**ME 232: Kinematics and dynamics of Machines**

**Term Project**

**Yash Gupta(22b2151)**

**Pranab Baro(22b2224)**

A team of two students will have to select a mechanism and prepare a report about it. The different sections of the report and their contents are being described below. Students must follow this format (sections numbering and titles) and not merge the sections into a single long description. Division of these sections into subsections is acceptable.

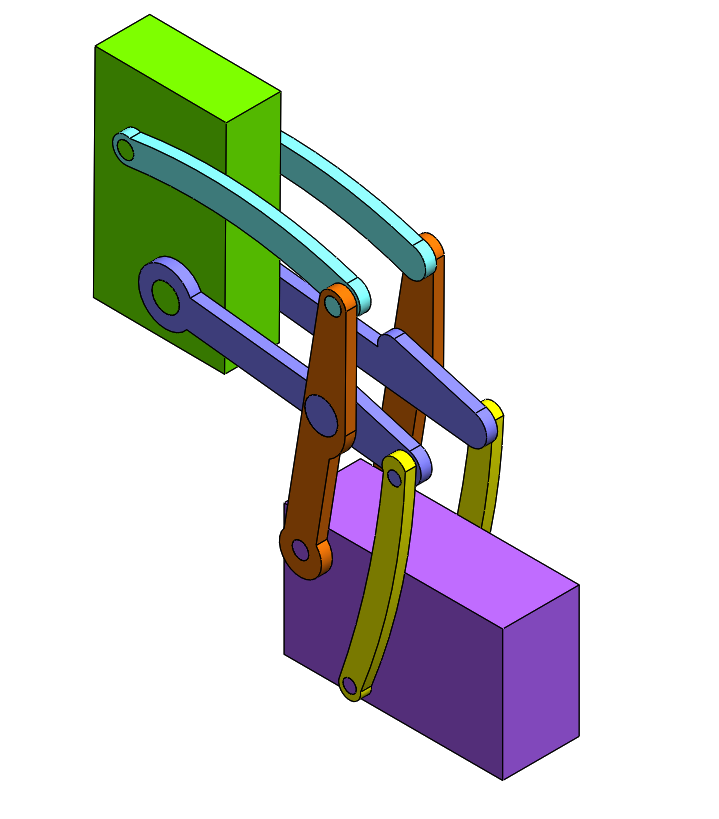
# Section 1: Mechanism Description

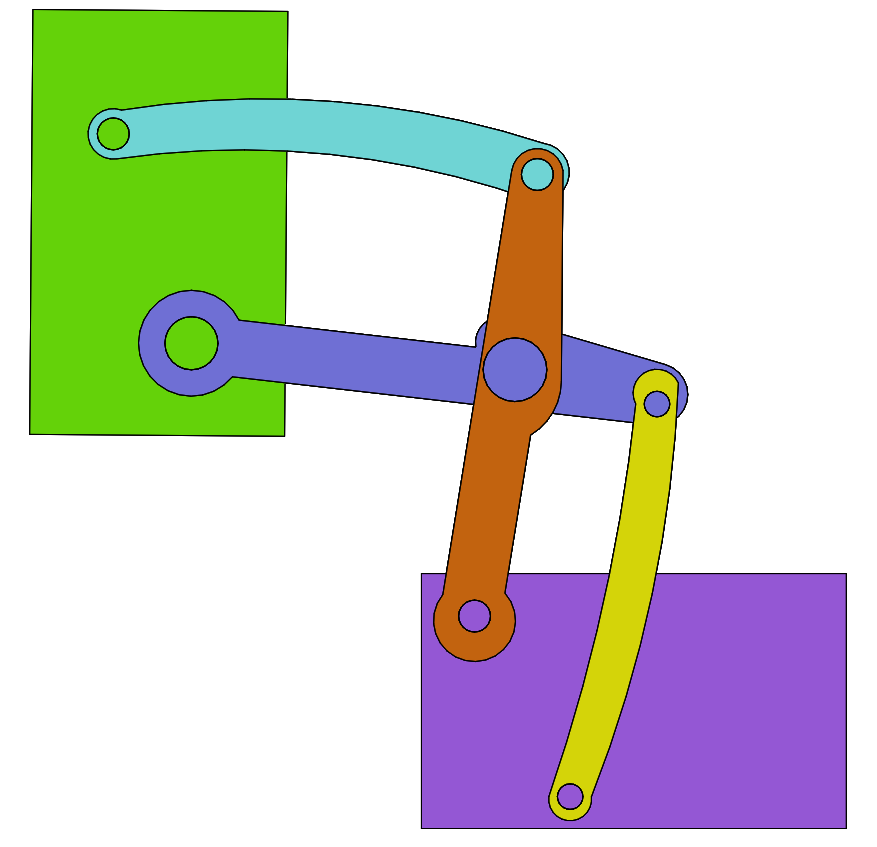
**Section 1.1: Picture (or sketch) of the mechanism with labels of all (important) parts**

References:

https://grabcad.com/library/multiple-bar-mechanism-of-a-automobile-door-hinge-1

https://www.europeana.eu/en/item/2020801/dmglib\_handler\_image\_17632023





1

6

5

4

3

2

**Section 1.2: Description of the mechanism**

**Multiple Bar mechanism of an Automobile Door Hinge**

The linkage in above figure is shown in the open position. Rocker arms 2 and 3 turn about fixed axes A and D which belong to the automobile body. Link 6 is connected by turning pairs F and G to link 5 and connecting rod 4. Link 5 is connected by a turning pair E to rocker arm 2. As the door opens, link 6 rotates about points F and G, causing rocker arm 2 to move through its range of motion. This movement is due to the connection between link 5 and rocker arm 2 at point E. The movement of rocker arm 2 causes link 5 to push connecting rod (4), which in turn swings the door open.

