b/w Follower and crank as varying and done the plot.

```
for i=1:length(R3) %For changing length
y1=(alpha+1)-x; % Grashof's criterion R2+R3=5 (R2+R3<alpha+1)
y2=(alpha-1)+x; % Grashof's criterion R2-R3=3 (R2-R3>alpha+1)
y3=x-(alpha-1); % Grashof's criterion R3-R2=3 (R3-R2>alpha+1)

a=1;
b=-2*R3(i)*cos((pi/180)*MinAngle);
c=(R3(i)^2)-((R0-1)^2);

R2(i)=(-b+sqrt((b^2)-4*a*c))/(2*a); % R2 finding with minimum transmission ar


MaxAngle(i)=(180/pi)*acos(((R2(i)^2)+(R3(i)^2)-((R0+1)^2))/(2*R2(i)*R3(i)));

end
```

By using the above formulas we are computing the R2 w.r.t R3.

```
plot(R3,R2)
hold on
plot(x,y1)
hold on
plot(x,y2)
hold on
plot(x,y3)
grid
title("Min Transmission angle is  "+MinAngle+"  Max Transmission angle is  "+MaxAnglei)
xlabel("Ratio R3(L3/L1)")
ylabel("Ratio R4(L4/L1)")

axis([0 8 0 8])
```

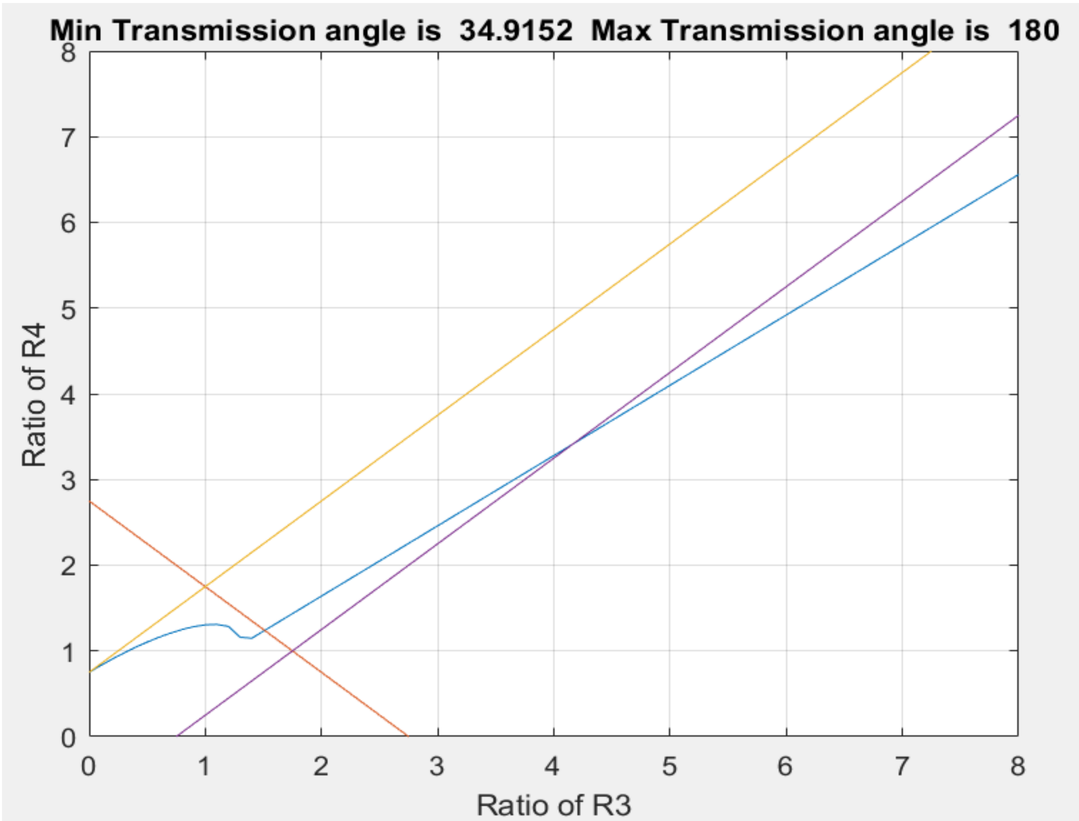
This is for plotting the boundary lines and for (R2,R3).

