```
from sympy import *
import math
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

th, al, a, d, th0, th1, th2, th3, th4, th5 = symbols('theta alpha a d theta0 theta1
theta2 theta3 theta4 theta5', real=True)

# """To compute the DH parameters matrix"""
def Transformation_matrix(a, al, d, th):
      return Matrix([
      [sp.cos(th), -sp.sin(th) * sp.cos(al), sp.sin(th) * sp.sin(al), a *
      sp.cos(th)],
      [sp.sin(th), sp.cos(th) * sp.cos(al), -sp.cos(th) * sp.sin(al), a *
      sp.sin(th)],
      [0, sp.sin(al), sp.cos(al), d],
      [0, 0, 0, 1]
      ])


def j_matrix (th0,th1,th2):
    T1 = (Transformation_matrix(0, sp.pi/2, 1.2, rad(th0)))
    T2 = (Transformation_matrix(1, 0, 0, rad(th1)))
    T3 = (Transformation_matrix(1, 0, 0, rad(th2)))

    T06 = ((T1 * T2 * T3))
    T02 =(  T1 * T2)
    T03 = (T1 * T2 * T3)
    print("T03.........................................")
    pprint(N(T03))
    print("\n")
    O06 = np.array([[T06[0, 3]], [T06[1, 3]], [T06[2, 3]]])
    O01 = np.array([[T1[0, 3]], [T1[1, 3]], [T1[2, 3]]])
    O02 = np.array([[T02[0, 3]], [T02[1, 3]], [T02[2, 3]]])

    R0 = sp.Matrix([[1, 0, 0, 0],
                    [0, 1, 0, 0],
                    [0, 0, 1, 0],
                    [0, 0, 0, 1]])

    R01 = np.array(T1[:3, :3])
    R02 = np.array(T02[:3, :3])
    R00 = np.array(R0[:3,:3])

    d1 = O06 - O01
    d2 = O06 - O02

    first_col_1 = np.cross((R00[:,-1]).flatten(),O06.flatten())
    second_col_1 = np.cross((R01[:, -1]).flatten(), d1.flatten())
    third_col_1 = np.cross((R02[:,-1]).flatten(), d2.flatten())


    first_col_2 = R00[:,-1]
    second_col_2 = R01[:,-1]
    third_col_2 = R02[:,-1]

    j1 = np.concatenate((first_col_1, first_col_2), axis=None)
```

```
    j2 = np.concatenate((second_col_1, second_col_2), axis=None)
    j3 = np.concatenate((third_col_1, third_col_2), axis=None)

    J= np.column_stack((j1, j2, j3))
    J_matrix = (sp.Matrix(J))
    return J_matrix
print("final tranformation matrix and Jacobian matrix for home position")
j= j_matrix(0,0,0)

pprint(N(j))
print("\n")
print("final tranformation matrix and Jacobian matrix for 45 degree position")
j= j_matrix(0,45,-45)

pprint(N(j))
print("\n")




# j_matrix(rad(0),0,rad(0))da_dth1 = diff(a, th1)
```