Design and Fabrication of Experiment for Dynamic Analysis of Mechanisms

submitted in partial fulfilment of the requirements
for the degree of

BACHELOR OF TECHNOLOGY
in
MECHANICAL ENGINEERING
by

AAKASH	**ME16B001**
A. AKHIL	**ME16B003**

Supervisor(s)

Dr. Sriram Sundar

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

December 2019

BONA FIDE CERTIFICATE

This is to certify that the thesis titled **Design and Fabrication of Experiment for Dynamic Analysis of Mechanisms**, submitted by **Aakash** and **A. Akhil**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Tirupati
Date: 08-12-2019

Dr. Sriram Sundar
Guide
Assistant Professor
Department of Mechanical Engineering
IIT Tirupati - 517501

ACKNOWLEDGEMENTS

Thanks to all those who helped us during the course of our B.Tech. project at IIT Tirupati. We are thankful to and fortunate enough to get constant encouragement, support and guidance from Dr. Sriram Sundar of Mechanical Engineering Department which helped us in successfully completing first phase our project work. We would also like to extend our sincere esteems to all staff in laboratory for their timely support.

To our families, thank you for encouraging us in all of our pursuits and inspiring us to follow our dreams. We are especially grateful to our parents, who supported us emotionally and financially.

ABSTRACT

KEYWORDS: Slider Crank Mechanism; Four bar mechanism; Six-bar mechanism; Dynamic analysis; Kinematic analysis; Simulation.

To design and fabricate an experimental setup for static, kinematic and dynamic analysis of mechanism(s). This will integrate the theoretical knowledge with the practical experiment, giving learner a chance to carry out the analysis experimentally, to plot the results, and most importantly, to bolster their understanding by providing them the opportunity to match the analytical results with the actual experimental results. The setup will have retrofitted parts in order to let the student carry out multiple experiments in the same setup.

This is abstract. It should tell in short what you have done. The first sentence is an objective. Please rewrite this section for abstract.

TABLE OF CONTENTS

Even there are kinematic analyses right?

2.1 Kinematic analysis
2.1.1 Math formulation
2.1.1.1 Slider crank.
⋮
2.1.2 Using commercial software
2.1.2.1 Geogebra
2.1.2.2 Python.
2.2 Dynamic analysis

LIST OF FIGURES

ABBREVIATIONS

The following list describes the significance of various abbreviations and acronyms used throughout the report.

KDM	Kinematics and Dynamics of Machinery
PMSM	Permanent Magnet Synchronous Motor
CAD	Computer Aided Design
CSV	Comma Seperated Values
FBD	Free Body Diagram

CHAPTER 1

INTRODUCTION

1.1 Motivation and Objective

The study of the ~~*ME2206*~~ *Kinematics and Dynamics of Machinery* (ME2206) lies at the very core of a mechanical engineering background. Although, little has changed in the way the subject is presented, our methodology brings the subject alive and current. We present the design and fabrication of a novel experimental setup for carrying out static, kinematic and dynamic analysis of mechanism (Figure 1.1). The mechanism is designed to be configurable to three different types of mechanisms namely - four bar, slider crank and six bar depending on the use case as shown in Figure 1.2. The mechanism has retrofitted parts (different link lengths and sliders) to facilitate multiple experiments in the same setup.

this part has to be emphasized in report as well as slides. A single expt. for static, kinematic & dynamic analysis.

The learner gets to "play" with the mechanism parameters and immediately see their effects. This will enhance one's grasp of the concepts and the development of analytical skills. Hence greatly supplementing and reinforcing the theoretical understanding of the undergraduate students taking the course.

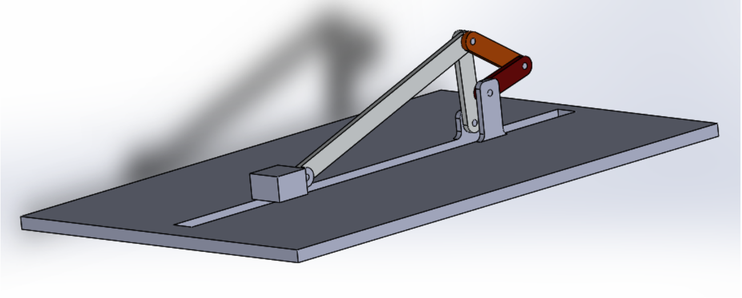


Figure 1.1: CAD model of the mechanism

1.2 Present method(s) and Challenges

In the present scenario finding relevant mechanisms integrated with different sensor measurement and data acquisition is hard to find. Almost all the mechanisms that are