**Section 1.3: Purpose (application) of the mechanism**

The purpose of the automobile door hinge mechanism is to provide a stable and reliable connection between the vehicle's body and the door. This mechanism allows the car door to swing open and closed in a controlled manner while maintaining structural integrity and safety. The linkage system translates the movement of the door (link 6) into a wider opening arc while providing a stopping mechanism to hold the door open.

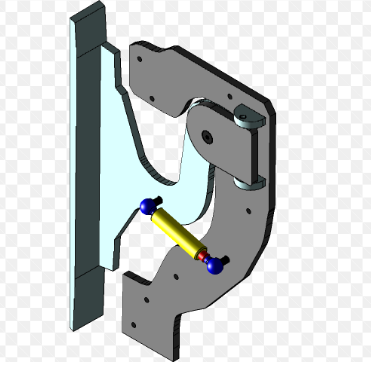
# Section 2: Context of use

**Section 2.1 History (i.e. first reported or past use of a mechanism for a similar purpose)**

The first reported use of a similar door hinge mechanism in automobiles dates back to the late 19th century when car manufacturers began incorporating hinged doors into their vehicle designs. Over time, advancements in materials and manufacturing processes have led to the development of more durable and efficient door hinge mechanisms.

**Section 2.2 (Optional) Evolution of design of mechanisms for a similar purpose OR Alternate mechanisms**

**Scissor Door Hinge**

****

Modern car doors uses variations of this mechanism, incorporating additional linkages or struts for improved strength, lighter weight materials, or features like automatic door opening.

This design uses a scissor-like linkage configuration to achieve a similar opening motion for the car door. It offers a wider door opening compared to the four-bar linkage, which can be beneficial for entry and exit, especially for larger or sportier cars.

# Section 3: Model specification

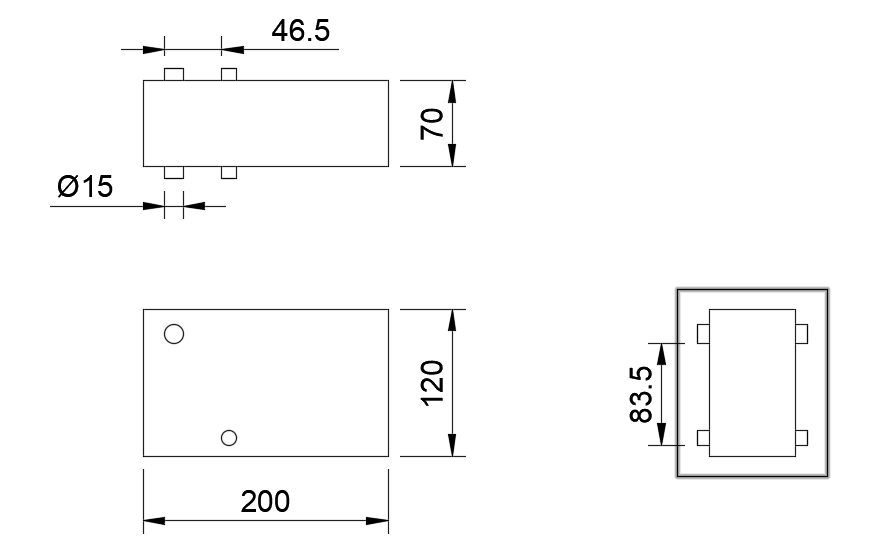
## Section 3.1 Geometry and Material

Section 3.1.1 Dimensions of every component

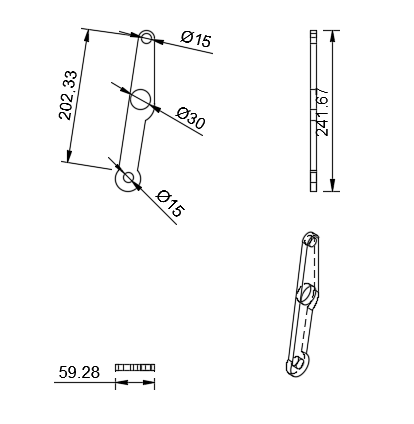
(Any other geometric detail which would be needed for creating the model)

All the dimensions are taken in mm

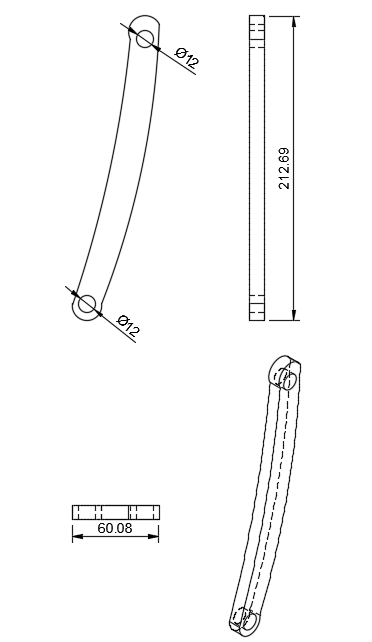
Link 1(Base)(Fixed)



