שיטות נומריות למהנדסים (019003)

HW 3

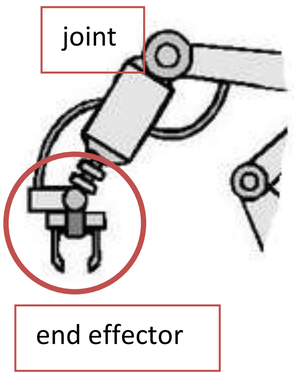
Roots of a System of Equations

1. Inverse kinematics of a serial robot

The inverse kinematics (IK) problem for a robot is as follows, for a given location of the end effector $[x, y, z]$, determine the corresponding joints angles of the of the robot $\theta_i, i = 1..3$.

Usually, there are multiple solutions to this problem i.e., multiple combinations of joint angles have the same end effector location.

In this question, the location of a robot's end effector is defined by the following equations:



$$X = c_1\left(l_2 + s_2 l_3 + s_{23} l_4 + c_{23} l_5\right)$$
$$Y = s_1\left(l_2 + s_2 l_3 + s_{23} l_4 + c_{23} l_5\right)$$
$$Z = l_1 + c_2 l_3 + c_{23} l_4 - s_{23} l_5$$

(given in the function *ForwardKinematics.m*).

The angles are denoted in the following way:

$$s_i = \sin\theta_i,\ c_i = \cos\theta_i,\ s_{ij} = \sin\left(\theta_i + \theta_j\right),\ c_{ij} = \cos\left(\theta_i + \theta_j\right)$$

The angles are defined in the interval $[0\ 2\pi]$

The lengths of the robot links are: $l_1 = 330,\ l_2 = 88,\ l_3 = 400,\ l_4 = 40,\ l_5 = 405[mm]$.

In the following questions you will solve the IK problem for the following end-effector locations P_k:

$$P_1 = (500\ 0\ 500), P_2 = (700\ 500\ 400), P_3 = (700\ 500\ 600)$$

Useful functions (not mandatory):

```
unique(A,'rows')
scatter3
```

1. Use MATLAB's function *fsolve* to find **all** the real solutions for all the given points (P_1, P_2, P_3) with precision $\left\|P_i - P_i^k\right\|_2 < 10^{-4}$.

   It is known that the **maximal** number of solutions for each point, P_i ,is 4. In case there is no solution, your code should print an error massage.

   1.1. Start with equally spaced grid of angles with 20X20X20 points (8000 in total). These points are the initial guesses for the solver. For each point P_i, find all the solutions. Assume this grid is sufficiently dense to catch all the existing solutions.

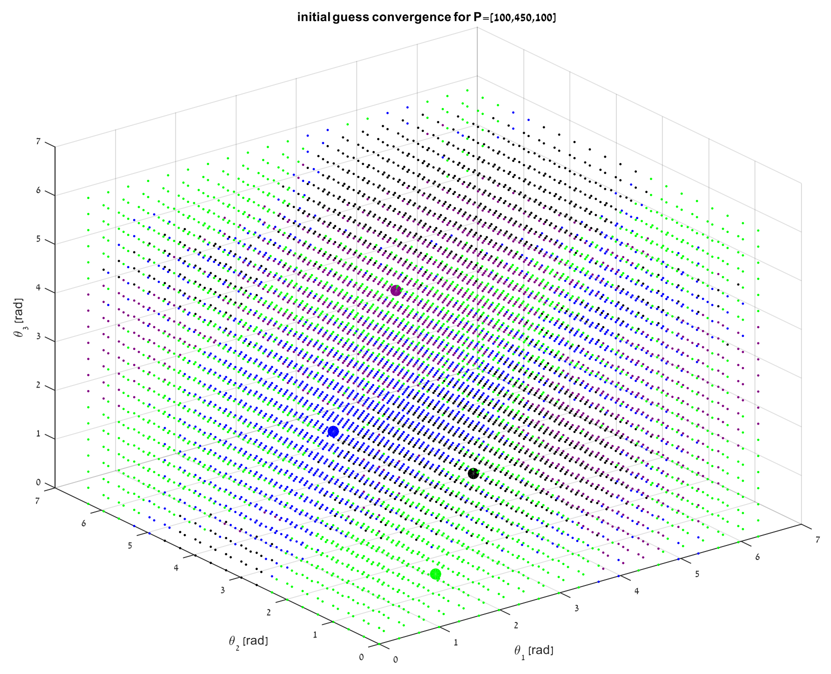
   1.2. Find the minimal number of initial guesses needed to obtain all the solutions by gradually lowering the grid density. Note that the resolution should be kept equal in all angles. What is the minimal number, n_{min}, of the initial guesses needed to find **all** the solutions for each end-effector location P_i?

   1.3. For each location P_i, create a random value matrix of initial guesses:

   $$randguess = 2pi \cdot rand(n_{min}, 3).$$

   Try finding the solutions. Were you able to find all of them? Discuss what are the pros and cons of using randomly distributed initial guesses.

   1.4. For each location P_i, plot, a point cloud of all the initial guesses. Each initial guess will be colored according to the arbitrary solution number that it converges to (1-Green, 2-Blue, 3-Black, 4-Red, no solution-yellow). The point cloud will include the final solution clearly marked with the same color (see example). Repeat this process for the maximal grid resolution, the minimal grid resolution you found at section 1.2, and the random guesses from section 1.3. (3x3=9 graphs in total). Discuss your results.



Q1.3 figure example

2. In the MATLAB function, *fsolve*, there is an option to supply the Jacobian in addition to the function value. Such an example can be found in recitation 4.
   2.1. Calculate the analytic Jacobian using Matlab's symbolic toolbox.
   2.2. Add the Jacobian to the Forwordkinematics output (you will also need to set the appropriate *fsolve* option). For one of the points P_i and one of the initial-guesses grid, compare the average number of iterations and overall runtime for the point cloud calculation with and without the Jacobian matrix. Summarize your findings and discuss.

(Hint `options = optimoptions('fsolve','SpecifyObjectiveGradient',true)`)