RBE 501 – Robot Dynamics

Fall 2022 – Instructor: Andre Rosendo

Homework 4

INSTRUCTIONS

- **This homework assignment includes Gravity Compensation and Inverse Dynamics Control**
- **Total possible points on this homework: 100.**
- **Submit your solutions as a zip file through Canvas.**
  - The zip file should include any MATLAB code you have developed as well as a PDF detailing the steps you took to solve the problems and any results that you wish to show.
  - The PDF must be clearly legible. Plots and figures must be of professional quality. Figure DPI >= 300. Remember to add axis labels and units.
  - Useful tips on how to write a good report: https://advice.writing.utoronto.ca/types-of-writing/lab-report/
  - Check your code before submitting it! Code that runs with an error will not be graded.
  - To solve section 2 of this homework, you will have to have Peter Corke's Robotics toolbox installed on your system (ver 10.4 or newer). See https://petercorke.com/toolboxes/robotics-toolbox/.
- **Due date: Monday 5-Dec-22 at 3:00 pm**

Dynamics of Serial Robotics Arms (100 points total)

Problem 1: Inverse Dynamics Control of 7-DoF arm

The robot in Figure 1 is a three degrees-of-freedom manipulator, with link lengths $L_1 = L_2 = 0.3$ m, $L_3 = 0.15$ m, and link masses $m_1 = 5$ kg, $m_2 = m_3 = 1$ kg. A model of this manipulator was pre-created for your convenience: open the zip archive with the starting MATLAB code provided on Canvas, then run the `hw4problem1.m` script.

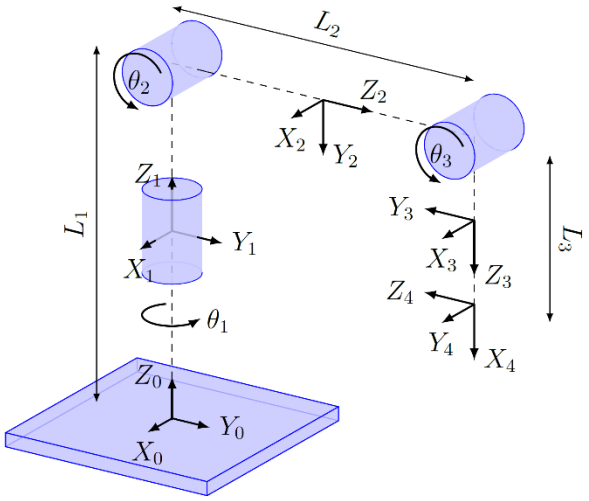

Figure 1: 3-DoF robotic arm in its home configuration.

***Note**: running this script will result in an error. The error is expected, and it occurs because the dynamics of this robot is not implemented yet – we will work on this next.*

Robot frames: The schematic in Figure 1 includes four Cartesian reference frames. Frames {0} and {4} are the space and end-effector (body) frames, respectively. Frames {1}, {2}, and {3} are the link frames. These frames are located at the center of mass of the corresponding links, and they are aligned with the link's principal axes of inertia (refer to section 8.2.1 of *Modern Robotics*). The coordinates

of these links, expressed in the space frame, are:

- Coordinates of {1} in {0}: (0, 0, $L_1/2$)
- Coordinates of {2} in {0}: (0, $L_2/2$, L_1)
- Coordinates of {3} in {0}: (0, L_2, L_1- $L_3/2$)

a. Calculate the homogeneous transformation matrices representing the pose of each frame with respect to the previous one (5 points)

Complete lines 48-51 of `hw4problem1m.m`. You should be able to calculate these matrices simply based on visual inspection of Figure 1.

b. Calculate the Spatial Inertia Matrices of each link (5 points)

Complete lines 62-64 of `hw4problem1m.m`. For now, assume that each link has no rotational inertia, i.e., $\mathcal{I}_b = \mathbf{0}_{3\times3}$.

c. Simulate the motion of the robot falling under its own weight (0 points)

Familiarize yourself with lines 71-89 of `hw4problem1m.m`. If you completed all the previous steps correctly, you should see an animation of the robot collapsing under its own weight: Video attached

d. Gravity Compensation (0 points)