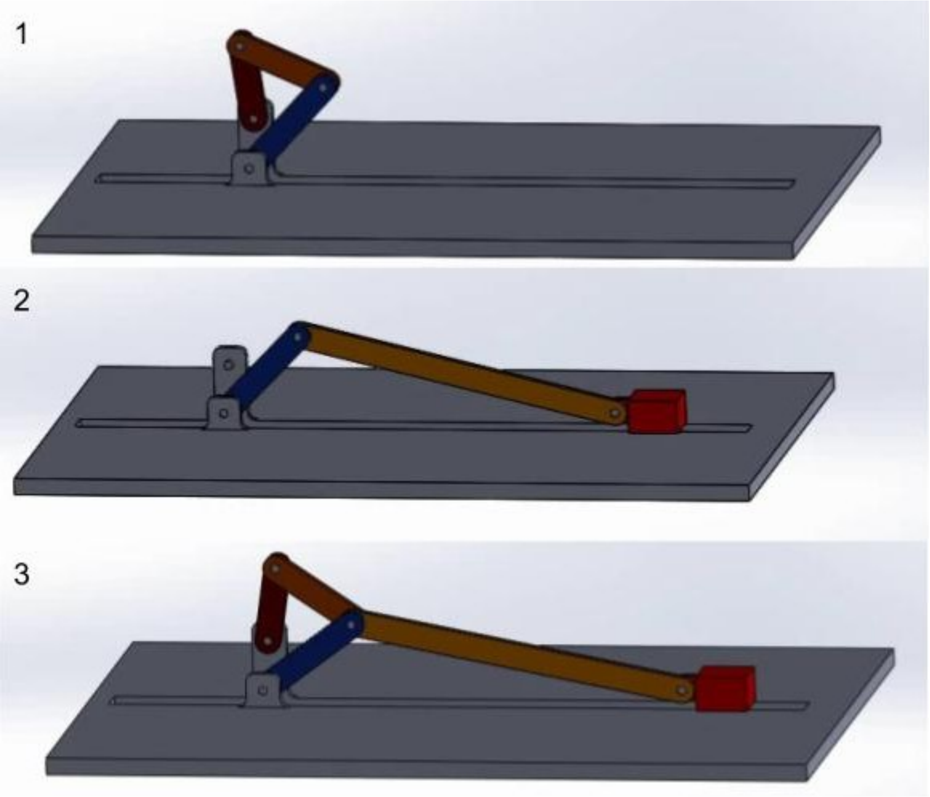


Figure 1.2: Three possible configurations of the mechanism: (1) Four bar, (2) Slider crank and (3) Six bar

off-the-shelf

available off the self are available only for demonstrations purpose (not does ~~providing~~ have any measurement) and not as an experiment. capability.

You can also say that doing static, kinematic & dynamic analysis with some mechanism follows the foot steps of how it is done in theory. ⇒ better appreciation of concepts.

In Chapter 2 we present thoroughly all the methodologies used in designing and modelling the mechanism. This includes both the analysis carried out by analytical as well as numerical method. In the chapter that follows we bring out the various design considerations that has to be taken into account while manufacturing the product. The final two sections present the result(s) and the conclusion(s).

CHAPTER 2

ANALYSIS OF MECHANISMS

2.1 Mathematical Formulation

In this section we bring out the equations of motion associated with the three different mechanisms. These equations are used to carry out the kinematic and dynamics method theoretically, which then in turn be compared with the numerical solutions obtained by other simulation tools.

2.1.1 Slider crank mechanism

The equations of motion pertaining to the slider crank mechanism are presented in this subsection.