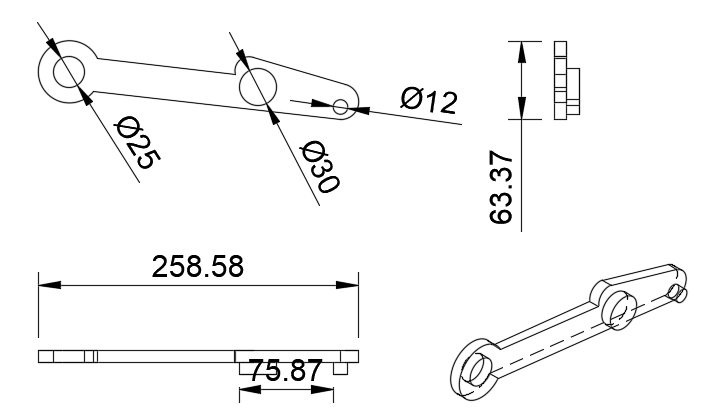
Link 2



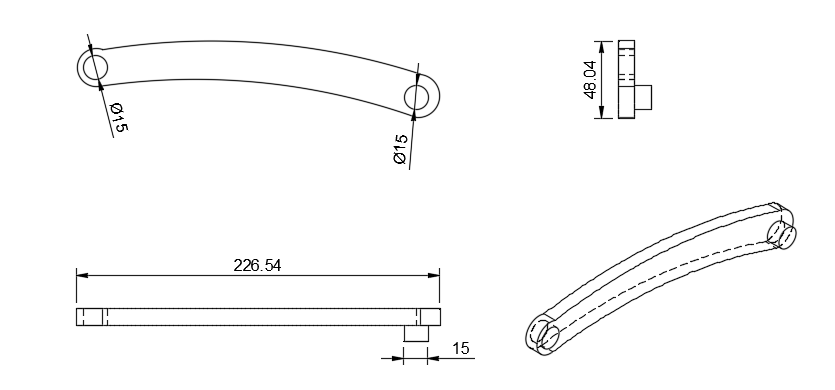
Link 3



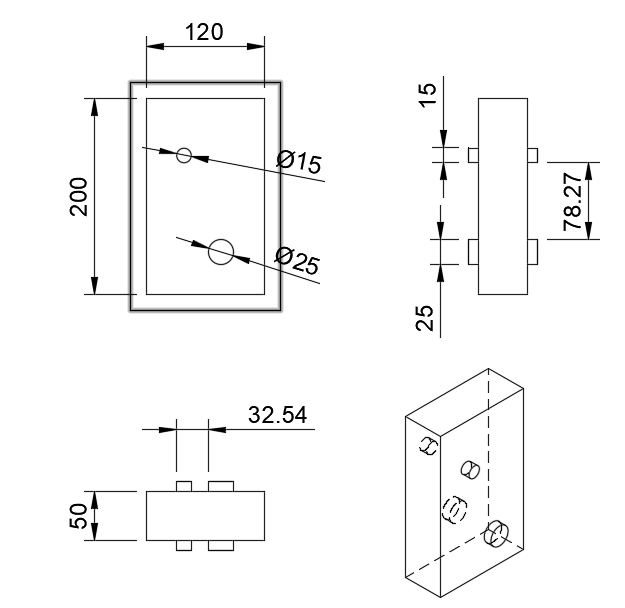
Link 4



Link 5



Link 6



Section 3.1.2 Masses of every component

Material used for each component is stainless steel(Density = 8000 kg/m^3)

|  |  |  |
| --- | --- | --- |
| Link | Volume(m^3) | Mass(kg) |
| 1 | 0.001654 | 13.234 |
| 2 | 0.000063 | 0.505 |
| 3 | 0.000041 | 0.329 |
| 4 | 0.000086 | 0.684 |
| 5 | 0.000055 | 0.437 |
| 6 | 0.001191 | 9.524 |

Section 3.1.3 Moments of inertia of every component

All are calculated at the centre of mass

All the Iij are written in terms of \*10^-3 kg-m^2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Link | Ixx | Ixy | Ixz | Iyy | Iyx | Iyz | Izz | Izx | Izy |
| 1 | 21.36 | 0.059 | 0 | 49.59 | 0.059 | 0.014 | 60.04 | 0 | 0.014 |
| 2 | 2.278 | -0.399 | 0 | 0.105 | -0.399 | 0 | 2.375 | 0 | 0 |
| 3 | 1.077 | -0.221 | 0 | 0.062 | -0.221 | 0 | 1.134 | 0 | 0 |
| 4 | 0.145 | 0.324 | -0.031 | 3.593 | 0.324 | 0.002 | 3.685 | -0.031 | 0.002 |
| 5 | 0.053 | 0.175 | -0.012 | 1.744 | 0.174 | 0.002 | 1.787 | -0.012 | 0.002 |
| 6 | 33.75 | 0.094 | -0.138 | 13.4 | 0.094 | 0.852 | 43.04 | -0.138 | 0.094 |

## Section 3.2 Constraints on the motion of each component

(require details of the joints and Degree of freedom of Mechanism)

J1 joints between 1🡨🡪2, 1🡨🡪3, 2🡨🡪4, 3🡨🡪4, 2🡨🡪5, 4🡨🡪6, 5🡨🡪6

N = 6 J1 = 7

Mobility = 3(N-1) -2J1 = 3(6-1) – 2\*7 = 1

Degree of Freedom = 1

All the 7 joints are revolute joints

Link 4 is slotted into the tab of link 2, similarly for links 4🡨🡪3, 5🡨🡪2

## Section 3.3 Motion transmission

Section 3.3.1 Driving component

Driving component is the rotating link 2

Section 3.3.2 Component which gives output of interest.

