Robot Dynamics Midterm 2

Prof. Marco Hutter
Teaching Assistants: Jan Carius, Ruben Grandia, Joonho Lee, Takahiro Miki, Koen Krämer

November 4, 2020

Duration: 1h 15min
Permitted Aids: Everything; no communication among students during the test

1 Instructions

1. Download the ZIP file for midterm 2 from Piazza. Extract all contents of this file into a new folder and set MATLAB's[1] current path to this folder.
2. Run `init_workspace` in the Matlab command line
3. All problem files that you need to complete are located in the `problems` folder
4. Run `evaluate_problems` to check if your functions run. This script does not test for correctness. You will get 0 points if a function does not run (e.g., for syntax errors).
5. When the time is up, zip the entire folder and name it
   `ETHStudentID_StudentName.zip`
   Upload this zip-file through the following link
   `https://www.dropbox.com/request/jvKhbXNBOh3eVcBfvaWx`.
   You will receive a confirmation of receipt.
6. If the previous step did not succeed, you can email your file to
   `robotdynamics@leggedrobotics.com`
   from your ETH email address with the subject line
   `[RobotDynamics] ETHStudentID - StudentName`

[1] Online version of MATLAB at `https://matlab.mathworks.com/`

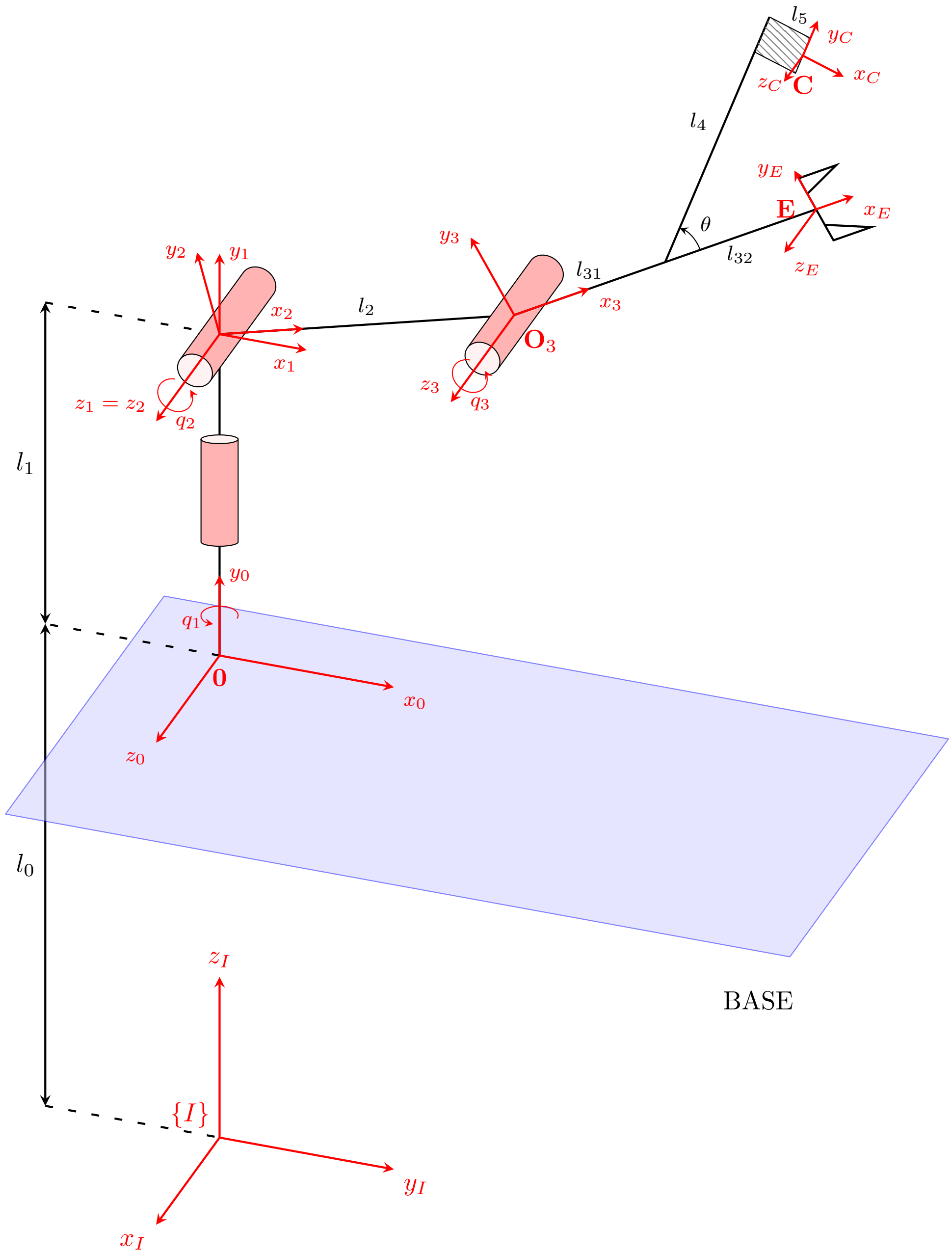

Figure 1: Schematic of a 3 degrees of freedom robotic arm attached to a fixed base. A camera is rigidly mounted on the last link of the arm.