$$\vec{V}_{B/A} = \vec{V}_B - \vec{V}_A \tag{2.1}$$

For link OA we have

$$\vec{V}_{A/O} = \vec{\omega}_{AO} \times \vec{r}_{A/O} = \vec{V}_A \tag{2.2}$$

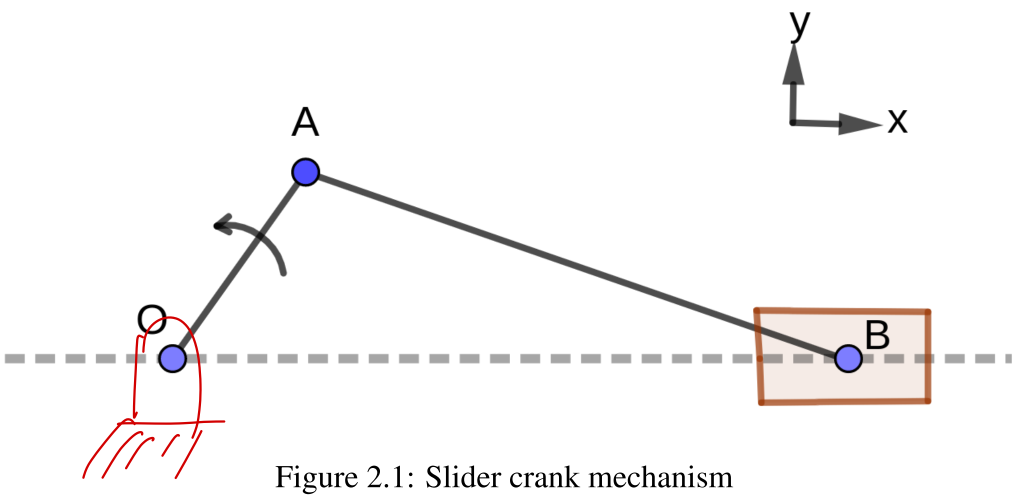


Figure 2.1: Slider crank mechanism

Since 'O' is ground link we have $\vec{V}_O = 0 \implies \vec{V}_{A/O} = \vec{V}_A - \vec{V}_O = \vec{V}_A$. As point B lies on the slider and is moving leftwards it will have velocity along the negative x-direction i.e. $\vec{V}_B = -V_B\hat{i}$; because $\vec{\omega}_{AO} = \omega_{AO}\hat{k}$. For link AB we have

$$\vec{V}_{B/A} = \vec{\omega}_{BA} \times \vec{r}_{B/A} \tag{2.3}$$

Substituting equation 2.3 in equation 2.2 we obtain

$$\vec{\omega}_{BA} \times \vec{r}_{B/A} = \vec{V}_B - \vec{V}_A \tag{2.4}$$

Using equation 2.4 we can find ω_{BA} and V_B. For finding the accelerations we can write

$$\vec{a}_{A/O} = \vec{a}_A = \vec{a}_A^t + \vec{a}_A^n \tag{2.5a}$$

$$\vec{a}_A^t = \vec{\alpha} \times \vec{r} \tag{2.5b}$$

$$\vec{a}_A^n = \vec{\omega} \times \vec{\omega} \times \vec{r} \tag{2.5c}$$

where, $\vec{a}_A^t$ and $\vec{a}_A^n$ denote the tangential and normal acceleration of point A. As point B lies on the slider, we have $\vec{a}_B = a_B\hat{i}$.

$$\vec{a}_{B/A} = \vec{a}_{B/A}^t + \vec{a}_{B/A}^n = \vec{a}_B - \vec{a}_A \tag{2.6}$$

Using the above equation we can find α_{BA} and $\vec{a}_B$.

Force balance for link OA

$$F_{OX} + F_{AX} = 0 \tag{2.7a}$$

$$F_{OY} + F_{AY} = m_1 g \tag{2.7b}$$

$$I_{AO}\vec{\alpha}_{AO} = \vec{r}_{C_1 O} \times m_1 g(-\hat{j}) + \vec{r}_{A/O} \times \vec{F}_A \tag{2.7c}$$

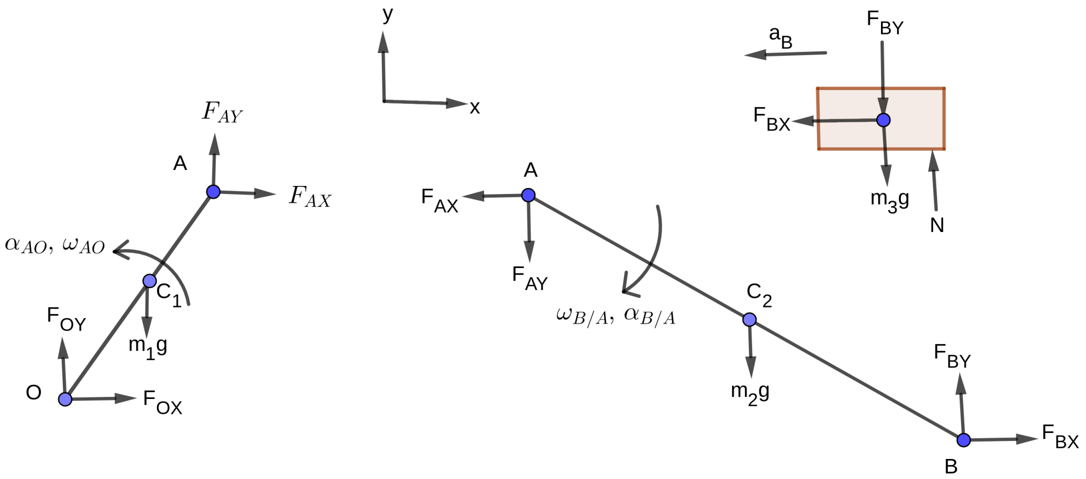


Figure 2.2: Free Body Diagram of link OA, AB and slider

where, $\vec{F}_A = \vec{F}_{AX}\hat{i} + \vec{F}_{AY}\hat{j}$. Similarly for link AB we have

