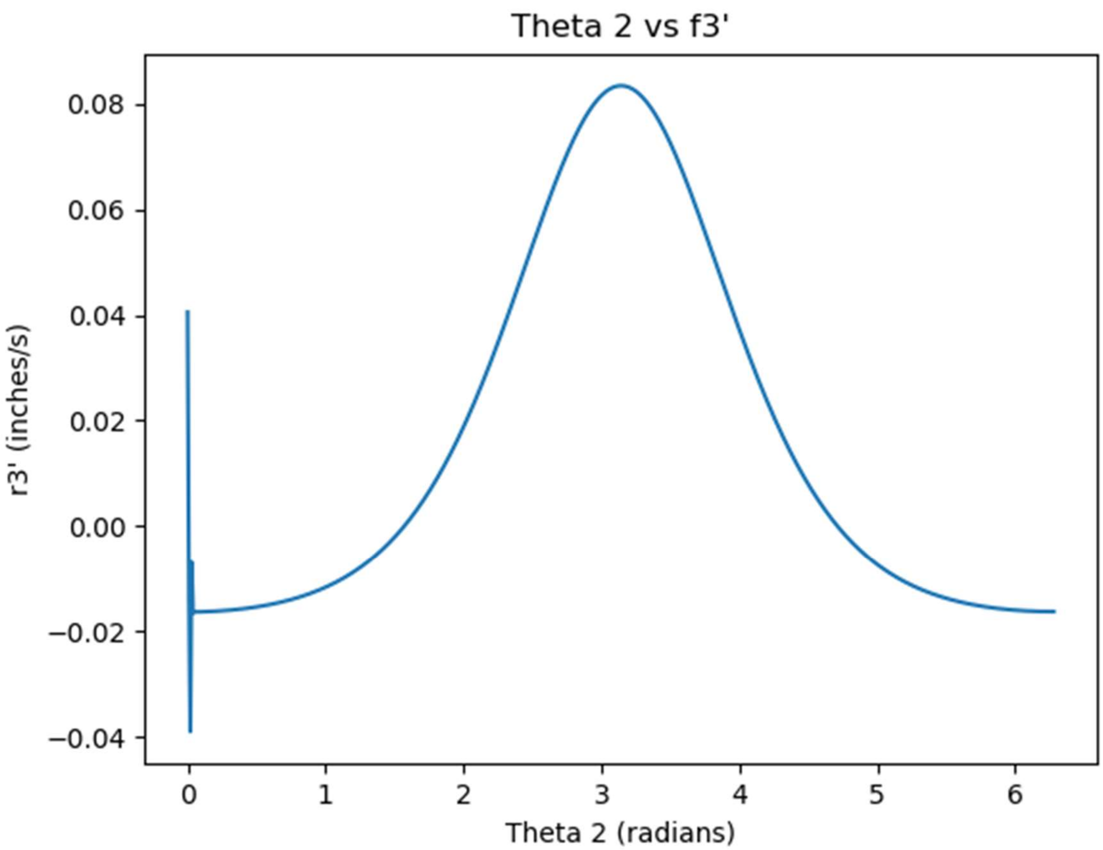
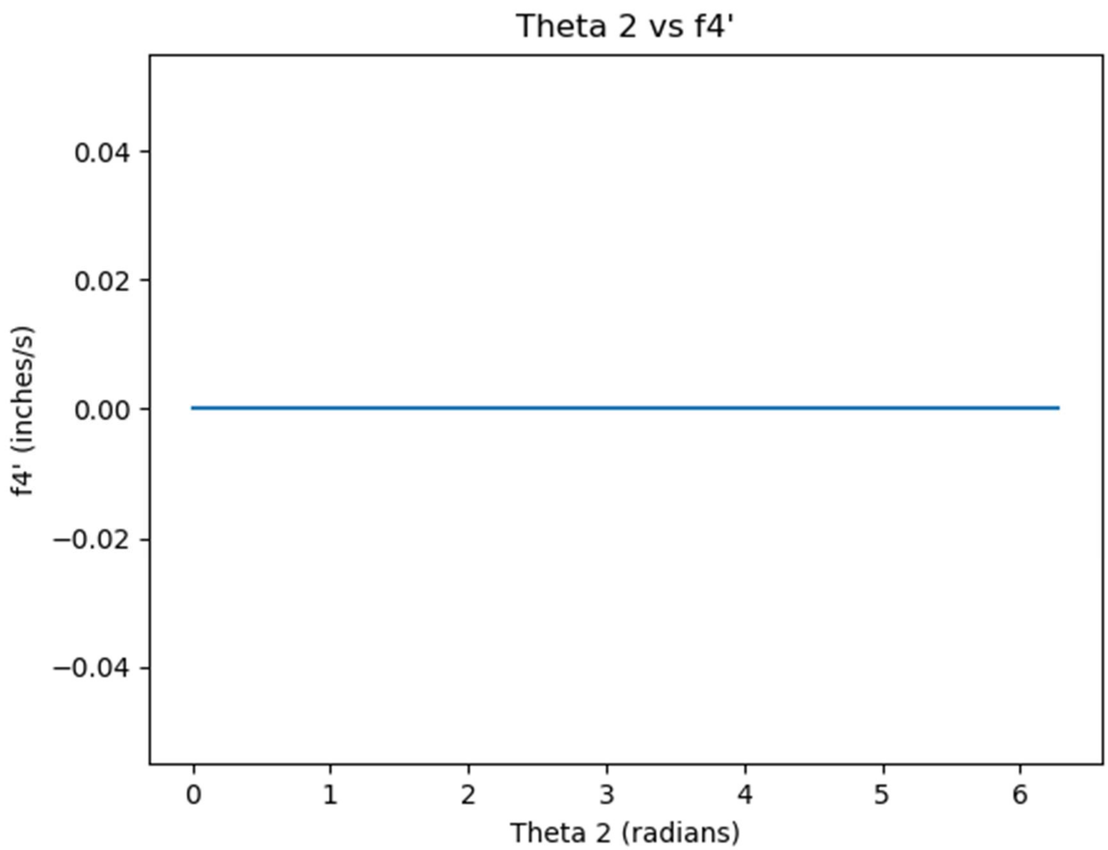
Jon Murders, Noah Ruby
ME 4133
Dr. Waggenspack
November 8th 2019

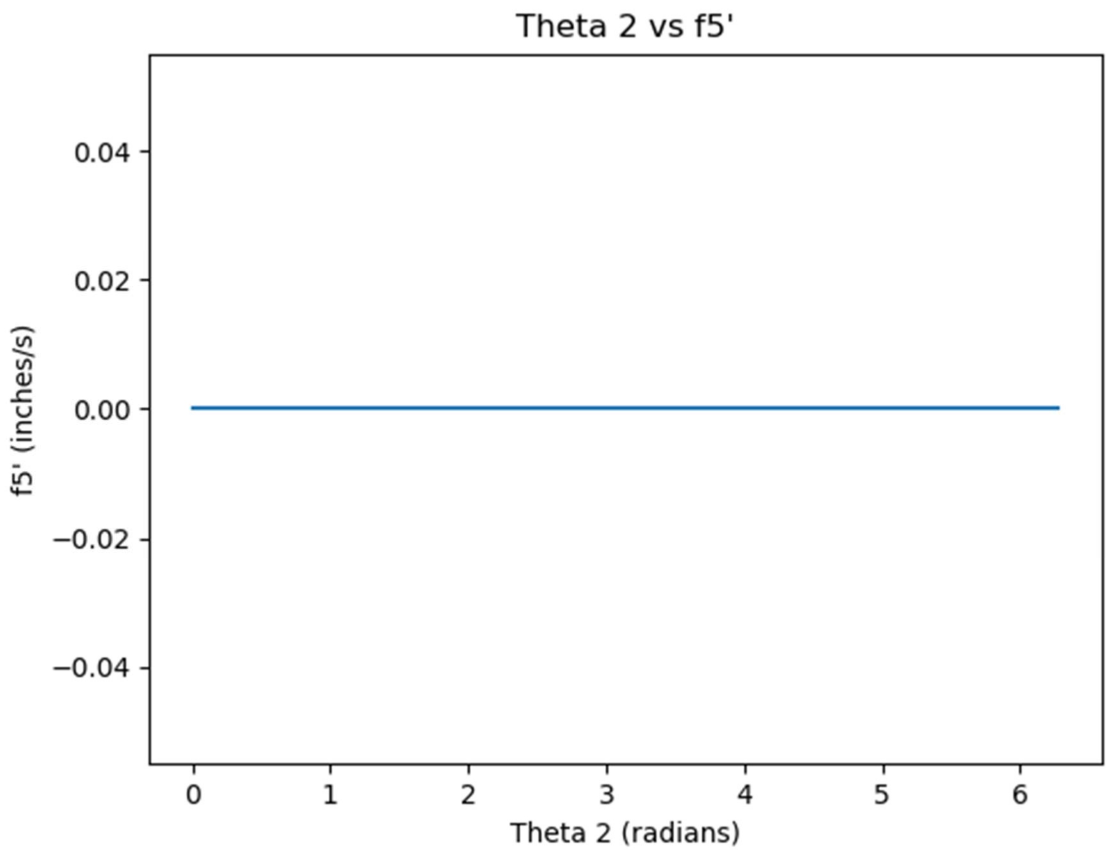
Up to this point we have found the positions and the kinematic coefficients of the mechanism, the second order kinematic coefficients and will continue to add core functions such as finding the inverse dynamic analysis. All these functions can have their output graphed and when the script is run it saves and titles all figures properly in the parent directory. You can have multiple mechanisms and use the script to run an analysis if the necessary inputs on the included CSV file are filled out. We were able to complete this using python 3.7 with the pandas module with the NumPy and math modules. The plan is to continue to develop to the final stage through the coming weekend and the beginning of the week.

