FR- Home task 2

Task:

1) Derive inverse kinematics for your robot model
2) Solve inverse kinematics for multiple positions
3) Track the number of solutions along the way and choose the correct one and closest to the previous (current) configuration.
4) Derive the jacobian matrix for your robot model.
5) Plan a synchronized trajectory for all 6 joints between two poses. (consider 20Hz controller frequency)
6) Use the Jacobian matrix to check for singularities along the planned trajectory.

References:

It is recommended to refer to these references for information about the implementation of Jacobian and singularity analysis.

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 need to submit code and report in either of these forms:

1) Python file with code together with report in PDF format.
2) Colab notebook that contains both report and code.

In your report:

- A scheme of your robot model.
- Derivation of your solution.

Bonus: (you're a cool guy who has time to do it!)

- Drive the robot's end-effector from one pose to another using the inverse jacobian solution.
  You have the rule for mapping joint velocity to end-effector velocity:

$$\dot{x} = J(q)\,\dot{q}$$

Assume the robot is at position X_0, you need to drive the robot from X_0 to pose X_1 with inverse jacobian solution as follows:

$$J^{-1}(q)\,(X_1 - X_0) = \dot{q}$$

Report what problems are associated with inverse jacobian implementation.

- Plan the same path using polynomial trajectory.
  - You need to sample your path over (n) number of segments
  - Use 4-3-4 approach (refer to Siciliano)