Diagram, schematic

Description automatically generated

import numpy

rows = 4

cols = 4

sP = [[0 for \_ in range(cols)] for \_ in range(rows)]

sP[0][0], sP[0][1], sP[0][2], sP[0][3] = -1.3840, -0.9608, 1.3250, -1.3140

sP[1][0], sP[1][1], sP[1][2], sP[1][3] = 4.5620, 1.3110, -2.3890, 0.2501

sP[2][0], sP[2][1], sP[2][2], sP[2][3] = -0.1280, -1.6280, 1.7020, -0.7620

sP[3][0], sP[3][1], sP[3][2], sP[3][3] = 1, 1, 1, 1

oP = [[0 for \_ in range(cols)] for \_ in range(rows)]

oP[0][0], oP[0][1], oP[0][2], oP[0][3] = 2, 0, -1, -1

oP[1][0], oP[1][1], oP[1][2], oP[1][3] = 3, 0, -2, 0

oP[2][0], oP[2][1], oP[2][2], oP[2][3] = -3, -3, 2, -2

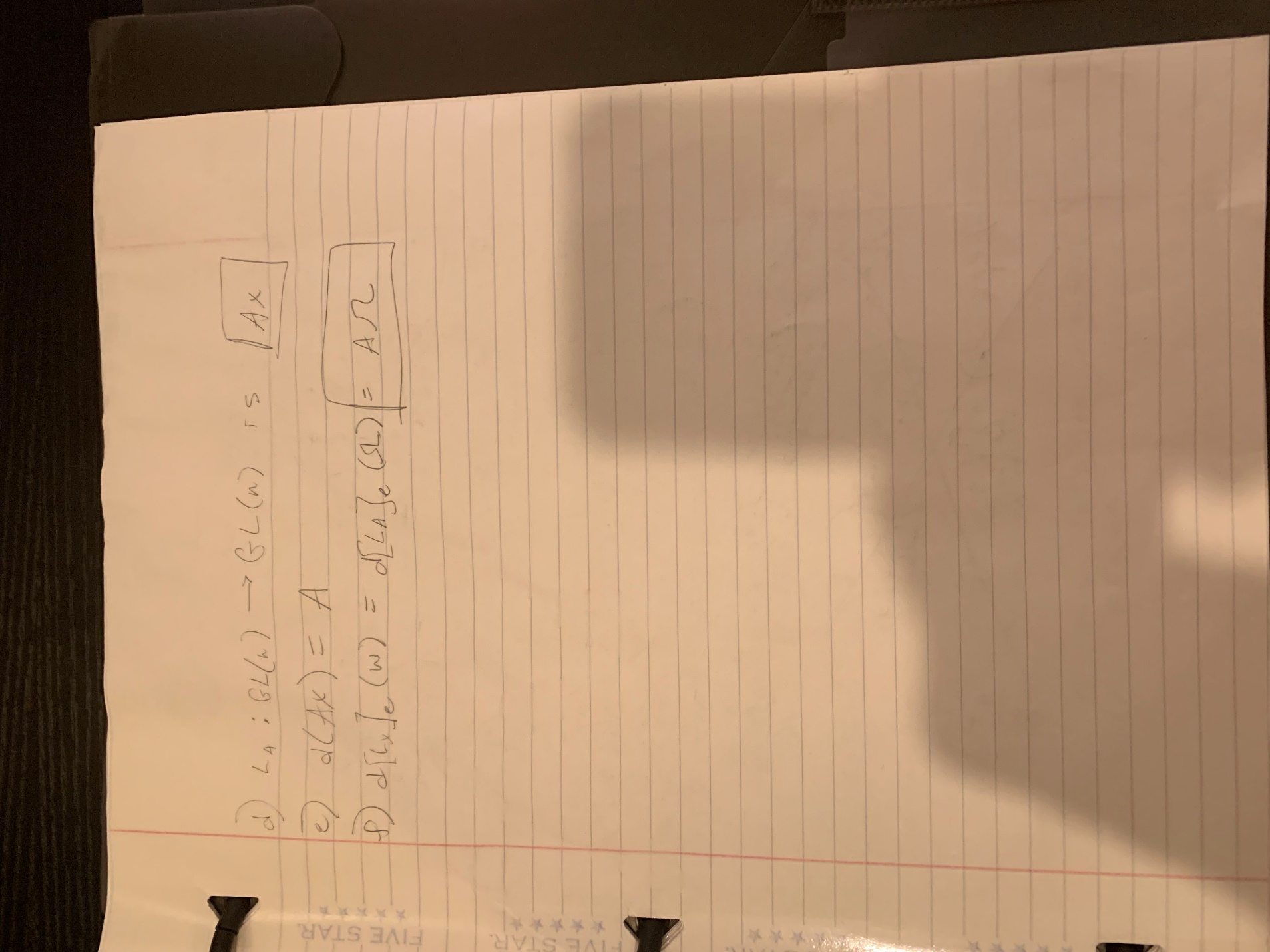
oP[3][0], oP[3][1], oP[3][2], oP[3][3] = 1, 1, 1, 1

oP\_inv = numpy.linalg.inv(oP)