Link 6 is the output of interest

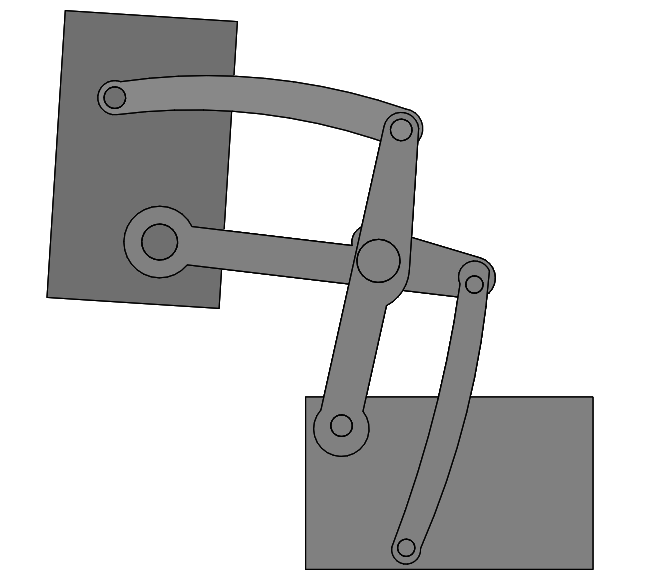
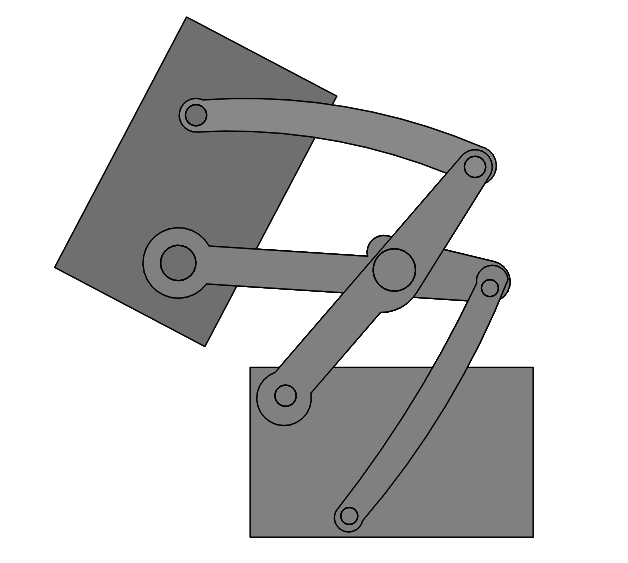
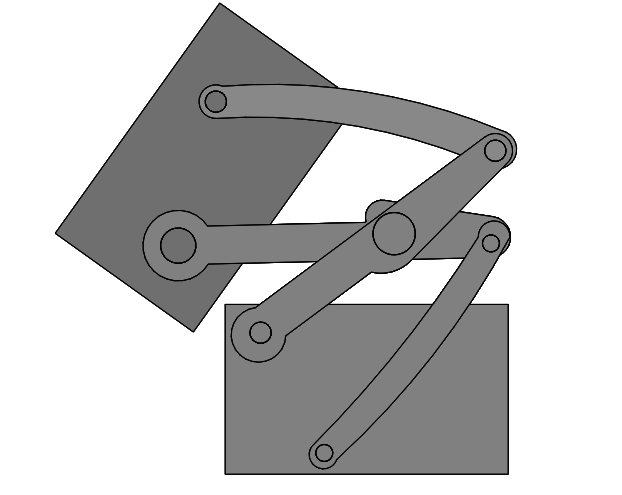
# Section 4: Kinematic simulation

This section should contain the plots of simulated kinematic parameters (i.e. displacement,

velocity, and acceleration) for the output component. The independent axis should be either Time or Motion of the input link.

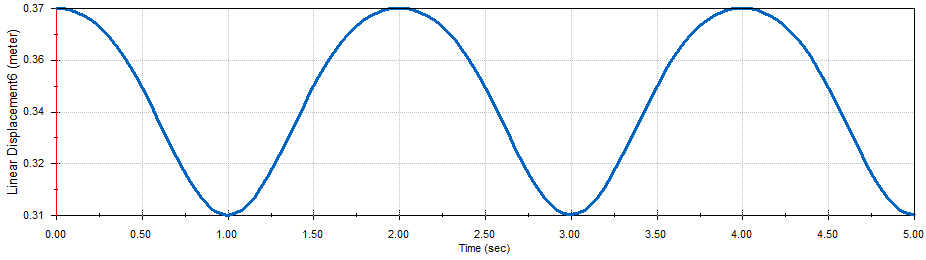
Input: A Motor gives Harmonic Angular Velocity(Frequency = 0.5Hz, Maximum Angle = 40°) to Link 2

Output: Motion of Link 6 is observed

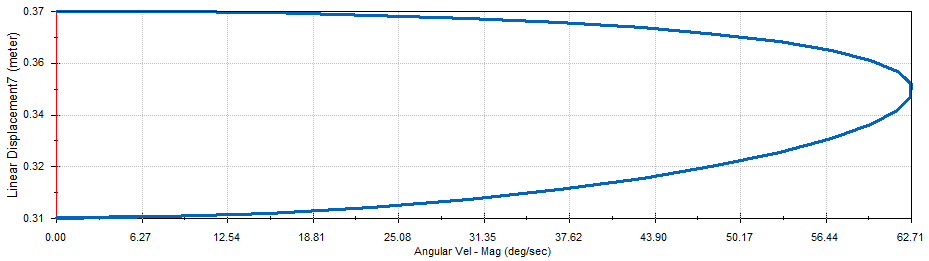
  

## Section 4.1 Output component

Section 4.1.1 Simulated profile of the output component.

