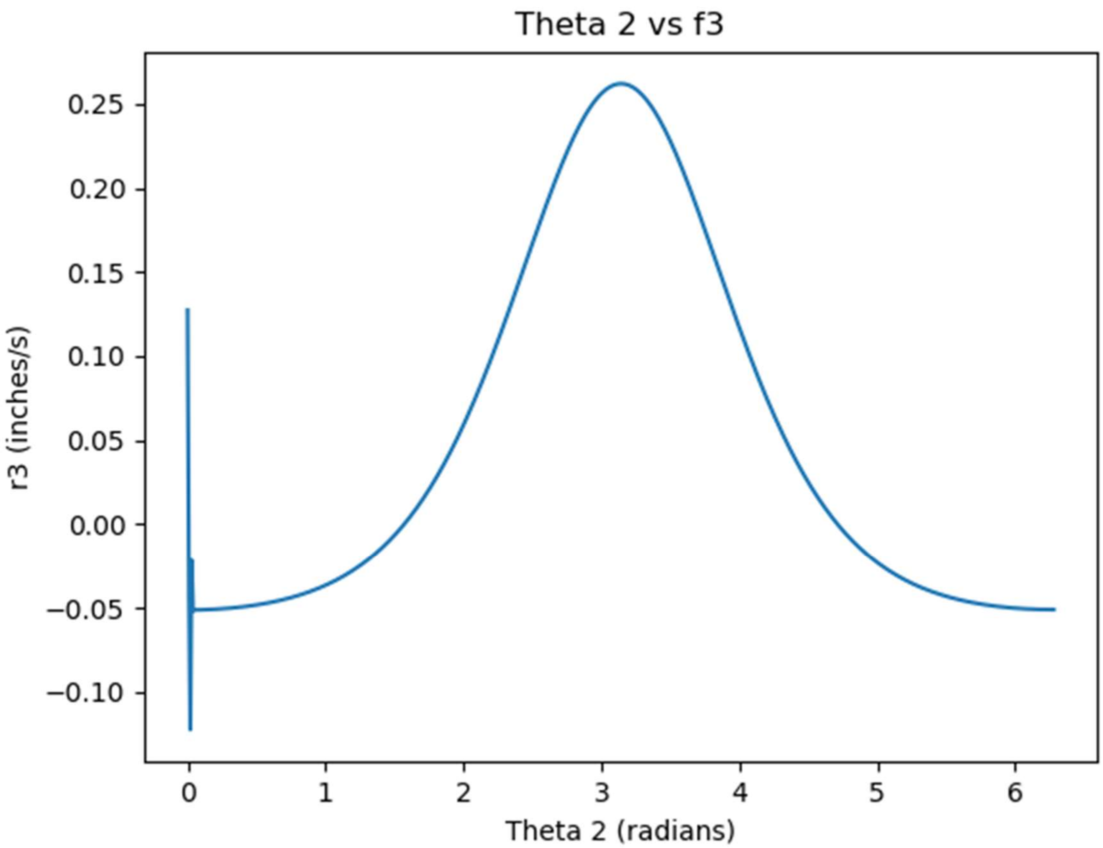