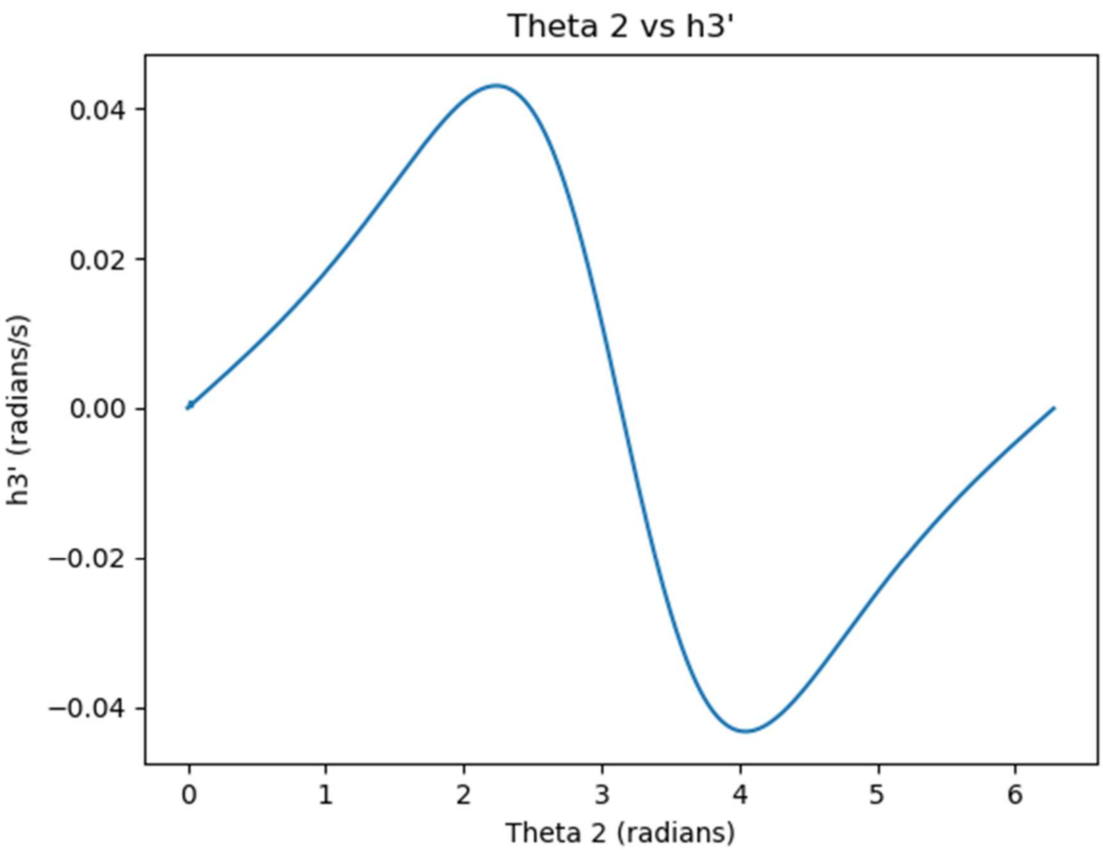
Points

- Positions Found
- First Order Kinematic Coefficients Found
- Second Order Kinematic Coefficients Found
- IDP Analysis In Progress (Target Completion: 11/12)
- CSV Input Complete

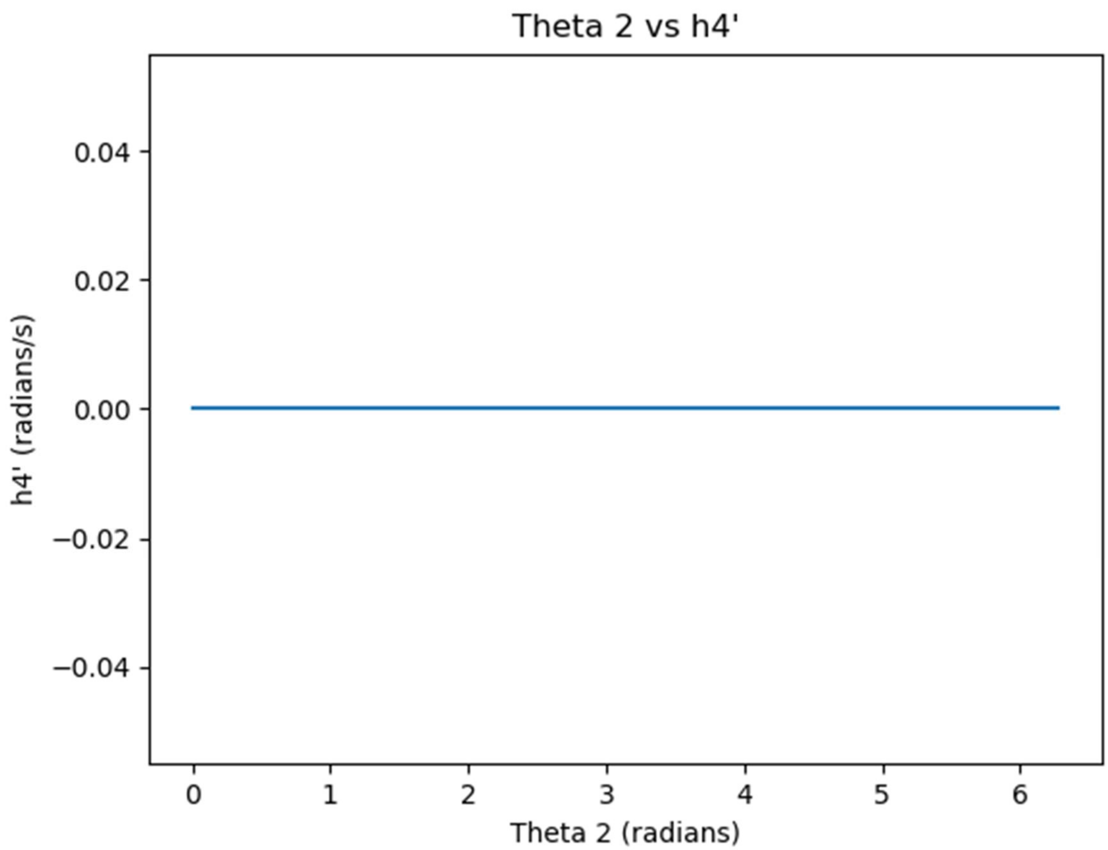

Theta 2 vs f3'
r3' (inches/s)
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs f4'
f4' (inches/s)
0.04
0.02
0.00
−0.02
−0.04
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs f5'
0.04
0.02
0.00
−0.02
−0.04
f5' (inches/s)
0
1
2
3
4
5
6
Theta 2 (radians)



Theta 2 vs h3'
0.04
0.02
0.00
−0.02
−0.04
h3' (radians/s)
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs h4'
h4' (radians/s)
0.04
0.02
0.00
−0.02
−0.04
0
1
2
3
4
5
6
Theta 2 (radians)

```
from math import pi
import math
import numpy as np
from numpy.linalg import inv
import pandas as pd
from matplotlib import pyplot as plt

sin = math.sin
cos = math.cos
tan = math.tan
#Bring In Input CSV
input = pd.read_csv('Input.csv')

#loading scalar knowns from input sheet
R1 = input.loc[0,'Value']  #inches - pg 96
R2 = input.loc[1,'Value'] #inches - pg 96
R6 = input.loc[5,'Value'] #inches - pg 96
theta_1 = input.loc[6,'Value']# pg 96
theta_5 = input.loc[10,'Value']# pg 96
theta_6 = input.loc[11,'Value']# pg 96

#Initial guess values from the input sheet
theta_3 = input.loc[8,'Value'] #radians
R3 = input.loc[2,'Value'] #inches
theta_4 = input.loc[9,'Value'] #radians
R4 = input.loc[3,'Value'] #inches
R5 =  input.loc[4,'Value'] #inches
x = np.array([theta_3,R3,theta_4,R4,R5], dtype=np.float)
#Input Setup from the input sheet
```

```
theta_2 = input.loc[7,'Value'] #radians

w2 = input.loc[20,'Value'] #radians per second

#Data collection table

positions =
pd.DataFrame(columns=['theta_2','theta_3','R3','theta_4','R4','R5','h3','f3','h4','f4','f5','h3p','f3p','h4p','f
4p','f5p'])

r=0 #row Counter


while  theta_2 < input.loc[19,'Value']:

   #finding the sines and cosines of all the angles

