Scara Robot

Vpython code:

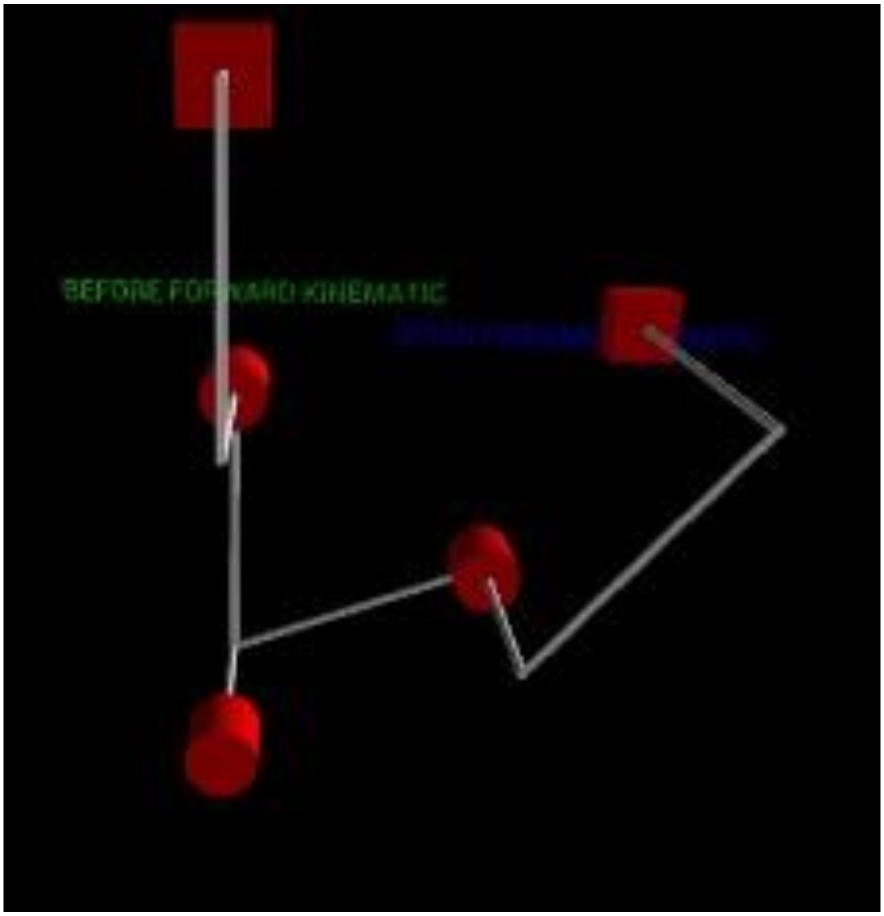
```
#150114074/TUNAHAN AYDIN

from vpython import *

from numpy import *

#Creating variables for rotation.

#Calculating forward kinematics for Scara Robot.

print("Enter your theta degree for motor 1:(Q1)")

theta_motor1 = int(input())#degree of motor 1 rotation

print("Enter your theta degree for motor 2:(Q2)")

theta_motor2 = int(input())#degree of motor 2 rotation

print("Enter your theta displacement d value less than 1.59: ")

dis_cube = double(input())#degree of motor 2 rotation

#for d3 motion.

if dis_cube > 1.59:

    L = label()

    L.text = "ENTER VALID DISPLACEMENT VALUE OF CUBE."

cyl4 = cylinder( pos=vector(1-0.15,1,0),axis=vector(1,0,0),color=color.red,radius=0.12,length=0.3)

c1c = curve(vector(0,0,0), vector(1,0,0))#a1=1

c2c = curve(vector(1,1,0), vector(1,0,0))#a2=1

c3c = curve(vector(1,1,0), vector(0,1,0))#a3=1

c4c = curve(vector(0,1,0), vector(0,2,0))#a4=1

c5c = curve(vector(0,2,0), vector(1,2,0))#a5=1

c6c = curve(vector(1,2,0), vector(2,2,0))#a6=1

T = text(pos=vector(3,1,0),axis=vector(0,0,1),text='BEFORE FORWARD KINEMATIC', align='center',
color=color.green,height =0.1)

#cube after forward kinematic

cube = box(pos=vector(0.18,2,0),axis=vector(0,0,0),length=0.25,height=0.25,
width=0.25,color=color.red)
```