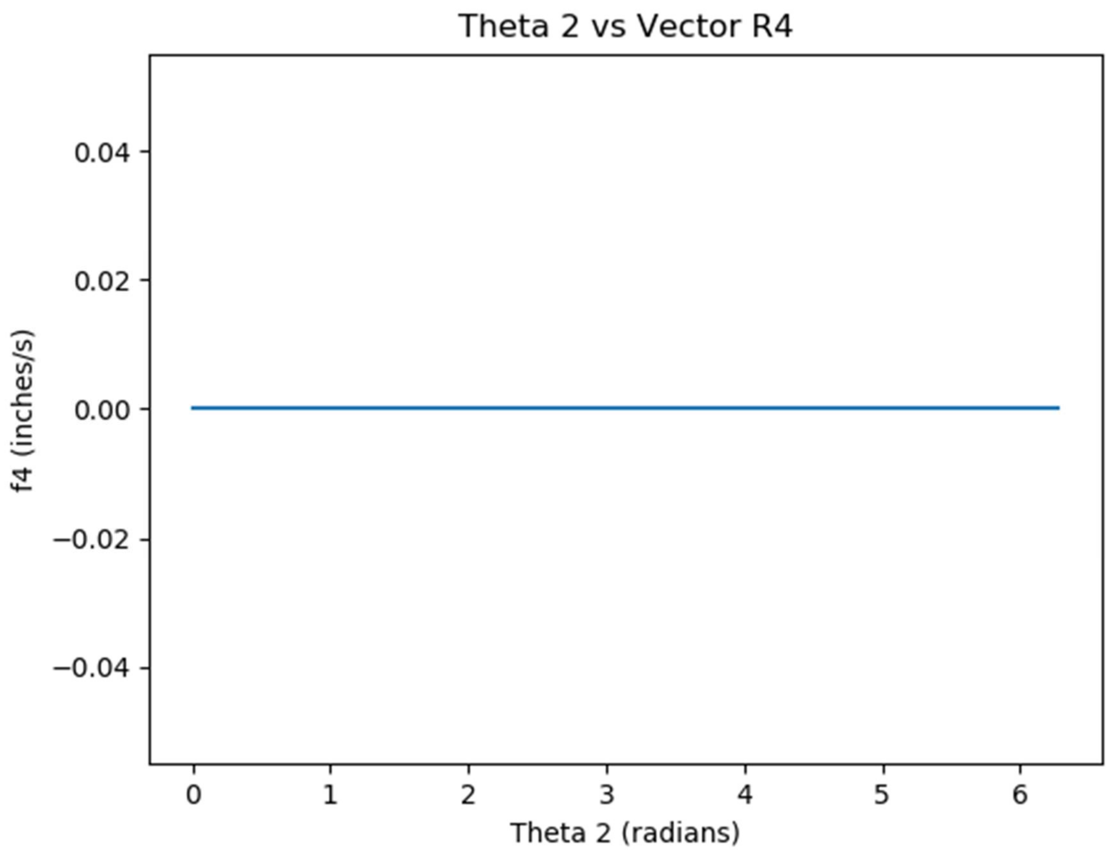
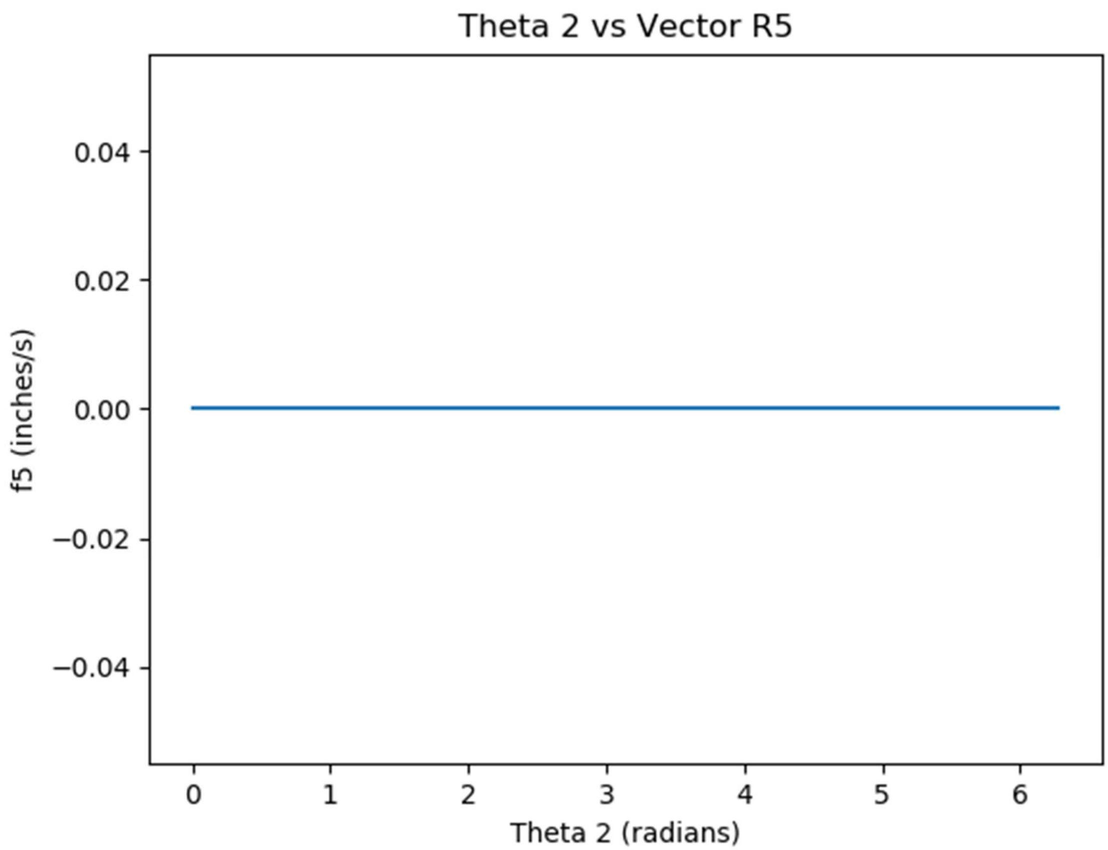