T = numpy.dot(sP, oP\_inv)

print(T)

Diagram

Description automatically generatedA picture containing whiteboard

Description automatically generatedA picture containing timeline

Description automatically generatedDiagram

Description automatically generatedA picture containing diagram

Description automatically generated

**PART C:**

import numpy

from scipy.linalg import logm, expm

import matplotlib.pyplot as plt

from mpl\_toolkits import mplot3d

SE\_rows = 4

SE\_cols = 4

R\_rows = 3

R\_cols = 3

t\_rows = 3

t\_cols = 1

def get\_rotation\_matrix(X):

rot\_mat = [[0 for \_ in range(R\_cols)] for \_ in range(R\_rows)]

rot\_mat[0][0], rot\_mat[0][1], rot\_mat[0][2] = X[0][0], X[0][1], X[0][2]

rot\_mat[1][0], rot\_mat[1][1], rot\_mat[1][2] = X[1][0], X[1][1], X[1][2]

rot\_mat[2][0], rot\_mat[2][1], rot\_mat[2][2] = X[2][0], X[2][1], X[2][2]

return rot\_mat

def get\_position\_vector(X):

pos\_vec = [[0 for \_ in range(t\_cols)] for \_ in range(t\_rows)]

pos\_vec[0][0] = X[0][3]

pos\_vec[1][0] = X[1][3]

pos\_vec[2][0] = X[2][3]

return pos\_vec

def homogenize(t, R):

res\_mat = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

res\_mat[0][0], res\_mat[0][1], res\_mat[0][2], res\_mat[0][3] = R[0][0], R[0][1], R[0][2], t[0][0]

res\_mat[1][0], res\_mat[1][1], res\_mat[1][2], res\_mat[1][3] = R[1][0], R[1][1], R[1][2], t[1][0]

res\_mat[2][0], res\_mat[2][1], res\_mat[2][2], res\_mat[2][3] = R[2][0], R[2][1], R[2][2], t[2][0]

res\_mat[3][0], res\_mat[3][1], res\_mat[3][2], res\_mat[3][3] = 0, 0, 0, 1

return res\_mat

#return the homogeneous representation after the group multiplication

def group\_multiply(X, Y):

X\_R = get\_rotation\_matrix(X)

X\_t = get\_position\_vector(X)

Y\_R = get\_rotation\_matrix(Y)

Y\_t = get\_position\_vector(Y)

return homogenize(numpy.add(numpy.dot(X\_R, Y\_t), X\_t), numpy.dot(X\_R, Y\_R))

def gamma(t):

X\_0 = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

X\_0[0][0], X\_0[0][1], X\_0[0][2], X\_0[0][3] = 0.4330, 0.1768, 0.8839, 1

X\_0[1][0], X\_0[1][1], X\_0[1][2], X\_0[1][3] = 0.2500, 0.9186, -0.3062, 1

X\_0[2][0], X\_0[2][1], X\_0[2][2], X\_0[2][3] = -0.8660, 0.3536, 0.3536, 0

X\_0[3][0], X\_0[3][1], X\_0[3][2], X\_0[3][3] = 0, 0, 0, 1

X\_1 = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

X\_1[0][0], X\_1[0][1], X\_1[0][2], X\_1[0][3] = 0.7500, -0.0474, 0.6597, 2

X\_1[1][0], X\_1[1][1], X\_1[1][2], X\_1[1][3] = 0.4330, 0.7891, -0.4356, 4

X\_1[2][0], X\_1[2][1], X\_1[2][2], X\_1[2][3] = -0.5000, 0.6124, 0.6124, 3

X\_1[3][0], X\_1[3][1], X\_1[3][2], X\_1[3][3] = 0, 0, 0, 1

X\_0\_inv = numpy.linalg.inv(X\_0)

point = group\_multiply(X\_0, expm(t\*logm(group\_multiply(X\_0\_inv, X\_1))))

return point

#plot gamma from start to end inclusive

def plot\_gamma(start, end):

xline = []

yline = []

zline = []

for i in range(start, end+1):

xline.append(get\_position\_vector(gamma(i))[0][0])

yline.append(get\_position\_vector(gamma(i))[1][0])

zline.append(get\_position\_vector(gamma(i))[2][0])

fig = plt.figure()

ax = plt.axes(projection='3d')

print(xline,yline,zline)

ax.plot3D(xline, yline, zline)

plt.show()