   ct2 = cos(theta_2)

   st2 = sin(theta_2)

   ch3 = cos(theta_3)

   sh3 = sin(theta_3)

   ct4 = cos(theta_4)

   st4 = sin(theta_4)

   ct5 = cos(theta_5)

   st5 = sin(theta_5)

   ct6 = cos(theta_6)

   st6 = sin(theta_6)

   #Loop Counter

   i = 0


   #Newton's Method Loop

   while i<100:



      #find the values of the VLEs provided on page 96

      f1 = R2*ct2-R3*ch3+R1
```

```
    f2 = R2*st2-R3*sh3
    f3 = R6-R4*ct4+R1
    f4 = -R5-R4*st4
    f5 = theta_4-theta_3
    f = [f1, f2, f3, f4, f5]
    fa = np.array(f,dtype=np.float)
    #finding the derivatives
    dfdh3 = np.array([[R3*sh3], [-R3*ch3],[0], [0], [-1]], dtype=np.float)
    dfdr3 = np.array([[-ch3], [-sh3], [0], [0], [0]], dtype=np.float)
    dfdt4 = np.array([[0], [0], [R4*st4], [-R4*ct4], [1]], dtype=np.float)
    dfdr4 = np.array([[0], [0], [-ct4], [-st4], [0]], dtype=np.float)
    dfdr5 = np.array([[0], [0], [0], [-1], [0]], dtype=np.float)
    #Making 5x5 array of derivatives
    A = np.hstack((dfdh3,dfdr3,dfdt4,dfdr4,dfdr5))
    #Takes the inverse of Matrix A
    ainv = inv(A)
    #Newton's Method Applied
    x = x-(ainv*fa)
    #Extracts values
    theta_3 = x[0][0]
    R3 = x[1][0]
    theta_4 = x[2][0]
    R4 = x[3][0]
    R5 = x[4][0]


    i+=1
#Logging Data into the table
positions.loc[r,'theta_2'] = theta_2
```

```
positions.loc[r,'theta_3'] = theta_3

positions.loc[r,'R3'] = R3

positions.loc[r,'theta_4'] = theta_4

positions.loc[r,'R4'] = R4

positions.loc[r,'R5'] = R5


# velocity coefficents matricies

B = np.array([[R3*sh3, 0 , -ch3, 0, 0],

        [-R3*ch3, 0, -sh3, 0, 0],

        [0, -ct4, 0, -ct4, ct5],

        [0, -st4, 0, -st4, st5],

        [-1, 1, 0, 0, 0,]], dtype=np.float)


C = np.array([[R2*st2],[-R2*ct2],[0],[0],[0]], dtype=np.float)


binv = inv(B)

y = binv*C

#storing kinematic coefficent values

positions.loc[r,'h3'] = y[0][0]

positions.loc[r,'f3'] = y[1][0]

positions.loc[r,'h4'] = y[2][0]

positions.loc[r,'f4'] = y[3][0]