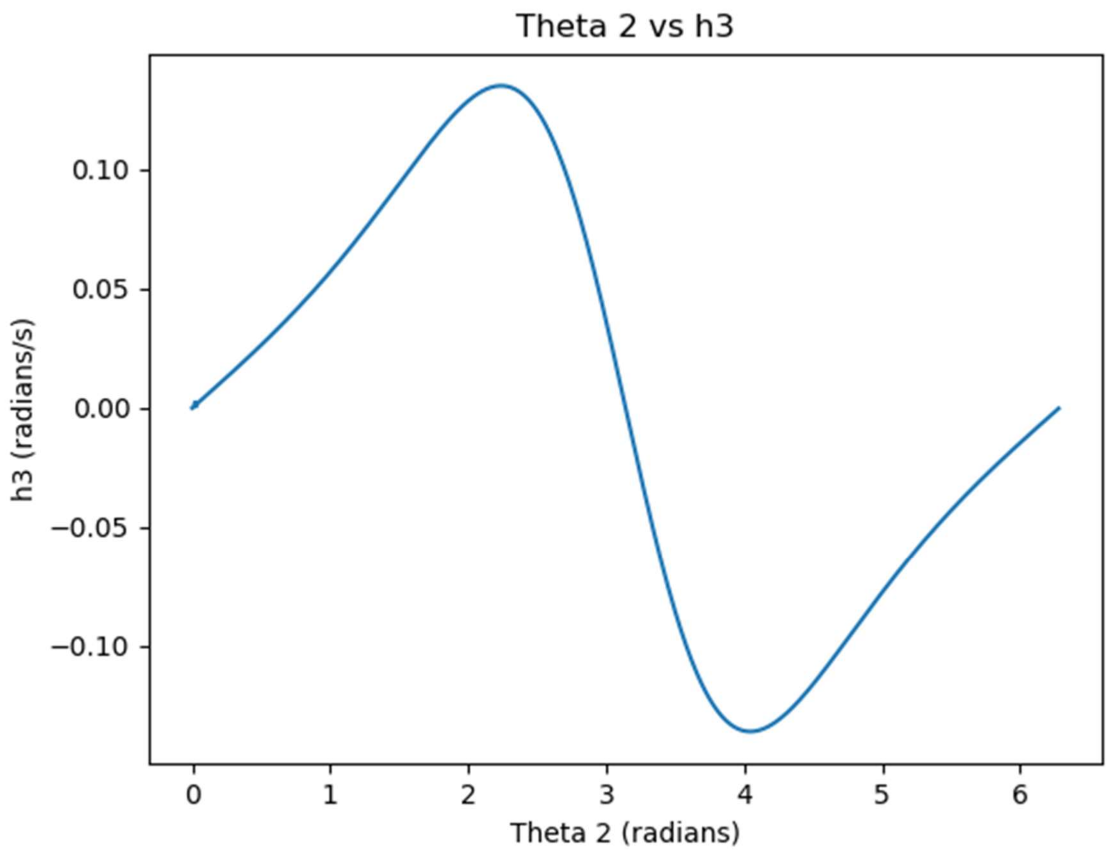