Familiarize yourself with lines 95-123 of `hw4problem1m.m`. If you completed all the previous steps correctly, you should see the robot standing still in the home configuration: Video attached. This code generates the joint torques necessary to prevent the robot from falling under its own weight.

e. Inverse Dynamics (10 points)

Familiarize yourself with lines 127-247 of `hw4problem1m.m`. This latter part of the script performs the following operations:

- Lines 127-178: Generation of a path in task space, and conversion of this path to joint space via IK
- Lines 184-247: Generation of quintic polynomial trajectories between each pair of setpoints, and calculation of the torque profiles necessary to move the robot (inverse dynamics).

On line 202, the code invokes a function `quinticpoly` to generate trajectories. You have already developed this function in section 1 of this homework. If trajectory generation was implemented correctly, you should see the robot moving along the prescribed trajectory:

Video attached

f. Calculate the Spatial Inertia Matrices of each link (reprisal - 10 points)

Re-calculate the Spatial Inertia Matrices of each link (lines 62-64 of `hw4problem1m.m`). This time, assume that each link has the following dimensions (refer to Figure 2): $\ell = 4$ cm, $w = 4$ cm, and h is equal to the link length. Now repeat step e. Are the joint torque profiles different than what you observed before?

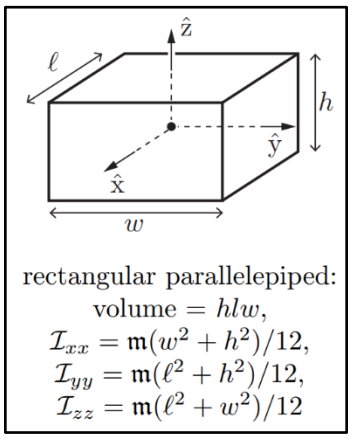

Figure 2: Principal Axes of Inertia of a rectangular bar, and corresponding entries of the rotational inertia matrix $\mathcal{I}_b$. Reproduced from Modern Robotics.

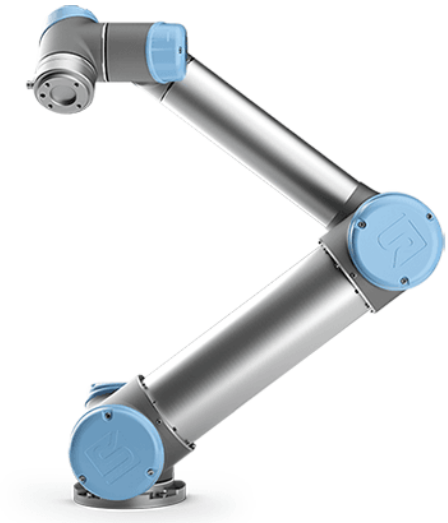

Figure 3: The Universal Robots UR-5.

g. Inverse Dynamics with a Payload (10 points)

Repeat step e, this time assuming that the robot is holding a payload of 1 kg. Are the joint torque profiles different than what you observed before?

Problem 2: Inverse Dynamics Control of the UR-5 Robot

Figure 2 shows the UR-5 (Universal Robots, Odense, Denmark), a compact 6-DoF robotic arm for collaborative applications. A MATLAB model of this robot was pre-created for your convenience – run `hw4problem2.m`.

***Note**: running this script will result in an error. The error is expected, and it occurs because the dynamics of this robot is not implemented yet – we will work on this next.*

a. Calculate the Spatial Inertia Matrices of each link (10 points)

Complete lines 39-44 of `hw4problem2.m`. The link masses and rotational inertia matrices are available on lines 23-36.

b. Calculate the joint torques necessary to make the robot move along a spiral trajectory (20 points)

Lines 81-86 generate a spiral in task space. Perform inverse kinematics on each of the points in the spiral, then define a quintic polynomial trajectory between each set of waypoints (you decide the robot speed!), and finally

perform inverse dynamics to calculate the joint torque profiles necessary to move the robot. Use forward dynamics to simulate the robot motion. ***Note***: *This step is essentially the same as step f in the previous problem, but this time you will have to do the bulk of the coding yourself.*