$$F_{AX} = F_{BX} \tag{2.8a}$$

$$F_{BY} = F_{AY} + m_2 g \tag{2.8b}$$

$$I_{BA}\vec{\alpha}_{B/A} = \vec{r}_{C_2/A} \times m_2 g(-\hat{j}) + \vec{r}_{B/A} \times \vec{F}_B \tag{2.8c}$$

where, $\vec{F}_B = \vec{F}_{BX}\hat{i} + \vec{F}_{BY}\hat{j}$. Performing force balance for the slider

$$F_{BY} + m_3 g = N \tag{2.9a}$$

$$m_3 a_B = F_{BX} \tag{2.9b}$$

2.1.2 Four bar mechanism (with 4 revolute joints)

$$\vec{V}_{A/O} = \vec{\omega}_{AO} \times \vec{r}_{A/O} = \vec{V}_A \tag{2.10a}$$

$$\vec{V}_{C/D} = \vec{\omega}_{CD} \times \vec{r}_{C/D} = \vec{V}_C \tag{2.10b}$$

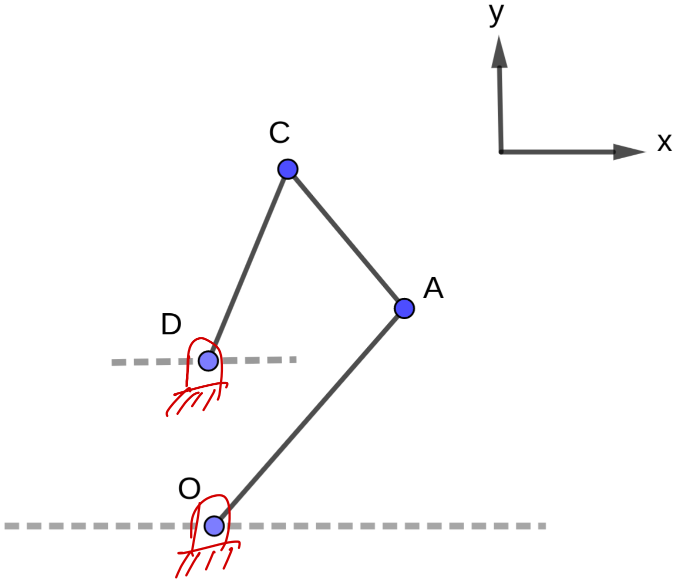


Figure 2.3: Four bar mechanism

$$\vec{V}_{A/C} = \vec{\omega}_{AC} \times \vec{r}_{A/C} = \vec{V}_A - \vec{V}_C = \vec{\omega}_{AO} \times \vec{r}_{A/O} - \vec{\omega}_{CD} \times \vec{r}_{C/D} \tag{2.11}$$

Using the above equation we can find ω_{CD} and ω_{AC}.

$$\vec{a}_{A/O} = \vec{a}_A = \vec{a}^t_A + \vec{a}^n_A \tag{2.12a}$$

$$\vec{a}_{C/D} = \vec{a}_C = \vec{a}^t_C + \vec{a}^n_C \tag{2.12b}$$

$$\vec{a}_{A/C} = \vec{a}^t_{A/C} + \vec{a}^n_{A/C} = \vec{a}_A - \vec{a}_C \tag{2.13}$$

Here $\vec{a}^t = \vec{\alpha} \times \vec{r}$ and $\vec{a}^n = \vec{\omega} \times \vec{\omega} \times \vec{r}$. Using the above equations we can find α_{AC} and α_{CD}

2.1.3 Six bar mechanism

2.2 Kinematic analysis

Please avoid first person usage.- check the entire report.