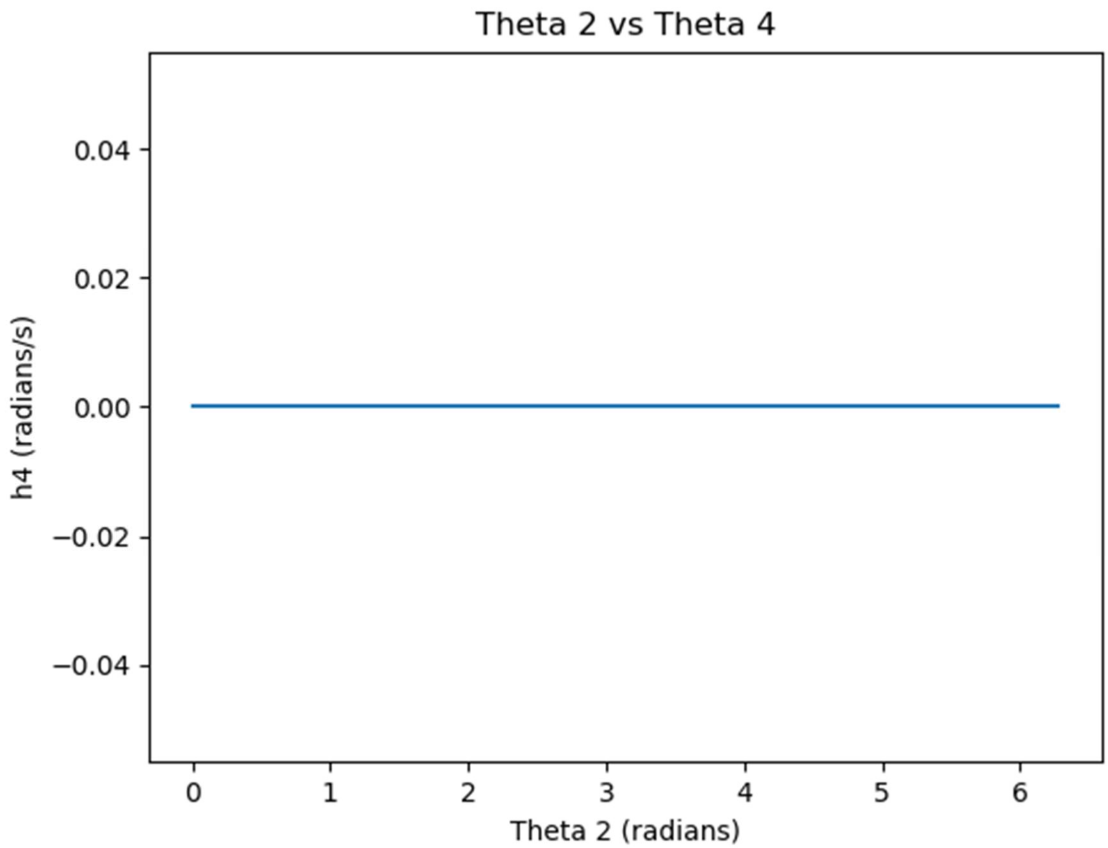