positions.loc[r,'f5'] = y[4][0]

h3 = y[0][0]

f3 = y[1][0]

h4 = y[2][0]

f4 = y[3][0]

f5 = y[4][0]

#solving for second kinematic coefficents
```

```
    h3p = h3/w2
    h4p = h4/w2
    f3p = f3/w2
    f4p = f4/w2
    f5p = f5/w2
    #storing second kinematic coefficents
    positions.loc[r,'h3p'] = h3p
    positions.loc[r,'f3p'] = f3p
    positions.loc[r,'h4p'] = h4p
    positions.loc[r,'f4p'] = f4p
    positions.loc[r,'f5p'] = f5p

    theta_2 +=.01
    r += 1

#Generating t2 vs h3 plot
plt.figure(1)
plt.plot(positions.theta_2,positions.theta_3)
titlet3 = 'Theta 2 vs Theta3'
plt.title(titlet3)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('Theta 3 (radians)')
plt.savefig(titlet3)

#Generating t2 vs R3 plot
plt.figure(2)
plt.plot(positions.theta_2,positions.R3)
titler3 = 'Theta 2 vs Vector R3'
plt.title('Theta 2 vs Vector R3')
```

```
plt.xlabel('Theta 2 (radians)')
plt.ylabel('Vector R3 (inches)')
plt.savefig(titler3)

#Generating t2 vs t4 plot
plt.figure(3)
plt.plot(positions.theta_2,positions.theta_4)
titlet4 = 'Theta 2 vs Theta 4'
plt.title(titlet4)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('Theta 4 (radians)')
plt.savefig(titlet4)

#Generating t2 vs R4 plot
plt.figure(4)
plt.plot(positions.theta_2,positions.R4)
titler4 = 'Theta 2 vs Vector R4'
plt.title(titler4)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('Vector R4 (inches)')
plt.savefig(titler4)

#Generating t2 vs R5 plot
plt.figure(5)
plt.plot(positions.theta_2,positions.R5)
titler5 = 'Theta 2 vs Vector R5'
plt.title(titler5)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('Vector R5 (inches)')
```

```
plt.savefig(titler5)

#Generating t2 vs h3 plot
plt.figure(6)
plt.plot(positions.theta_2,positions.h3)
titleh3 = 'Theta 2 vs h3'
plt.title(titleh3)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('h3 (radians/s)')
plt.savefig(titleh3)

#Generating t2 vs R3 plot
plt.figure(7)
plt.plot(positions.theta_2,positions.f3)