2 Questions

In this midterm, you will model the dynamics of the robot arm shown in Fig. 1 and use it for control. It is a 3 degrees of freedom arm connected to a **fixed** base.

Let $\{0\}$ be the base frame, which is displaced by l_0 from the inertial frame $\{I\}$ along the $_I z$ axis. The arm is composed of three links. The reference frames attached to each link are denoted as $\{1\}, \{2\}, \{3\}$. The links' segments have lengths l_1, l_2, $l_{31} + l_{32}$. Additionally, a camera is mounted on the last link of the arm. As shown in figure 1, the camera link is mounted at a constant angle θ around the axis z_3. The mass and inertia of the camera is included in the link $\{3\}$.

The generalized coordinates are defined as

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^\top . \tag{1}$$

In the following questions, we have already provided the kinematics (transforms, Jacobians) and controller gains (k_P, k_D) for you. The variables stored as Matlab *cells* may be accessed as follows:

```
m{k};         % mass of link k
k_r_ks{k};    % Position of com of link k in frame k
k_I_s{k};     % Rotational inertia of link k in frame k
R_Ik{k};      % Rotation matrix from frame k to I
I_Jr{k};      % Rotational Jacobian of link k in I frame
etc..
```

Question 1. 4 P.

Calculate the mass matrix $\boldsymbol{M}(\boldsymbol{q})$, nonlinear terms $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ (Coriolis and centrifugal), and gravitational terms $\boldsymbol{g}(\boldsymbol{q})$. A helper function, `dAdt.m`, is available in the `utils` folder, to compute the time derivative of a matrix.

Implement your solution in `Q1_generate_eom.m`.

Question 2. 2 P.

In this question, you will adapt the dynamics to the case where a known object is grasped with the end-effector. Adapt the mass matrix $\boldsymbol{M}(\boldsymbol{q})$, nonlinear terms $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ (Coriolis and centrifugal), and gravitational terms $\boldsymbol{g}(\boldsymbol{q})$ when the robot arm is additionally holding an object of mass m_o and rotational inertia $_E I_o$ (expressed in frame E) with the end-effector. The object properties are available as parameters `params.m_o`, `params.E_I_o`.

As a starting point, use the mass matrix, non-linear terms and gravitational terms of the robot arm (without object) obtained from `M_fun_solution(q)`, `b_fun_solution(q, dq)` and `g_fun_solution(q)`.

You should implement your solution in `Q2_grasp_object.m`.

Question 3. 2 P.

Implement a forward dynamics simulator that computes the joint accelerations $\ddot{\boldsymbol{q}}$ and integrates them to get $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. You should implement the calculation of $\ddot{\boldsymbol{q}}$, given $\boldsymbol{\tau}$, the input torque for each joint.

Use the mass matrix, non-linear terms and gravitational terms obtained from `M_fun_solution(q)`, `b_fun_solution(q, dq)` and `g_fun_solution(q)`.

You should implement your solution in `Q3_forward_simulator.m`.

Question 4. 2 P.

Implement a joint-level PD controller that compensates for the gravitational terms and tracks desired joint positions and velocities. Calculate $\boldsymbol{\tau}$, the control torque for each joint.

Current joint positions $\boldsymbol{q}$ and joint velocities $\dot{\boldsymbol{q}}$, as well as desired joint positions $\boldsymbol{q}^d$ and desired joint velocities $\dot{\boldsymbol{q}}^d$ are given to the controller as input arguments to the Matlab file. You should obtain the gravitational terms in this question through the provided `g_fun_solution(q)`. Use the PD gains provided in the parameters (`params`).

Implement your solution in `Q4_gravity_compensation.m`. A simulation of your implemented controller (or the solution) is available in `simulate_robot_Q4.m`.

Question 5. 2 P.

Implement a controller that uses a task-space inverse dynamics algorithm, i.e. a controller which compensates the entire dynamics and tracks a desired motion in the task-space. Calculate $\boldsymbol{\tau}$, the control torque for each joint.

The inputs to this controller are the desired linear motion for the camera frame C as well as the current joint position $\boldsymbol{q}$ and joint velocities $\dot{\boldsymbol{q}}$. The desired motion has the following components (all expressed in the inertial frame):

- desired camera position ${}_I r_{IC}^d \in \mathbb{R}^3$
- desired camera velocity ${}_I v_C^d \in \mathbb{R}^3$
- desired camera acceleration ${}_I a_C^d \in \mathbb{R}^3$

Implement a controller that computes the torques necessary for following the desired linear acceleration of the end-effector in task-space as well as a feedback on the position and velocity of the end-effector. The desired motion is passed as input arguments to the Matlab file. The PD gains are provided in the parameters (`params`). Use the mass matrix, non-linear terms and gravitational terms obtained from `M_fun_solution(q)`, `b_fun_solution(q, dq)` and `g_fun_solution(q)`.

Implement your solution in `Q5_task_space_control.m`. A simulation of your implemented controller (or the solution) is available in `simulate_robot_Q5.m`.

Hint: The task specification only contains linear motion. When deriving the task space equations, use only those rows of the geometric Jacobian that correspond to this task.