Theta 2 vs f3
r3 (inches/s)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
0
1
2
3
4
5
6
Theta 2 (radians)

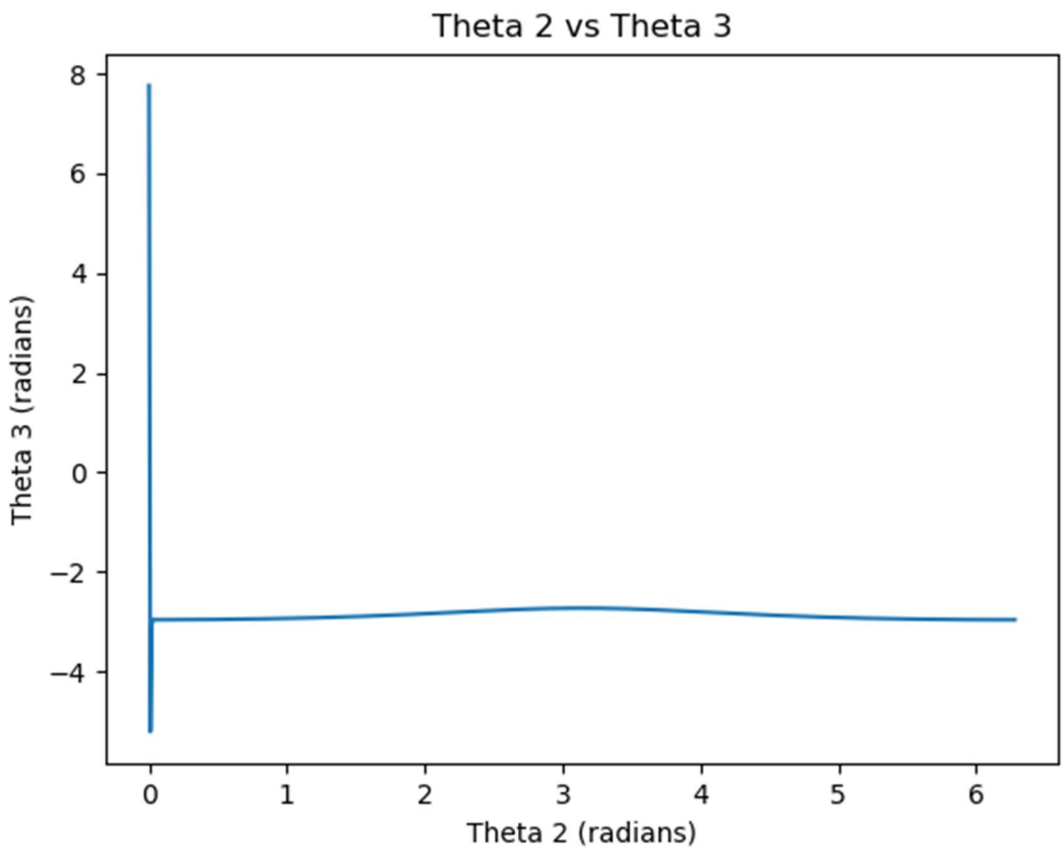

Theta 2 vs Vector R4
f4 (inches/s)
0.04
0.02
0.00
−0.02
−0.04
0
1
2
3
4
5
6
Theta 2 (radians)