In this section we present the methodology employed in order to simulate the kinematics of the various mechanisms. In order to carry out the preliminary analysis of the mechanism we used the *GeoGebra*, which is an interactive math and geometry software. This

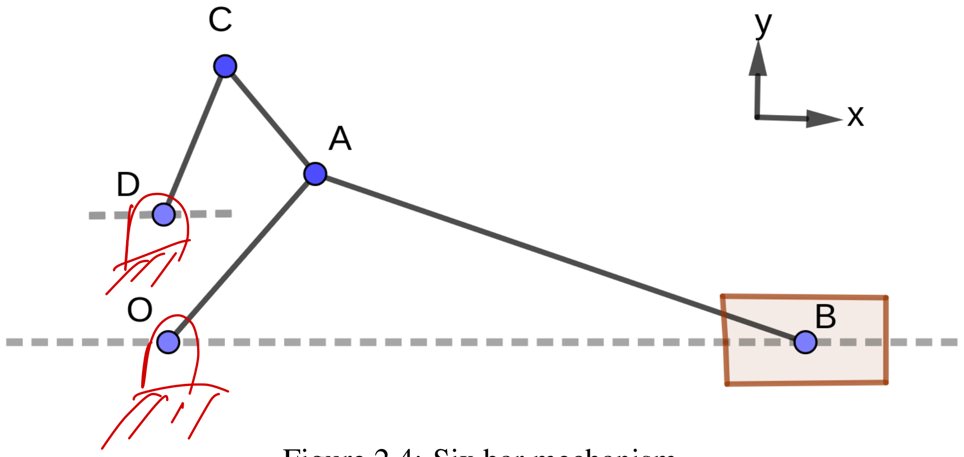
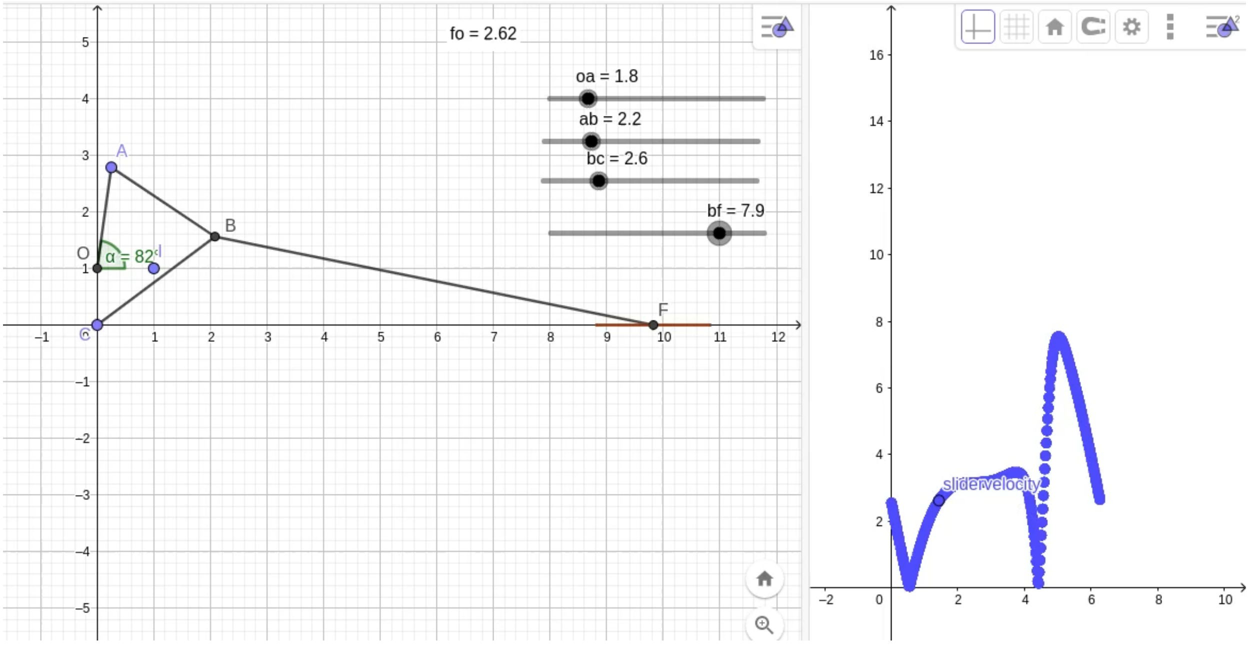


Figure 2.4: Six bar mechanism

provided the coarse link lengths required for having the least deviant constant velocity mechanism as shown in Figure 2.5. Once the coarse link lengths were obtained, we developed an in-house Python code that can do iterations on the link lengths so as to minimise any fluctuations in the constant velocity mechanism. The visual output of the Python simulation can be seen in Figure 2.6 while rest of the simulation data is store in a CSV file.



fo = 2.62

oa = 1.8

ab = 2.2

bc = 2.6

bf = 7.9

α = 82°

Figure 2.5: Snapshot from GeoGebra showing the mechanism (left) and the velocity profile (right)