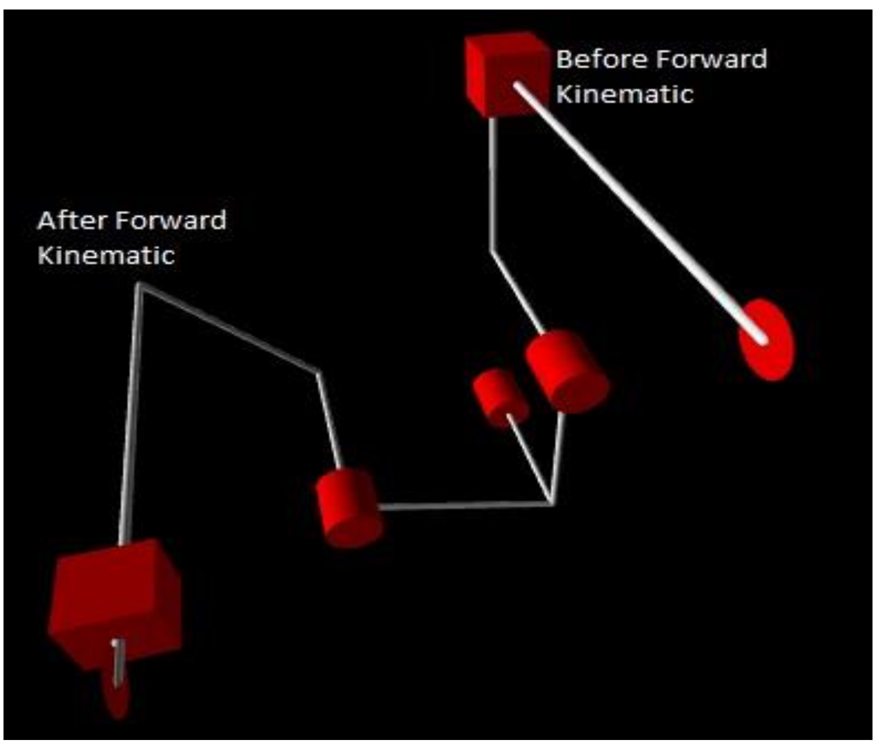
```
#cube before forward kinematic.

cube2 = box(pos=vector(0.18,2,0),axis=vector(0,0,0),length=0.25,height=0.25,
width=0.25,color=color.red)

circle1 = shapes.circle(radius=0.1)

circle_path=[vector(2,2,0),vector(2,2,0.0001)]

crc=extrusion(path=circle_path,shape=circle1,color=color.red)

#FORWARD KINEMATICS

# Link lengths between joints

a1 = 1    # length of link a1

a2 = 1    # length of link a2

a3 = 1    # length of link a3

a4 = 1    # length of link a4

# Angles in radians

theta1 = (theta_motor1/180)*pi # theta 1 in radians for joint 1

theta2 = (theta_motor2/180)*pi # theta 2 in radians for joint 2

PT = [[theta1, 0, a2, a1],

      [theta2, 0, a4, a3]]

# Rotation Matrices H01 and H12

i = 0

H0_1 = [[cos(PT[i][0]), -sin(PT[i][0])*cos(PT[i][1]), sin(PT[i][0])*sin(PT[i][1]), PT[i][2]*cos(PT[i][0])],

        [sin(PT[i][0]), cos(PT[i][0])*cos(PT[i][1]), -cos(PT[i][0])*sin(PT[i][1]), PT[i][2]*sin(PT[i][0])],

        [0, sin(PT[i][1]), cos(PT[i][1]), PT[i][3]],

        [0, 0, 0, 1]]

i = 1

H1_2 = [[cos(PT[i][0]), -sin(PT[i][0])*cos(PT[i][1]), sin(PT[i][0])*sin(PT[i][1]), PT[i][2]*cos(PT[i][0])],

        [sin(PT[i][0]), cos(PT[i][0])*cos(PT[i][1]), -cos(PT[i][0])*sin(PT[i][1]), PT[i][2]*sin(PT[i][0])],

        [0, sin(PT[i][1]), cos(PT[i][1]), PT[i][3]],

        [0, 0, 0, 1]]

print("H0_1 =")
```

```
print(matrix(H0_1))

#Forward kinematic calculation

H0_2 = dot(H0_1,H1_2)

print("Forward Kinematic,dot product of H01 * H12 = H0_2 =")

print(matrix(H0_2))

cyl1 = cylinder( pos=vector(0,0,0),axis=vector(1,0,0),color=color.red,radius=0.12,length=0.3)

c1 = curve(vector(0,0,0), vector(1,0,0))#a1=1

c2 = curve(vector(1,0,0), vector(H0_1[2][3],H0_1[1][3],H0_1[0][3]))

cyl2 = cylinder(
pos=vector(1-0.15,H0_1[1][3],H0_1[0][3]),axis=vector(1,0,0),color=color.red,radius=0.12,length=0.3)

c3 = curve(vector(1,H0_1[1][3],H0_1[0][3]), vector(0,H0_1[1][3],H0_1[0][3]))#a3=1

c4 = curve(vector(0,H0_1[1][3],H0_1[0][3]),vector(0,H0_2[1][3],H0_2[0][3]))#a4=1

c5 = curve(vector(0,H0_2[1][3],H0_2[0][3]),vector(1,H0_2[1][3],H0_2[0][3]))#a5=1

cube.pos=vector(0.18+dis_cube,H0_2[1][3],H0_2[0][3])

T = text(pos=vector(3,1-0.2,H0_2[0][3]),axis=vector(0,0,1),text='AFTER FORWARD KINEMATIC',
align='center', color=color.blue,height =0.1)

c6 = curve(vector(1,H0_2[1][3],H0_2[0][3]), vector(2,H0_2[1][3],H0_2[0][3]))#a5=1

circle_path2=[vector(2,H0_2[1][3],H0_2[0][3]), vector(2,H0_2[1][3],H0_2[0][3]+0.0001)]

extrusion(path=circle_path2,shape=circle1,color=color.red)
```