OUTPUT:

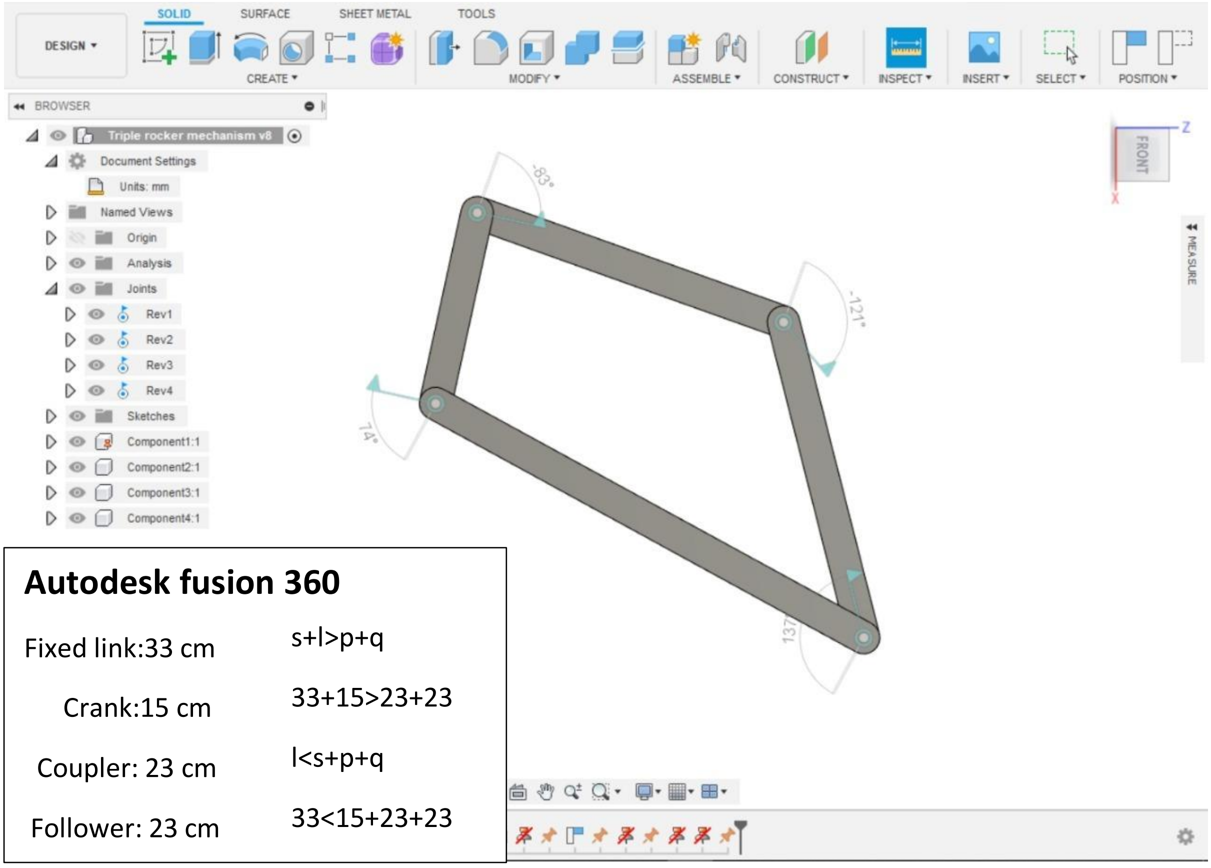
In Command Window:

```
All lengths should be in same SI unit
What is fixed length:  7
What is crank length:  4
What is coupler length:  5
What is rocker length:  5

alpha =

          1.75


MinAngle =

         34.92


MaxAnglei =

        180.00
```