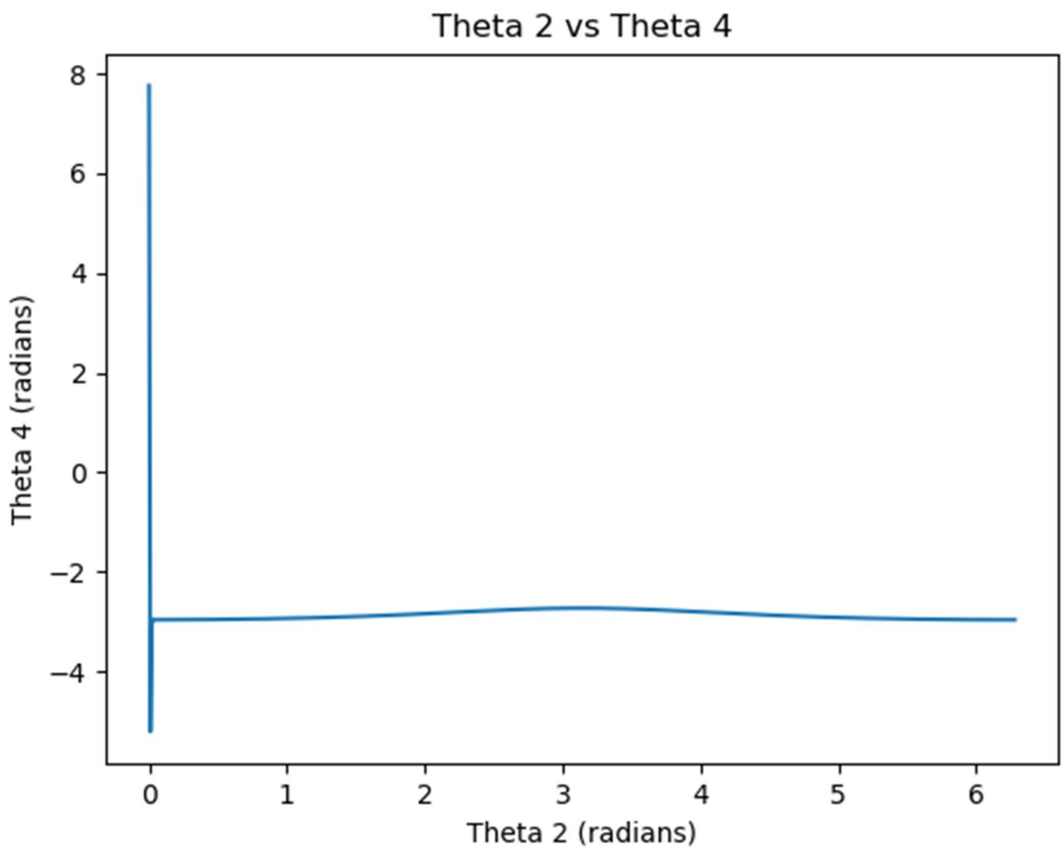


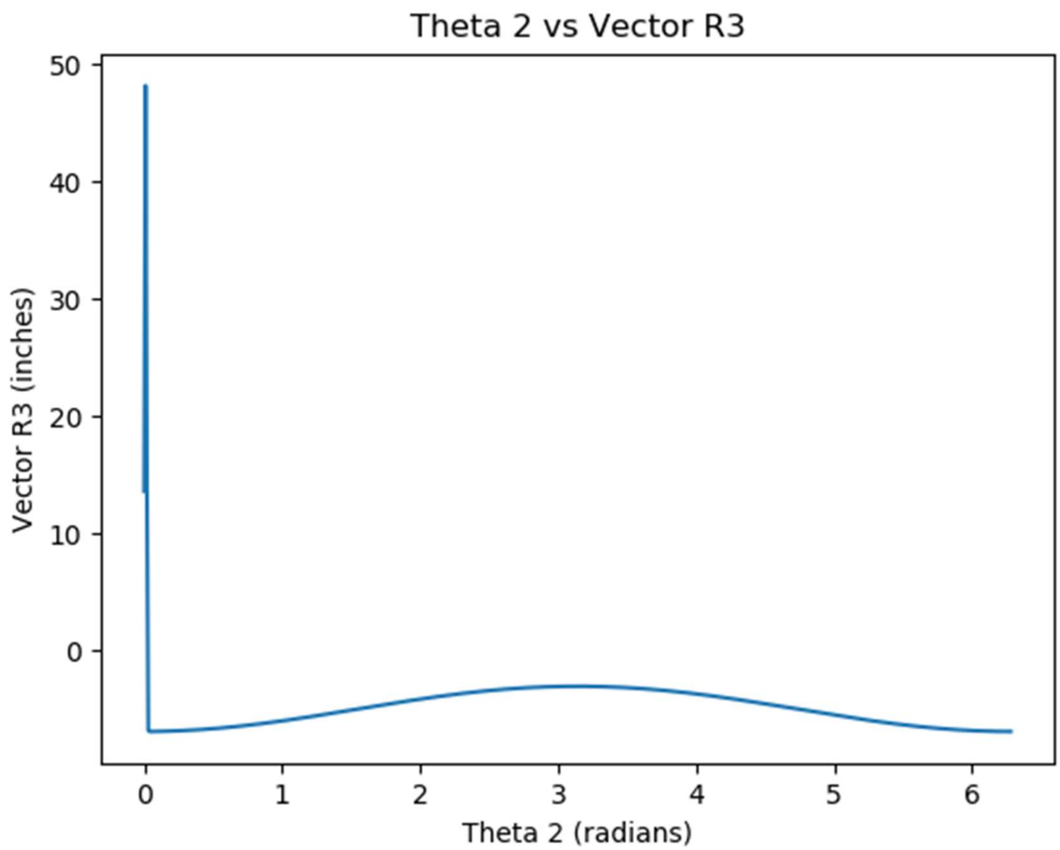
Theta 2 vs Vector R5
0.04
0.02
0.00
−0.02
−0.04
f5 (inches/s)
0
1
2
3
4
5
6
Theta 2 (radians)

