DNRS- Home task (Kinematics)

Task:

1) Select a robot's elbow model from one of the following: Assign yourself a model from here
    a. Scara manipulator (refer to Spong)
    b. Stanford Manipulator
    c. Spherical Manipulator
    d. Antropomorphic
    e. Puma
2) Couple your elbow model with a spherical wrist that satisfies Euler angles arrangement (xyx, xzx, yxy, ..etc.)
3) Assign coordinate frames on your model while maintaining local z-axis as the actuation axis for all joints.
4) Solve Forward kinematics problem (**Without** using DH-Parameter). Use 6 transformations between two frames.
5) Solve inverse kinematics problem using any approach you like.

Important Notes:

You need to write everything in code from scratch. Basing you solution on libraries or toolboxes (e.g. Peter corke) is not allowed. However, you are allowed to use robotics libraries and toolboxes to verify that your solution is correct.

References:

It is recommended to refer to these references for information about Euler Angles and the listed manipulator models.

1) B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, "Robotics: Modelling, Planning and Control", 3rd Edition, Springer, 2009
2) Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control, Second Edition, John Wiley & Sons, Inc. 2008

Submission:

You should submit a working code python and a report.

In your report:

- The model you chose.
- Wrist configuration you chose (explained by assigned coordinate frames).
- A simple graph of your coupled model.
- Assignment of Coordinate frames on the robot
- Forward kinematics solution (only symbolic, the code is responsible for calculations)
- Inverse kinematics solution

- Results of code implementation
- Validation of inverse kinematics solution (one of inverse kinematics solutions should correspond with forward kinematics input)
- Explanation of any extra functionalities implemented in code.

In your code:

- Function “FK_solve(q, flag)”:
  - Takes input list of q [q1, q2 ….q6] and flag (“ee” or “full”)
  - Output should be:
    - If flag == “ee”, the function should return 4x4 homogenous transformation matrix represents end effector frame.
    - If flag ==”full”, the function should return a list of homogenous transformation matrices. Each of these matrices represents transformation from base frame to frame i.
- Function “transform_base(trans,)”:
  - This function is responsible for placing the whole robot anywhere in space based on input “trans”
  - You should call FK_solve inside this function
- Function “IK_solve(base_frame, ee_frame)”:
  - Base_frame: 4x4 homogenous transformation matrix that represent robot base.
  - ee_frame : 4x4 homogenous transformation matrix that represent end effector pose.
  - Output is two lists:
    - List of all possible solutions
    - List of the solution that corresponds to your input into FK_solve function.
  - The function should perform inverse transformation from the input base towards zero frame. Which means input of this function should be the output of “transform_base()”.
- Name your file “{last name}_{first name}_HA1.py”

Bonus task (up to 20%):

Bonus option 1:

- Write a function to Plot full manipulator.
- Simple sliders represent actuator input to animate your graph.
- Function to animate model between two configurations.
- A function that extracts (x,y,z, Rx, Ry, Rz) from transformation matrix.

Bonus option 2:

- Write a Class “robot” which contains following:
  - Link lengths and other parameters
  - A default zero configuration of the robot
  - All transformed frames from base{0} through joint{i}
  - Function to solve FK
  - Function to solve IK
  - Function for plotting

- Function to extract position and orientation from 4x4 homogenous transformation.
- Function to perform FK for a series of configurations
- Function to solve IK for a series of poses.

Grading:

- Derivation of solution (50%)
- Organization of report (20%)
- Code returns correct output (30%)