Plot:

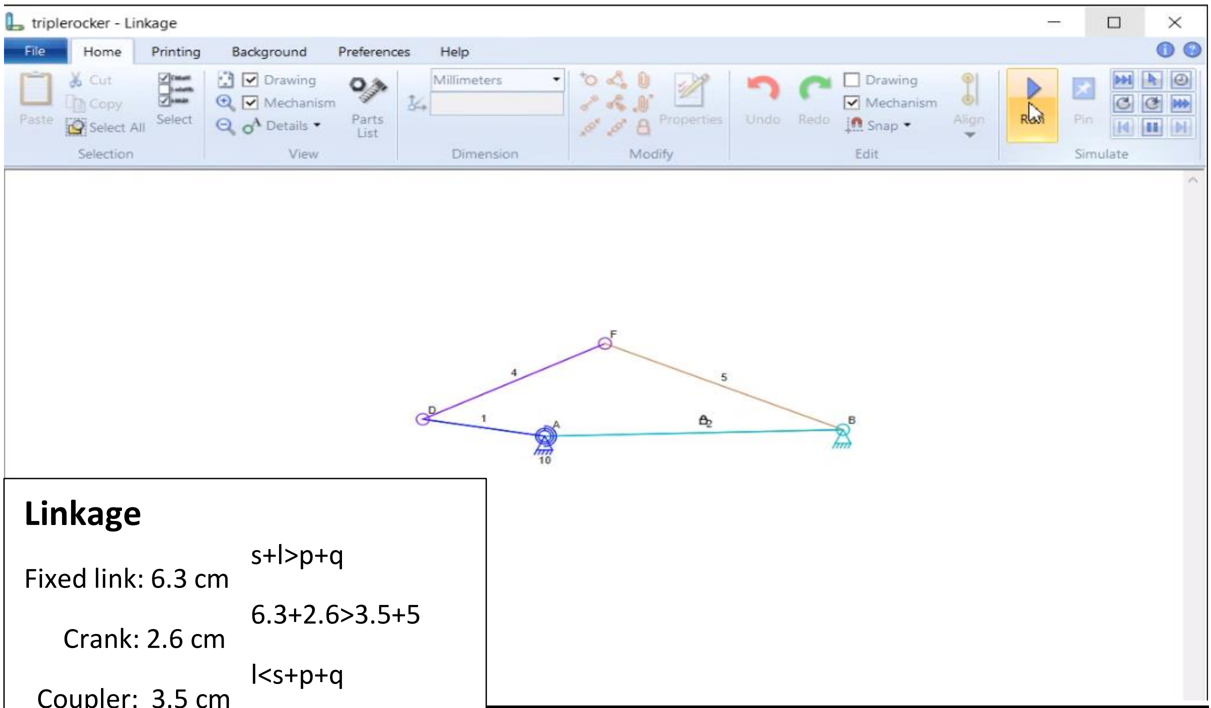


IMPLEMENTATION IN VARIOUS SOFTWARES:



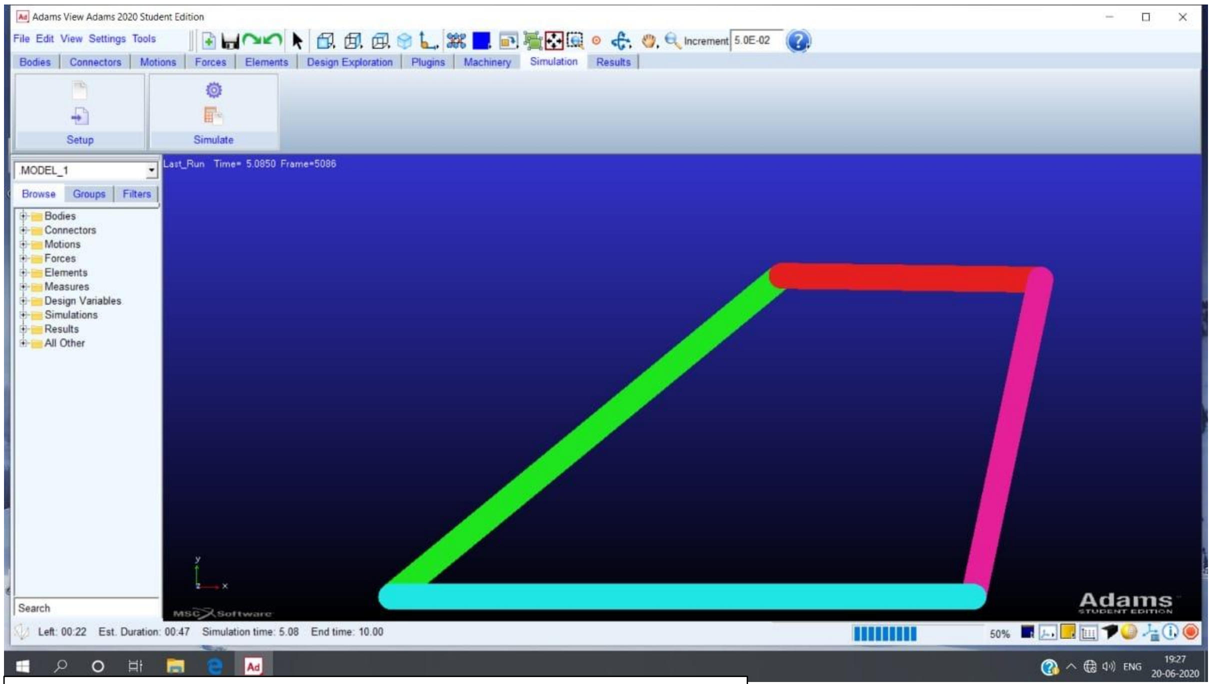
Autodesk fusion 360

Fixed link:33 cm	s+l>p+q
Crank:15 cm	33+15>23+23
Coupler: 23 cm	l<s+p+q
Follower: 23 cm	33<15+23+23



Linkage

Fixed link: 6.3 cm	s+l>p+q
Crank: 2.6 cm	6.3+2.6>3.5+5
Coupler: 3.5 cm	l<s+p+q
Follower: 5 cm	6.3<2.6+3.5+5



ADAMS

Fixed link: 900

Crank: 780

Coupler: 400

Follower: 500

s+l>p+q

900+400>500+780

l<s+p+q

900<400+500+500

WORK BREAKDOWN

CB.EN.U4AIE19003 : P.ADHITHAN -**MATLAB** simulation: of Four bar mechanism, angular velocity and acceleration w.r.t time graph , Transmission Angle Graph for applicable lengths , Report Making

CB.EN.U4AIE19018 : Ch. SAI HARSHA REDDY –Analysis for angular velocity and acceleration formulas for Report, PPT making

CB.EN.U4AIE19025 :S.DINESH - Analysis for Position for Report ,Formatting the report, Implementation in **Linkage** ,Report Making

CB.EN.U4AIE19033 : N.KABILAN - Report making : Formatting the report , Implementation in **Autodesk fusion 360** , Report Making , PPT making

CB.EN.U4AIE19061 : M.A.C.SIVA MARAN - **PYTHON** Stimulation: of Four bar mechanism , position, angular velocity and acceleration w.r.t time graph and analysis.

CB.EN.U4AIE19068 : S.VIJAY SIMMON -Analysis for transmission angle graphs for a given triple rocker mechanism for report, Implementation in **ADAMS** , PPT making

CONCLUSION:

The purpose of the project was to develop an understanding of triple rocker mechanism. For understanding of the mechanism, we have done position, velocity and acceleration analysis and also transmission angle graphs for the mechanism in MATLAB. To provide a visual representation of our mechanism we have done the simulation in various Software such as Linkage, Autodesk fusion 360, ADAMS. Additionally, we have simulated the triple rocker mechanism in python and MATLAB, including plots with the variation of angular velocity and acceleration with respect to time.