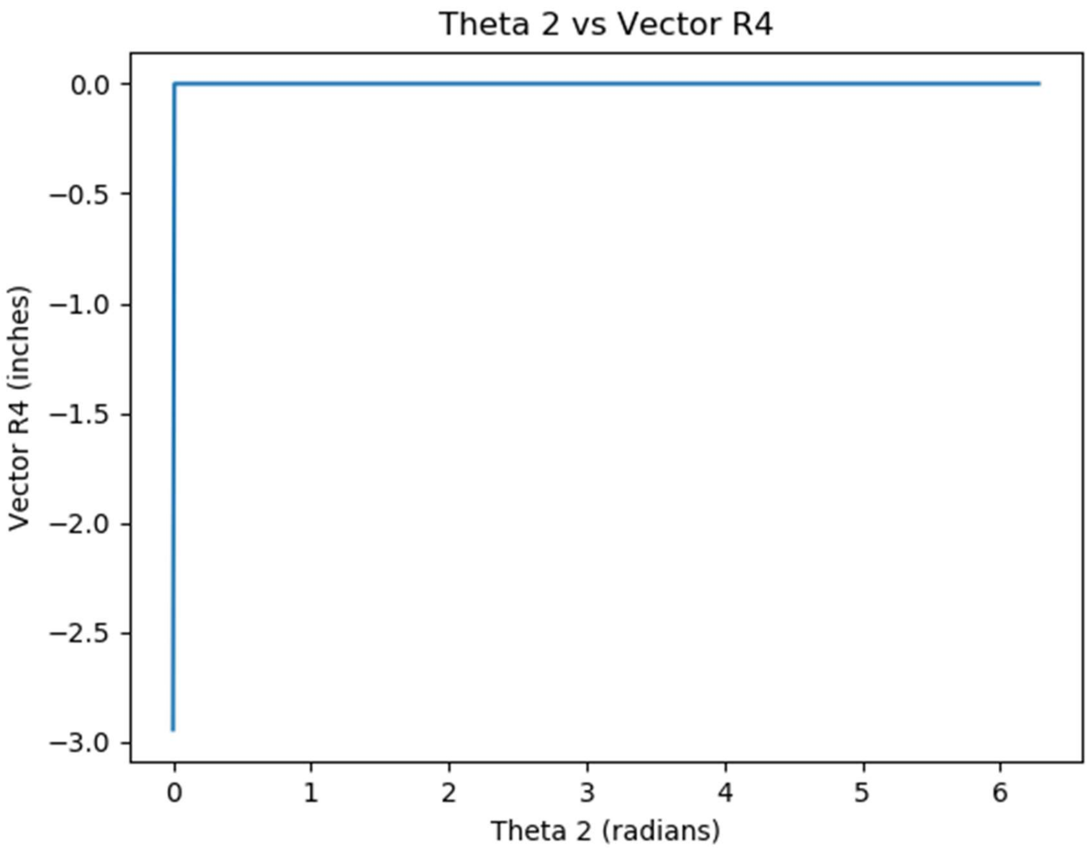

Theta 2 vs h3
h3 (radians/s)
0.10
0.05
0.00
−0.05
−0.10
0
1
2
3
4
5
6
Theta 2 (radians)