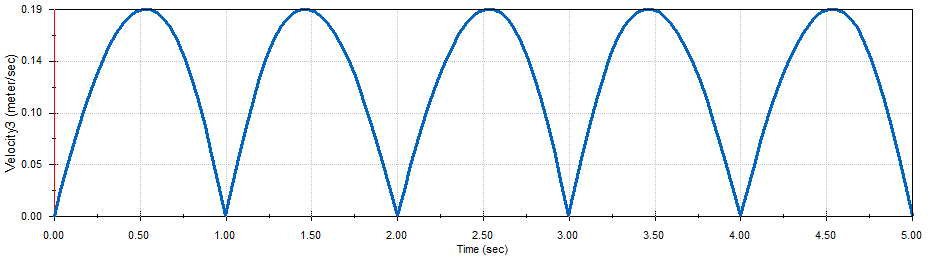


a)Linear Displacement v/s Time

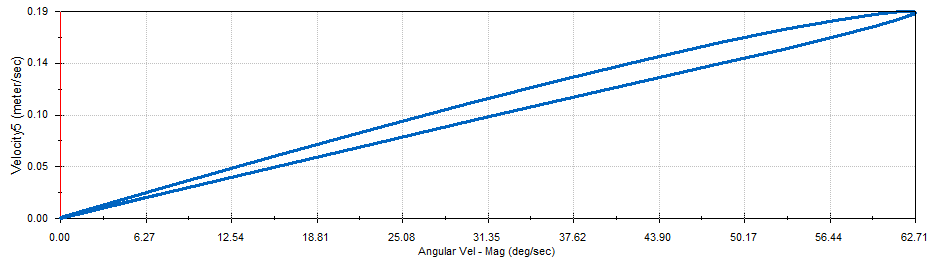


b)Linear Displacement v/s Angular Velocity of Input(Link 2)

Section 4.1.2 Simulated velocity profile of the output component

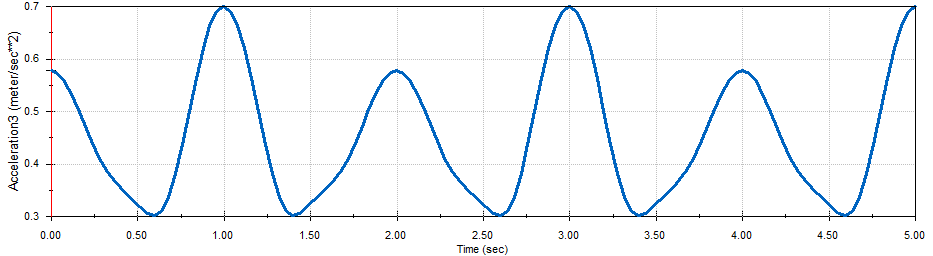


c)Linear Velocity(Magnitude) v/s Time

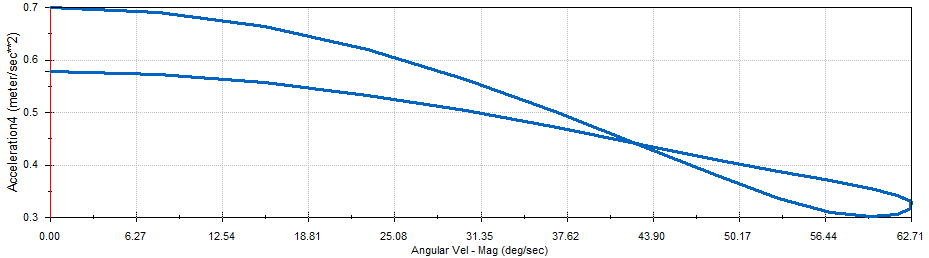


d)Linear Velocity(Magnitude) v/s Angular Velocity of Input(Link 2)

Section 4.1.3 Simulated acceleration profile of the output component



e)Linear Acceleration(Magnitude) v/s Time



f)Linear Acceleration(Magnitude) v/s Angular Velocity of Input(Link 2)

Section 4.1.4 Comments

a)The linear displacement v/s time plot is also harmonic because of harmonic input

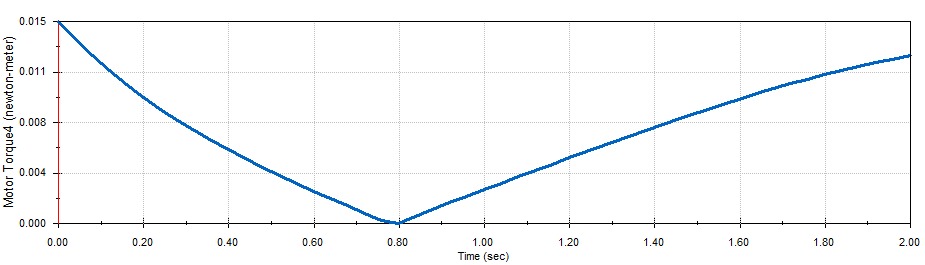
b)The upper curve of the linear displacement v/s angular velocity plot denotes the first part of the cycle in which the angular velocity increases and similarly the lower part

c)The linear acceleration plot is somewhat harmonic but with some jerks, probably because of the motor

# Section 5: Static force analysis

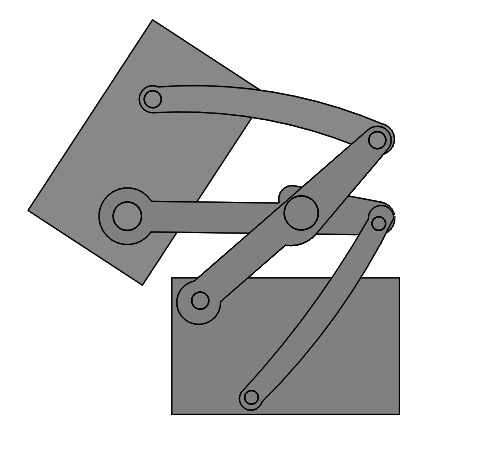
A constant force (in a constant global direction) should be applied to a specific point of the output component and the force/torque needed at the driving component to keep/hold the mechanism in this position under this condition needs to be obtained from the software. This should be done with the mechanism in four positions. The direction of the force should remain same in global coordinates. The point and direction of application of this force should be shown in four snapshots.

The point of application of force on the output component should not be chosen such that the force/torque required for holding the mechanism in the four positions is the same.