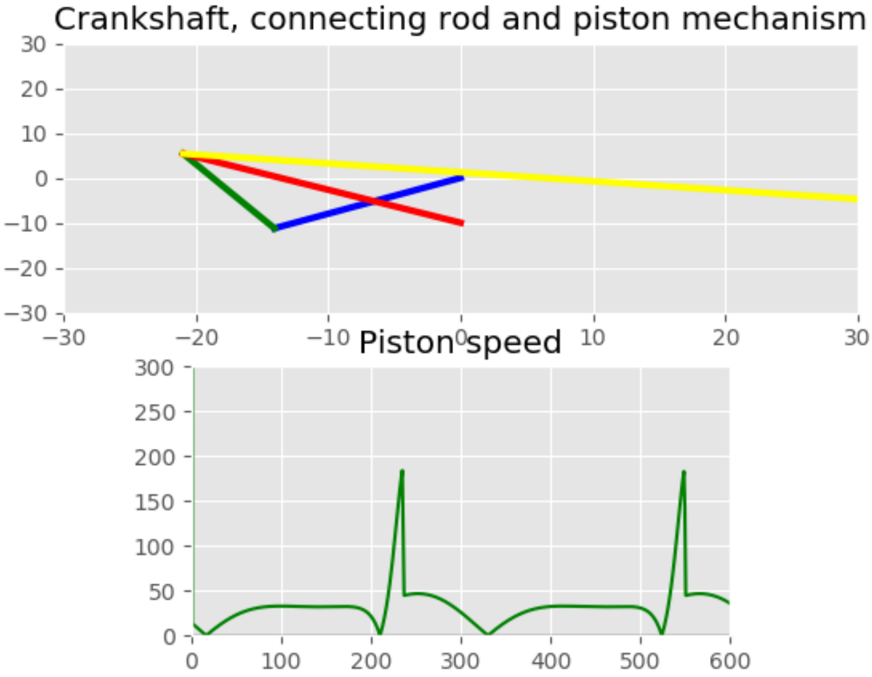


Figure 2.6: Output window from Python for kinematic simulation

2.2.1 GeoGebra Modelling

2.2.2 Python Simulation

2.3 Dynamic analysis

Mathworks'

For performing the dynamic analysis of the mechanism we used MATLAB Simscape Mechanical package. The CAD was created in the package itself and analysed the same for various forces, torques, acceleration and velocities by taking the material as aluminium (Figure 2.7, **??**). This analysis is necessary in order to ease the process of selection of actuator, material and the sensor system.