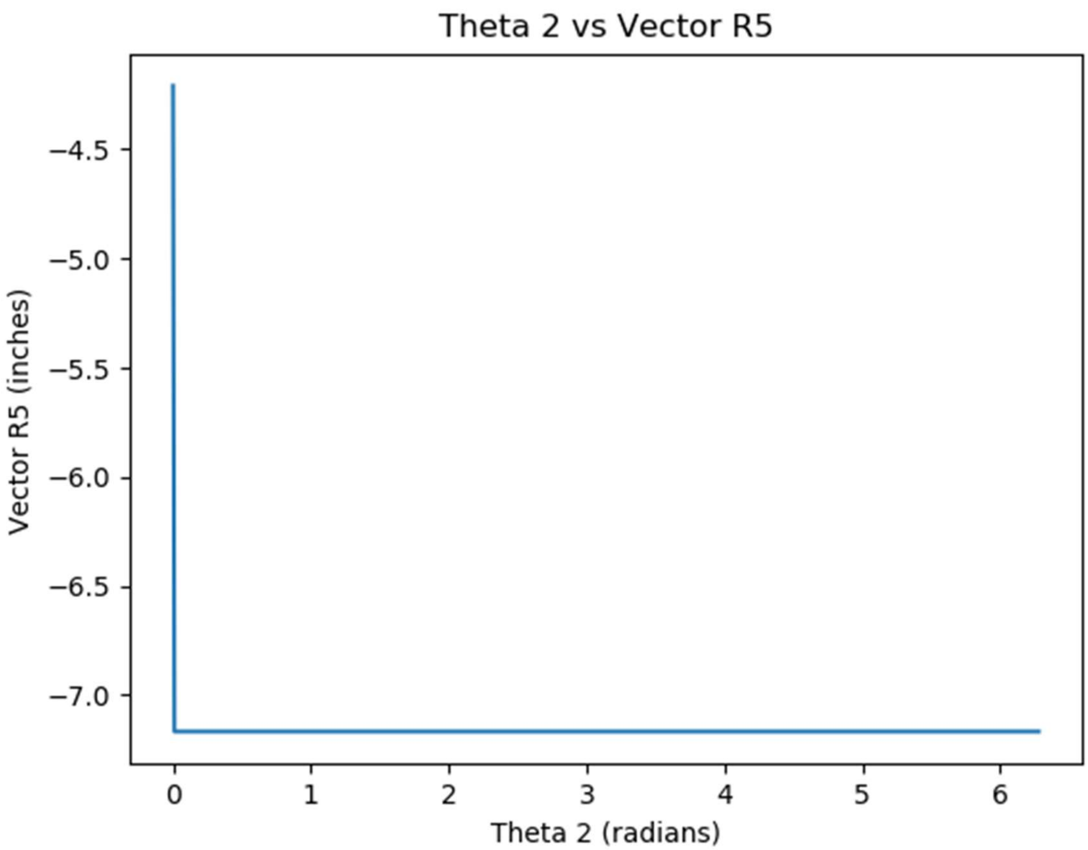

Theta 2 vs Theta 4
0.04
0.02
0.00
−0.02
−0.04
h4 (radians/s)
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs Theta 3
8
6
4
2
0
−2
−4
Theta 3 (radians)
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs Theta 4
8
6
4
2
0
−2
−4
Theta 4 (radians)
0
1
2
3
4
5
6
Theta 2 (radians)


Theta 2 vs Vector R3
Vector R3 (inches)
Theta 2 (radians)
50
40
30
20
10
0
0
1
2
3
4
5
6


Theta 2 vs Vector R4
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
Vector R4 (inches)
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


#scalar knowns

R1 = 4.8 #inches - pg 96
```

```
R2 = 2 #inches - pg 96

R6 = 3.65 #inches - pg 96

theta_1 = -(pi)# pg 96

theta_5 = -(pi)/2# pg 96

theta_6 = 0# pg 96

o2 = 3.14

#Initial guess values

theta_3 = 5.236 #inches

R3 = 3 #inches

theta_4 = 5.236 #inches

R4 = 11 #inches

R5 =  4 #inches

x = np.array([theta_3,R3,theta_4,R4,R5], dtype=np.float)

#Input Angle

theta_2 = 0 #radians


#Data collection table

positions = pd.DataFrame(columns=['theta_2','R3','theta_4','R4','R5'])

r=0 #row Counter


while theta_2 < 6.28:

    #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
```

```
    positions.loc[r,'h3'] = y[0][0]
    positions.loc[r,'r3'] = y[1][0]
    positions.loc[r,'h4'] = y[2][0]
    positions.loc[r,'r4'] = y[3][0]
    positions.loc[r,'r5'] = y[4][0]
    theta_2 +=.01
    r += 1
#Generating t2 vs h3 plot
plt.figure(1)
plt.plot(positions.theta_2,positions.theta_3)
titleh3 = 'Theta 2 vs h3'
plt.title(titleh3)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('h3 (radians)')
plt.savefig(titleh3)

#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.r3)
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.r4)
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.r5)
titler10 = 'Theta 2 vs f5'
plt.title(titler5)
plt.xlabel('Theta 2 (radians)')
plt.ylabel('f5 (inches/s)')
plt.savefig(titler10)
```