OUTPUTS

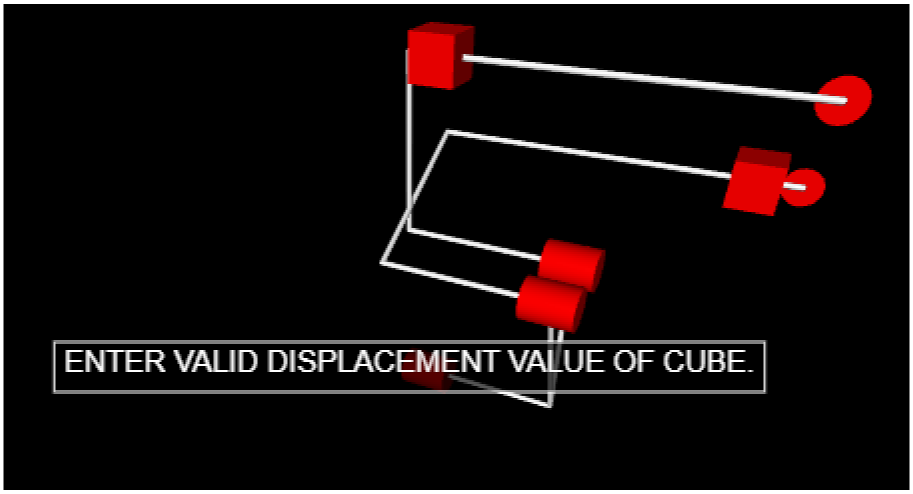
Design of articular robot:


BEFORE FORWARD KINEMATIC

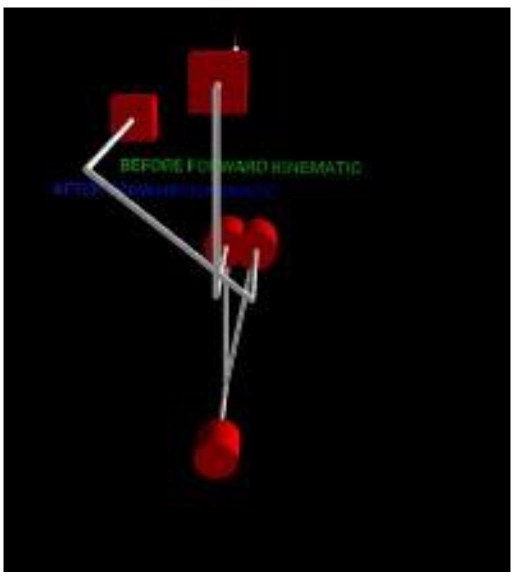

Before Forward
Kinematic
After Forward
Kinematic

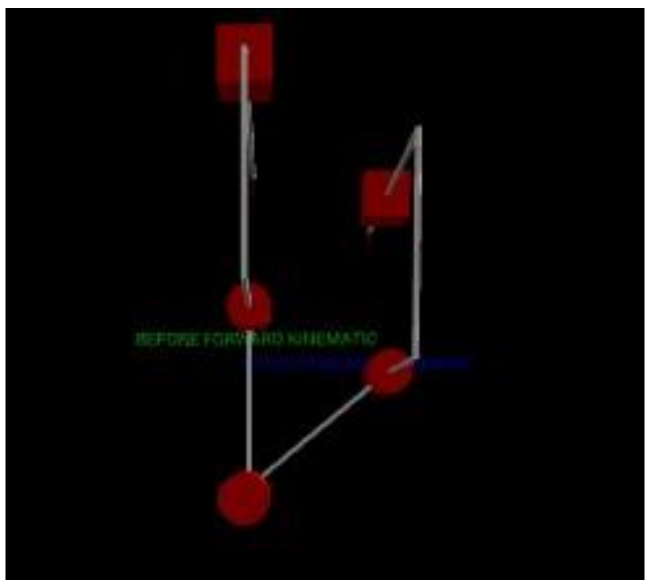
If user enters displacement is greater than 1.59, we will see an error message.

1.59 is range of displacement.

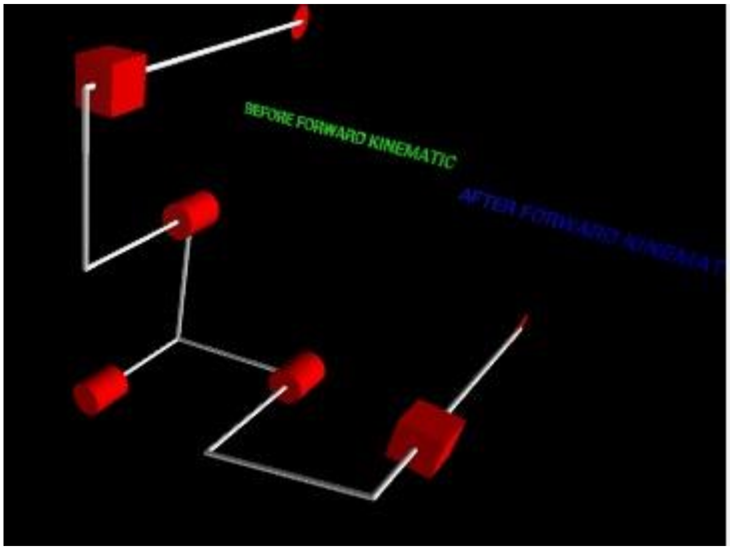


Theta 1=80,Theta 2=60,displacement=1.4 and Theta 1= 40,Theta 2=50, displacement=1





Theta 1=10,Theta 2=10,displacement=0.4



Forward kinematics calculation:

```
Enter your theta degree for motor 1:

20
Enter your theta degree for motor 2:

30
Enter your theta displacement d value less than 1.59:

1.57
H0_1 =
[[ 0.93969262 -0.34202014  0.          0.93969262]
 [ 0.34202014  0.93969262 -0.          0.34202014]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
H0_2 =
[[ 0.64278761 -0.76604444  0.          1.58248023]
 [ 0.76604444  0.64278761  0.          1.10806459]
 [ 0.          0.          1.          2.        ]
 [ 0.          0.          0.          1.        ]]
```

```
Enter your theta degree for motor 1:(Q1)

10
Enter your theta degree for motor 2:(Q2)

10
Enter your theta displacement d value less than 1.59:

0.4
H0_1 =
[[ 0.98480775 -0.17364818  0.          0.98480775]
 [ 0.17364818  0.98480775 -0.          0.17364818]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
Forward Kinematic,dot product of H01 * H12 = H0_2 =
[[ 0.93969262 -0.34202014  0.          1.92450037]
 [ 0.34202014  0.93969262  0.          0.51566832]
 [ 0.          0.          1.          2.        ]
 [ 0.          0.          0.          1.        ]]
```