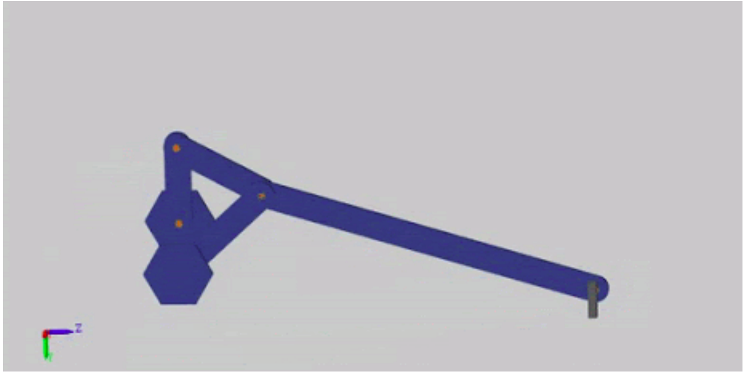
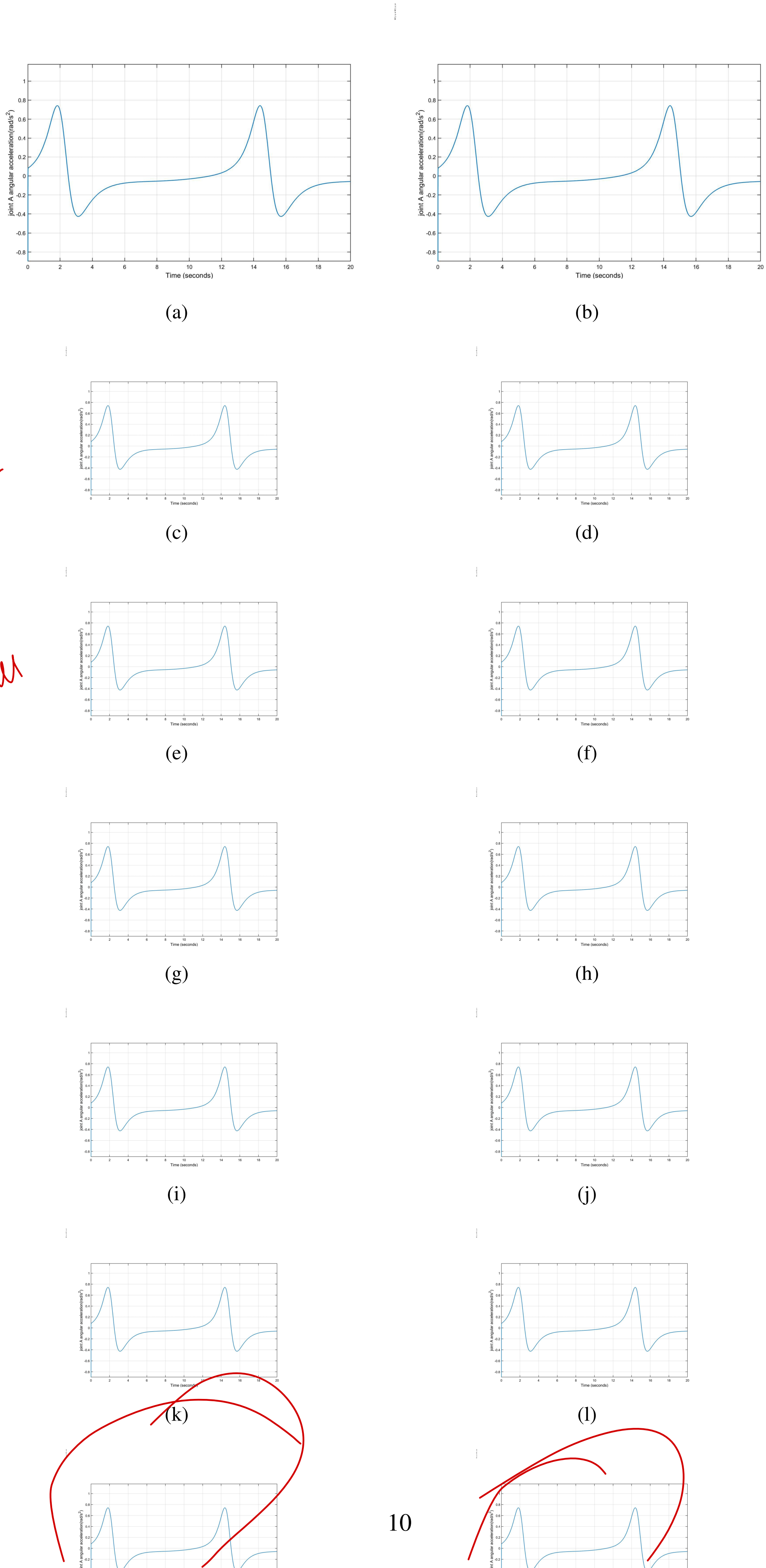


Figure 2.7: CAD model in MATLAB Simscape

Very small numbers - Not at all readable'



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Even fig is too small. We can't get anything from the figs.

going out of bounds

CHAPTER 3

DESIGN FOR MANUFACTURING

3.1 Design considerations

** TBD **

3.2 Material Selection

** TBD **

CHAPTER 4

RESULTS

TBD

CHAPTER 5

CONCLUSION AND FUTURE WORK

The complete analysis of the mechanism(s) was performed using applicable method and software. 3D CAD model of the mechanism was created to facilitate better visualisation. Various design considerations were addressed and manufacturing workflow and plan were accomplished as a prerequisite to manufacturing. Also, the BoM was prepared and materials acquisition from the vendor is in process.

In the near future we shall start the manufacturing process based on the work that has been already accomplished by now. After completing the manufacturing rigorous testing will be done on the mechanism and corrections be made accordingly. In the end we will work on preparing the documentation for the product.

APPENDIX A

CODE LISTING

The code used for carrying out this work is presented here. Although due to the high volume of the code involved in this project, it is hosted with complete documentation on GitHub at https://github.com/AakashSYadav/BTP .