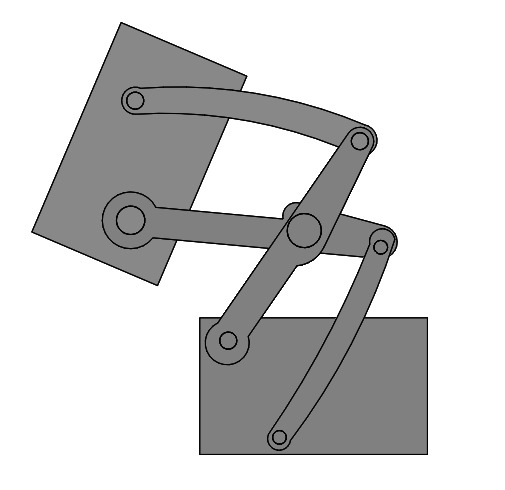


This is the graph obtained for the reaction torque when a 0.5N force is applied on the output link

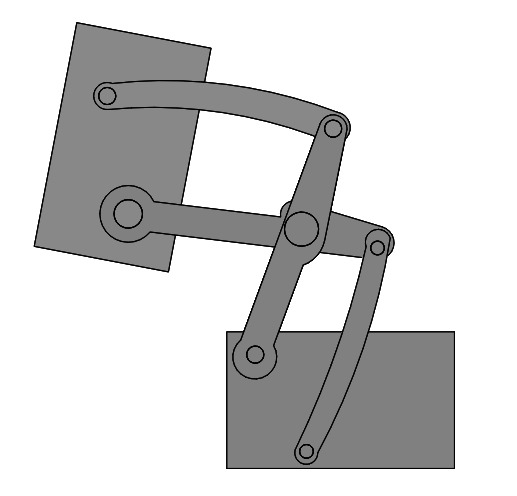
Snapshots of 4 different positions:



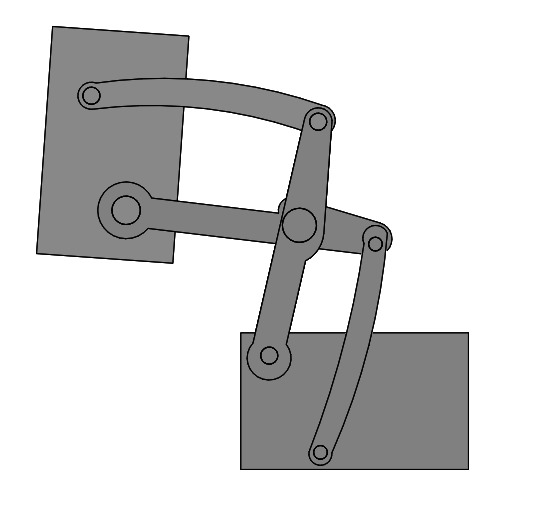
Torque Required = 0.012N-m @ 0.08s



Torque Required = 0.004N-m @ 0.48s



Torque Required – 0.001N-m @ 0.88s

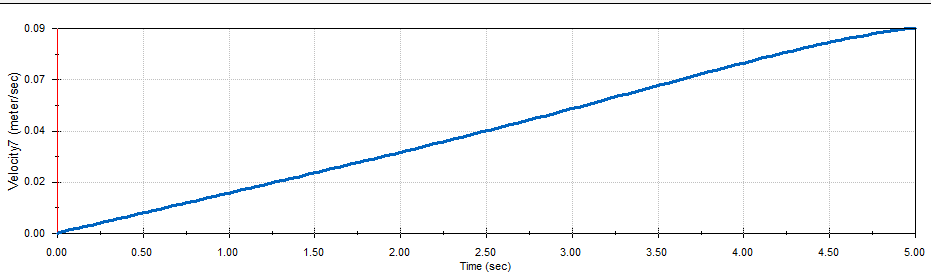
\

Torque Required = 0.004N-m @ 1.08s

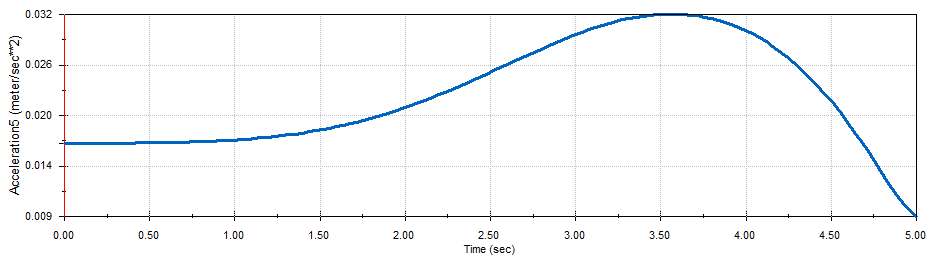
# Section 6: Dynamic motion analysis

## Section 6.1 Constant driving force/torque

Section 6.1.1 The driving component in the simulated model should be subjected to a constant force/torque and the effect of this on the velocity and acceleration of the output component should be plotted. This should be the same output component identified in Section 4.1. The plot should cover a duration of only two cycles.