titler7 = 'Theta 2 vs f3'
plt.title('Theta 2 vs f3')
plt.xlabel('Theta 2 (radians)')
plt.ylabel('r3 (inches/s)')
plt.savefig(titler7)

#Generating t2 vs t4 plot
plt.figure(8)
plt.plot(positions.theta_2,positions.h4)
titlet8 = 'Theta 2 vs h4'
plt.title(titlet4)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('h4 (radians/s)')
plt.savefig(titlet8)
```

```
#Generating t2 vs R4 plot
plt.figure(9)
plt.plot(positions.theta_2,positions.f4)
titler9 = 'Theta 2 vs f4'
plt.title(titler4)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('f4 (inches/s)')
plt.savefig(titler9)

#Generating t2 vs R5 plot
plt.figure(10)
plt.plot(positions.theta_2,positions.f5)
titler10 = 'Theta 2 vs f5'
plt.title(titler10)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('f5 (inches/s)')
plt.savefig(titler10)

#Generating t2 vs h3' plot
plt.figure(11)
plt.plot(positions.theta_2,positions.h3p)
titleh3p = 'Theta 2 vs h3\''
plt.title(titleh3p)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('h3\' (radians/s)')
plt.savefig(titleh3p)

#Generating t2 vs R3' plot
plt.figure(12)
```

```
plt.plot(positions.theta_2,positions.f3p)

titler3p = 'Theta 2 vs f3\''

plt.title(titler3p)

plt.xlabel('Theta 2 (radians)')

plt.ylabel('r3\' (inches/s)')

plt.savefig(titler3p)


#Generating t2 vs t4 plot

plt.figure(13)

plt.plot(positions.theta_2,positions.h4p)

titlet4p = 'Theta 2 vs h4\''

plt.title(titlet4p)

plt.xlabel('Theta 2 (radians)')

plt.ylabel('h4\' (radians/s)')

plt.savefig(titlet4p)


#Generating t2 vs R4 plot

plt.figure(14)

plt.plot(positions.theta_2,positions.f4p)

titler4p = 'Theta 2 vs f4\''

plt.title(titler4p)

plt.xlabel('Theta 2 (radians)')

plt.ylabel('f4\' (inches/s)')

plt.savefig(titler4p)


#Generating t2 vs R5 plot

plt.figure(15)

plt.plot(positions.theta_2,positions.f5p)

titler5p = 'Theta 2 vs f5\''
```

```
plt.title(titler5p)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('f5\' (inches/s)')
plt.savefig(titler5p)
```