A.1 Six-bar mechanism (Python)

```python
# http://firsttimeprogrammer.blogspot.com/2015/02/crankshaft-connecting-rod-and-piston.html

# import matplotlib.animation as animation
# import matplotlib.pyplot as plt
# from sympy import symbols, Eq, solve
import numpy as np
import time
import math
import csv

# plt.style.use('ggplot')
link_a_pivot = (0,-10)
link_p_pivot = (0,0)
# x, y, h = symbols('x y h')

class My_mechanism(object):

    # The init function
    def __init__(self,a,b,p,q,omega):
        self.a = a                            # Rod a
        self.b = b                            # Rod b
        self.p = p                            # Rod p
        self.q = q                            # Rod q
        self.omega = omega            # Angular speed of rod a in rad/s
        self.theta0 = 0               # Initial angular position of rod a theta()
        self.set0 = (0,-100)          # for selecting one of the two solutions
        self.k = 0.00001              # Initial time for animation
        self.c_position = []          # Piston position for animation
        self.c_speed = []             # storing piston speed
        self.c_time = []              # storing the time intervals
        self.pos_old = 0  # old position of the piston for c_dot calculation

    # Angular position of rod a as a function of time
    def theta(self,t):
        # if not all(t):
        #     raise ValueError(’Each time t must be greater than 0’)
        theta = self.theta0 + self.omega*t
        return theta

    def rod_p_position(self,t):
        p_y = self.p*np.sin(self.theta(t))
        p_x = self.p*np.cos(self.theta(t))
        return p_x,p_y

    def rod_q_position(self,t):
        px,py = self.rod_p_position(t)
        ax,ay = link_a_pivot
        c = ((self.p**2-px**2-py**2) - (self.a**2-ax**2-ay**2)) / (2*(ax-px))
        d = (py-ay)/(ax-px)
        D = (d*(px-c)+py)**2 - (1+d**2)*(py**2 + (px-c)**2 - self.p**2)
        if D<0:
            print("complex")
            raise Exception(’complex’)
        D2 = D**0.5
        y0 = (d*(px-c)+py+D2)/(1+d**2)
        y1 = (d*(px-c)+py-D2)/(1+d**2)
        x0 = c+d*y0
        x1 = c+d*y1
        set1=(x0,y0)
        set2=(x1,y1)
        # print(set1,set2)

        dist1 = math.hypot(set1[0] - self.set0[0], set1[1] - self.set0[1])
        dist2 = math.hypot(set2[0] - self.set0[0], set2[1] - self.set0[1])
        if dist1<dist2:
            self.set0=set1
            return set1[0],set1[1]
        else:
            self.set0=set2
            return set2[0],set2[1]

    def piston_position(self,t):
        q_x,q_y = self.rod_q_position(t)
        h0 = q_x+(self.b**2 - (q_y-link_a_pivot[1])**2)**0.5
        return h0

    # Piston speed
    def c_dot(self,t):
        c_x = self.piston_position(t)
        # c_dot = (c_x-self.pos_old)/0.05
        c_dot = abs(c_x-self.pos_old)/0.01  # dt = 0.01
        self.pos_old = c_x
        return c_dot

    def velocity_params(self):
        while self.k<3.5: # to get one complete cycle
            speed = self.c_dot(self.k)
            self.k+=0.01

            """
            Uncomment the below 3 lines and comment the rest
            to collect data for a single set of params using "c_out_2.py"
            """
            # with open('oneP_best2.csv', mode='a') as label_file:
            #    label_writer = csv.writer(label_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
            #    label_writer.writerow([self.k,speed])

            if self.k>0.4 and self.k<2.1:# store the ROI
                self.c_speed.append(speed)
                self.c_time.append(self.k)

        # print(self.c_speed)
        maxi = []
        mini = []
        for i in range(1,len(self.c_speed)-1):
            # local maxima
            if self.c_speed[i]>self.c_speed[i+1] and self.c_speed[i]>self.c_speed[i-1]:
                maxi.append(self.c_speed[i])
                # print("max",self.c_speed[i])
            # local minima
            if self.c_speed[i]<self.c_speed[i+1] and self.c_speed[i]<self.c_speed[i-1]:
                mini.append(self.c_speed[i])
                # print("min",self.c_speed[i])
        # print(self.a,self.b,self.p,self.q,max(maxi),min(mini))
        with open(’data_best.csv’, mode=’a’) as label_file:
           label_writer = csv.writer(label_file, delimiter=’,’, quotechar=’"’, quoting=csv.QUOTE_MINIMAL)
           label_writer.writerow([self.a,self.b,self.p,self.q,max(maxi),min(mini)])
        mini.clear()
        maxi.clear()
        self.c_speed.clear()
        self.time.clear()
```

```
from crankshaft import *
import time

initial_time = time.time()
# #a,b,p,q,omega
# best
for a in range(20,35):
    for b in range(70,80):
        for p in range(15,25):
            for q in range(15,30):
                m = My_mechanism(a,b,p,q,2)
                try:
                    m.velocity_params()
                except Exception:
                    continue

print("Processing time : ",time.time()-initial_time,"s")
```

REFERENCES

Please populate this ASAP.