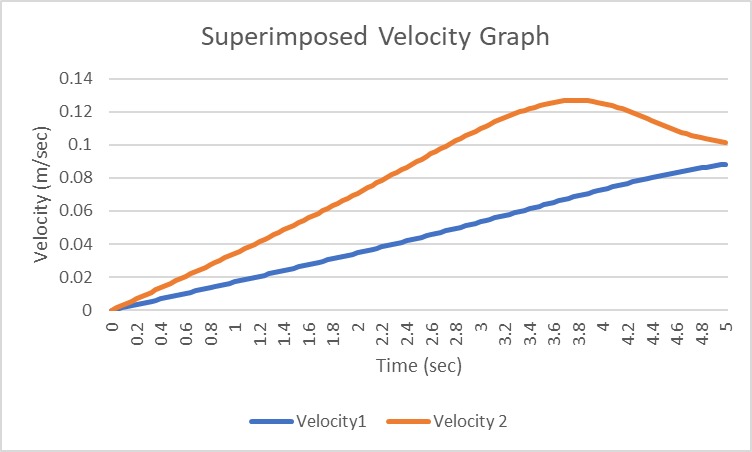
Linear Velocity of Output v/s Time for An Input Torque for 0.005 N-m



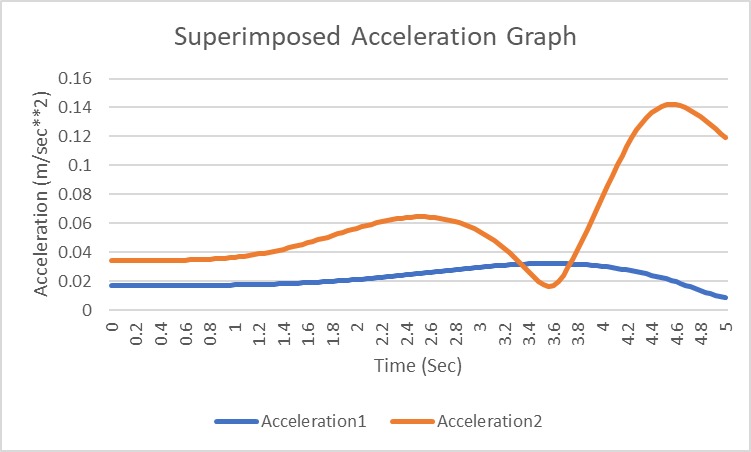
Linear Acceleration of Output v/s Time for An Input Torque for 0.005 N-m

Section 6.1.2 This will be a repeat of section 6.1.1 with the force/torque being double of the value of section 6.1.1. This is being done to observe whether there is any change in the results due to dynamic effects. The velocity and acceleration plots should be superimposed on the corresponding plots obtained in section 6.1.1.

The orange curve denotes the velocity and acceleration for the case of double torque applied



Velocity Graph for Different Input Torques(0.005N-m and 0.01N-m)



Acceleration Graph for Different Input Torques(0.005N-m and 0.01N-m)

# Section 7: Verification of kinematic analysis with a python program

Section 7.1.1 Write a python program for your mechanism to verify the displacement, velocity and acceleration of ONE chosen point of the mechanism. They should be the same with the software that you choose. Plot the displacement, velocity and acceleration of the chosen point with the help of both software and the python program.

import numpy as np

import matplotlib.pyplot as plt

#initializaion and constants

Lo = 96.177

L1 = 189.275

L2 = 70.901

L3 = 109.975

L4 = 99.901

L5 = 200.925

L6 = 106.008

L7 = 149.882

theta = np.deg2rad(82.3)      #convert degree to radian

alpha = np.deg2rad(297.9)

beta = np.deg2rad(77.8)

gamma = np.deg2rad(352.7)

delta = np.deg2rad(174.6)

sigma = np.deg2rad(291)

#defining list of beta values and creating empty lists for (x,y) of point p

theta\_star = np.arange(0, 42, 5)

theta\_star = theta\_star.tolist()

beta\_star = []

gamma\_star = []

s1 = []

s2 = []

t1 = []

t2 = []

# x is a vector, x = [beta\_star, gamma\_star]

def func(theta\_star, x):

    f1\_x = L1\*np.cos(theta+theta\_star) + Lo\*np.cos(alpha) - L2\*np.cos(beta+x[0])- L3\*np.cos(gamma+x[1])

    f1\_y = L1\*np.sin(theta+theta\_star) + Lo\*np.sin(alpha) - L2\*np.sin(beta+x[0])- L3\*np.sin(gamma+x[1])

    return np.array([f1\_x, f1\_y])

def jacobian(x):

    j1 = L2\*np.sin(beta+x[0])

    j2 = L3\*np.sin(gamma+x[1])

    j3 = -L2\*np.cos(beta+x[0])

    j4 = -L3\*np.cos(gamma+x[1])

    j = np.array([[j1, j2], [j3, j4]])

    return j

# Tolerence -> error should be less than the defined tolerence

max\_iterations = 10000

tolerance = 0.000000001

def solve(x0, b, theta\_star):

    x = x0

    for i in range(max\_iterations):

        j = jacobian(x)

        xs = np.linalg.solve(j, -func(theta\_star, x))  # Pass theta\_star to func()

        x = x + xs

        if np.linalg.norm(xs) < tolerance:

            return x

x0 = np.array([0,0])

for j in range(len(theta\_star)):

    b = np.deg2rad(theta\_star[j])

    x = solve(x0, b, theta\_star[j])  # Pass theta\_star[j] to solve()

    beta\_star.append(x[0])

    gamma\_star.append(x[1])

    x0 = x

# x is a vector, x = [delta\_star, sigma\_star]

def func(beta, alpha, x):

    f2\_x = L4\*np.cos(theta+theta\_star) + L5\*np.cos(delta+x[0]) + L6\*np.cos(sigma+x[1]) + L7\*np.cos(gamma+gamma\_star)

    f2\_y = L4\*np.sin(theta+theta\_star) + L5\*np.sin(delta+x[0]) + L6\*np.sin(sigma+x[1]) + L7\*np.sin(gamma+gamma\_star)

    return np.array([f2\_x, f2\_y])

def jacobian(x):

    j1 = -L5\*np.sin(delta+x[0])

    j2 = -L6\*np.sin(sigma+x[1])

    j3 = L5\*np.cos(delta+x[0])

    j4 = L6\*np.cos(sigma+x[1])