Articular Robot

Vpython code:

#150114074 TUNAHAN AYDIN

from vpython import *

from numpy import *

#Calculating forward kinematics for Articular Robot.

print("Enter your theta degree for motor 1: ")

```
theta_motor1 = int(input())#degree of motor 1 rotation

print("Enter your theta degree for motor 2: ")

theta_motor2 = int(input())#degree of motor 2 rotation

print("Enter your theta degree for motor 3: ")

theta_motor3 = int(input())#degree of motor 3 rotation

#Creating cylinders for showing motors.

cyl1 = cylinder( pos=vector(0,0,0),axis=vector(0,1,0),color=color.red,radius=0.12,length=0.3)

cyl2 = cylinder( pos=vector(-0.15,1,0),axis=vector(1,0,0),color=color.red,radius=0.12,length=0.3)

cyl3 = cylinder( pos=vector(-0.15,1,2),axis=vector(1,0,0),color=color.red,radius=0.12,length=0.3)

#Curves show links between joints.

c1 = curve(vector(0,0,0), vector(0,1,0))

c2 = curve(vector(-0.15,1,0), vector(-0.15,1,1))

c3 = curve(vector(0.15,1,0), vector(0.15,1,1))

c4 = curve(vector(+0.15,1,1), vector(-0.15,1,1))

c5 = curve(vector(0,1,1), vector(0,1,2))

c6 = curve(vector(0.15,1,2), vector(0.15,1,3))

c7 = curve(vector(-0.15,1,2), vector(-0.15,1,3))

c8 = curve(vector(+0.15,1,3), vector(-0.15,1,3))

c9 = curve(vector(0,1,3), vector(0,1,4))

#Creating variables for rotation.


# Link lengths

a1 = 1    # length of link a1 by calculating vector length

a2 = 1    # length of link a2 by calculating vector length

a3 = 1 # length of link a3 by calculating vector length

a4 = 1    # length of link a4 by calculating vector length

a5 = 1 # length of link a5 by calculating vector length

a6 = 1    # length of link a6 by calculating vector length

#Creating radians variables for calculating sin and cos values.
```

```
thetaRa1 = (theta_motor1/180)*pi # radians value of degree for motor1

thetaRa2 = (theta_motor2/180)*pi # radians value of degree for motor2

thetaRa3 = (theta_motor3/180)*pi  # radians value of degree for motor3

# Creating matrices

PT = [[thetaRa1, 0, a2, a1],

      [thetaRa2, 0, a4, a3],

      [thetaRa3, 0, a6, a5]]

# homogeneous rotation forward kinematics matrices and multiplication of matrices H01*H12*H23=H03

i = 0

H0_1 = [[cos(PT[i][0]), -sin(PT[i][0])*cos(PT[i][1]), sin(PT[i][0])*sin(PT[i][1]), PT[i][2]*cos(PT[i][0])],

        [sin(PT[i][0]), cos(PT[i][0])*cos(PT[i][1]), -cos(PT[i][0])*sin(PT[i][1]), PT[i][2]*sin(PT[i][0])],

        [0, sin(PT[i][1]), cos(PT[i][1]), PT[i][3]],

        [0, 0, 0, 1]]

i = 1

H1_2 = [[cos(PT[i][0]), -sin(PT[i][0])*cos(PT[i][1]), sin(PT[i][0])*sin(PT[i][1]), PT[i][2]*cos(PT[i][0])],

        [sin(PT[i][0]), cos(PT[i][0])*cos(PT[i][1]), -cos(PT[i][0])*sin(PT[i][1]), PT[i][2]*sin(PT[i][0])],

        [0, sin(PT[i][1]), cos(PT[i][1]), PT[i][3]],

        [0, 0, 0, 1]]

i = 2

H2_3 = [[cos(PT[i][0]), -sin(PT[i][0])*cos(PT[i][1]), sin(PT[i][0])*sin(PT[i][1]), PT[i][2]*cos(PT[i][0])],

        [sin(PT[i][0]), cos(PT[i][0])*cos(PT[i][1]), -cos(PT[i][0])*sin(PT[i][1]), PT[i][2]*sin(PT[i][0])],

        [0, sin(PT[i][1]), cos(PT[i][1]), PT[i][3]],

        [0, 0, 0, 1]]

H0_2 = dot(H0_1,H1_2)

H0_3 = dot(H0_2,H2_3)

print("H0_1 =")

print(matrix(H0_1))

print("H1_2 =")

print(matrix(H1_2))
```

print("H2_3 =")