**PART D:**

import numpy

from scipy.linalg import logm, expm

import matplotlib.pyplot as plt

from mpl\_toolkits import mplot3d

SE\_rows = 4

SE\_cols = 4

R\_rows = 3

R\_cols = 3

t\_rows = 3

t\_cols = 1

def get\_rotation\_matrix(X):

rot\_mat = [[0 for \_ in range(R\_cols)] for \_ in range(R\_rows)]

rot\_mat[0][0], rot\_mat[0][1], rot\_mat[0][2] = X[0][0], X[0][1], X[0][2]

rot\_mat[1][0], rot\_mat[1][1], rot\_mat[1][2] = X[1][0], X[1][1], X[1][2]

rot\_mat[2][0], rot\_mat[2][1], rot\_mat[2][2] = X[2][0], X[2][1], X[2][2]

return rot\_mat

def get\_position\_vector(X):

pos\_vec = [[0 for \_ in range(t\_cols)] for \_ in range(t\_rows)]

pos\_vec[0][0] = X[0][3]

pos\_vec[1][0] = X[1][3]

pos\_vec[2][0] = X[2][3]

return pos\_vec

def homogenize(t, R):

res\_mat = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

res\_mat[0][0], res\_mat[0][1], res\_mat[0][2], res\_mat[0][3] = R[0][0], R[0][1], R[0][2], t[0][0]

res\_mat[1][0], res\_mat[1][1], res\_mat[1][2], res\_mat[1][3] = R[1][0], R[1][1], R[1][2], t[1][0]

res\_mat[2][0], res\_mat[2][1], res\_mat[2][2], res\_mat[2][3] = R[2][0], R[2][1], R[2][2], t[2][0]

res\_mat[3][0], res\_mat[3][1], res\_mat[3][2], res\_mat[3][3] = 0, 0, 0, 1

return res\_mat

#return the homogeneous representation after the group multiplication

def group\_multiply(X, Y):

X\_R = get\_rotation\_matrix(X)

X\_t = get\_position\_vector(X)

Y\_R = get\_rotation\_matrix(Y)

Y\_t = get\_position\_vector(Y)

return homogenize(numpy.add(X\_t, Y\_t), numpy.dot(X\_R, Y\_R))

def gamma(t):

X\_0 = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

X\_0[0][0], X\_0[0][1], X\_0[0][2], X\_0[0][3] = 0.4330, 0.1768, 0.8839, 1

X\_0[1][0], X\_0[1][1], X\_0[1][2], X\_0[1][3] = 0.2500, 0.9186, -0.3062, 1

X\_0[2][0], X\_0[2][1], X\_0[2][2], X\_0[2][3] = -0.8660, 0.3536, 0.3536, 0

X\_0[3][0], X\_0[3][1], X\_0[3][2], X\_0[3][3] = 0, 0, 0, 1

X\_1 = [[0 for \_ in range(SE\_cols)] for \_ in range(SE\_rows)]

X\_1[0][0], X\_1[0][1], X\_1[0][2], X\_1[0][3] = 0.7500, -0.0474, 0.6597, 2

X\_1[1][0], X\_1[1][1], X\_1[1][2], X\_1[1][3] = 0.4330, 0.7891, -0.4356, 4

X\_1[2][0], X\_1[2][1], X\_1[2][2], X\_1[2][3] = -0.5000, 0.6124, 0.6124, 3

X\_1[3][0], X\_1[3][1], X\_1[3][2], X\_1[3][3] = 0, 0, 0, 1

X\_0\_inv = numpy.linalg.inv(X\_0)

point = group\_multiply(X\_0, expm(t\*logm(group\_multiply(X\_0\_inv, X\_1))))

return point

#plot gamma from start to end inclusive

def plot\_gamma(start, end):

xline = []

yline = []

zline = []

for i in range(start, end+1):

xline.append(get\_position\_vector(gamma(i))[0][0])

yline.append(get\_position\_vector(gamma(i))[1][0])

zline.append(get\_position\_vector(gamma(i))[2][0])

fig = plt.figure()

ax = plt.axes(projection='3d')

ax.plot3D(xline, yline, zline)

plt.show()

**PART E:**

Part c [0,1]**Chart

Description automatically generated**

Part c [0,30]**Chart, radar chart

Description automatically generated**

Part d [0,1]**Chart

Description automatically generated**

Part d [0,30]**Chart, radar chart

Description automatically generated**