    j = np.array([[j1, j2], [j3, j4]])

    return j

# Tolerence -> error should be less than the defined tolerence

max\_iterations = 100000

tolerance = 0.000001

def solve(x0, b, a):

    x = x0

    for i in range(max\_iterations):

        j = jacobian(x)

        xs = np.linalg.solve(j, -func(b, a, x))

        x = x+xs

        if np.linalg.norm(xs) < tolerance:

            return x

x0 = np.array([0,0])

for j in range(len(theta\_star)):

    b = np.deg2rad(theta\_star[j])

    a = np.deg2rad(gamma\_star[j])

    x = solve(x0, b, a)

    s1.append((L3+L4)\*np.cos(theta + b) + L5\*np.cos(delta + x[0]))

    t1.append((L3+L4)\*np.sin(theta + b) + L5\*np.sin(delta + x[0]))

    s2.append(L3\*np.cos(theta + b) - L7\*np.cos(delta + a))

    t2.append(L3\*np.sin(theta + b) - L7\*np.sin(delta + a))

    x0 = x

plt.plot(np.sqrt((np.square(s1-s2) + np.square(t1-t2))), theta\_star)

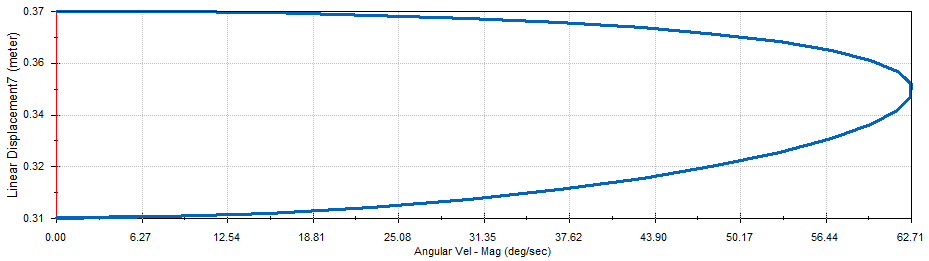
plt.grid(True)

plt.title("Displacement")

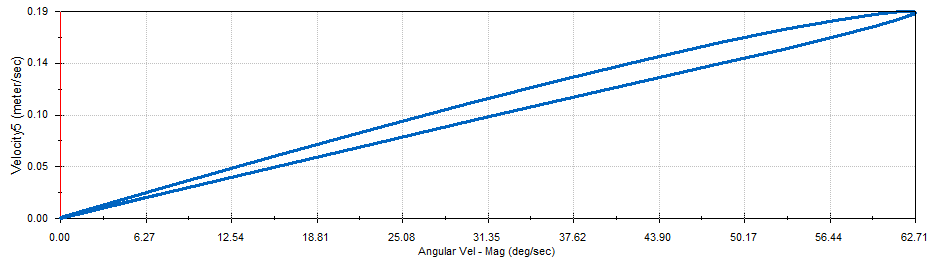
plt.xlabel("x (m)")

plt.ylabel("y (m)")

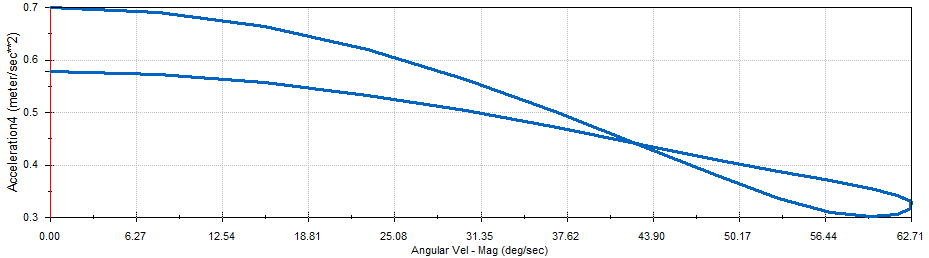
plt.show()



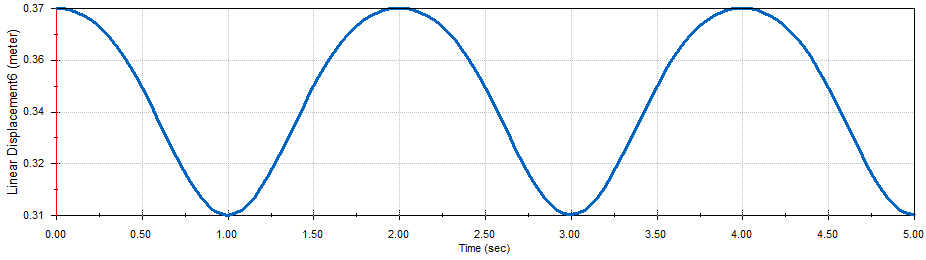
Linear Displacement



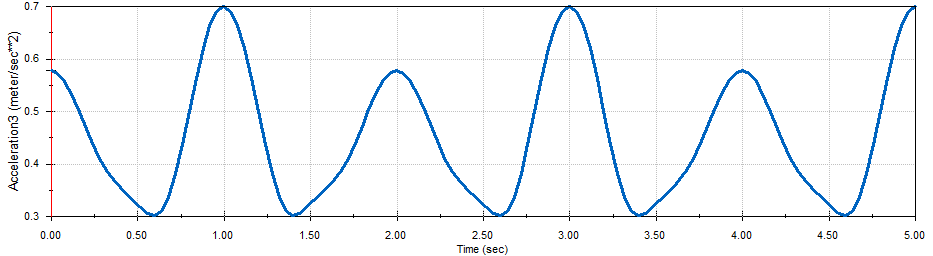
Linear Speed



Linear Acceleration



Linear Displacement vs Time



Acceleration vs Time