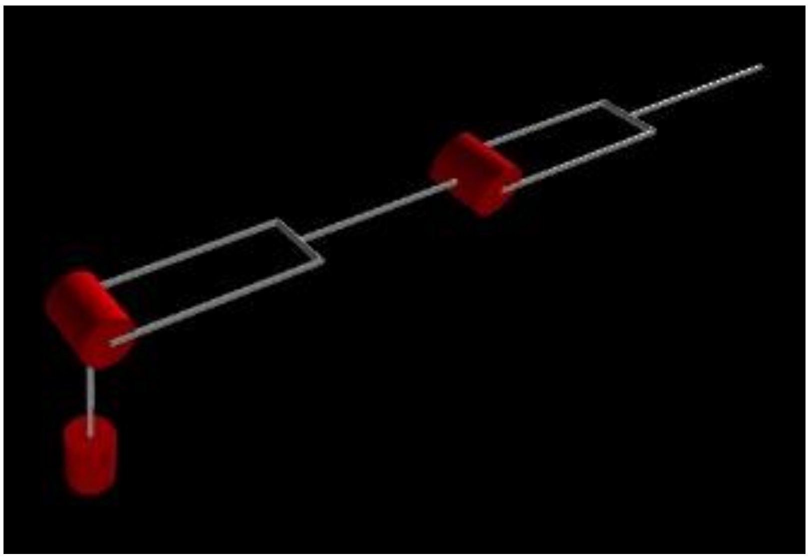
print(matrix(H2_3))

print("Forward Kinematic :dot multiplication of H0_1 * H1_2 * H2_3 = H0_3 =")

print(matrix(H0_3))

OUTPUTS

Design of articular robot:



Output of calculation for forward kinematics:

```
Enter your theta degree for motor 1:

10
Enter your theta degree for motor 2:

20
Enter your theta degree for motor 3:

30
H0_1 =
[[ 0.98480775 -0.17364818  0.          0.98480775]
 [ 0.17364818  0.98480775 -0.          0.17364818]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
H1_2 =
[[ 0.93969262 -0.34202014  0.          0.93969262]
 [ 0.34202014  0.93969262 -0.          0.34202014]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
H2_3 =
[[ 0.8660254 -0.5        0.         0.8660254]
 [ 0.5        0.8660254 -0.         0.5      ]
 [ 0.         0.         1.         1.       ]
 [ 0.         0.         0.         1.       ]]
Forward Kinematic :dot multiplication of H0_1 * H1_2 * H2_3 = H0_3 =
[[ 0.5        -0.8660254   0.          2.35083316]
 [ 0.8660254   0.5         0.          1.53967358]
 [ 0.          0.          1.          3.        ]
 [ 0.          0.          0.          1.        ]]
```

```
Enter your theta degree for motor 1:

50
Enter your theta degree for motor 2:

40
Enter your theta degree for motor 3:

100
H0_1 =
[[ 0.64278761 -0.76604444  0.          0.64278761]
 [ 0.76604444  0.64278761 -0.          0.76604444]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
H1_2 =
[[ 0.76604444 -0.64278761  0.          0.76604444]
 [ 0.64278761  0.76604444 -0.          0.64278761]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
H2_3 =
[[-0.17364818 -0.98480775  0.         -0.17364818]
 [ 0.98480775 -0.17364818  0.          0.98480775]
 [ 0.          0.          1.          1.        ]
 [ 0.          0.          0.          1.        ]]
Forward Kinematic :dot multiplication of H0_1 * H1_2 * H2_3 = H0_3 =
[[-0.98480775  0.17364818  0.         -0.34202014]
 [-0.17364818 -0.98480775  0.          1.59239627]
 [ 0.          0.          1.          3.        ]
 [ 0.          0.          0.          1.        ]]
```