Robot Dynamics Quiz 2

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

November 6, 2019

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

1 Instructions

1. Download the ZIP file for quiz 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 solutions must be written inside the provided MATLAB files in the problems folder. You should not create additional files.
4. Run `evaluate_problems` to check your solution.
5. Run `run_solutions` to see how the solutions should look like (Q.3,4,5).
6. When the time is up, zip the entire `problems` folder and name it `ETHStudentID_StudentName.zip`, i.e. `01-123-456_LastnameFirstname.zip`
   Upload this zip-file through the following link
   `https://www.dropbox.com/request/QpZZFQZhTAQpbJ8t1ZSi`.
   You will receive a confirmation of receipt.
7. 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`
8. Gains for PD controller are given in the script and they are used to evaluate your answers. **Do not change the numbers!**

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

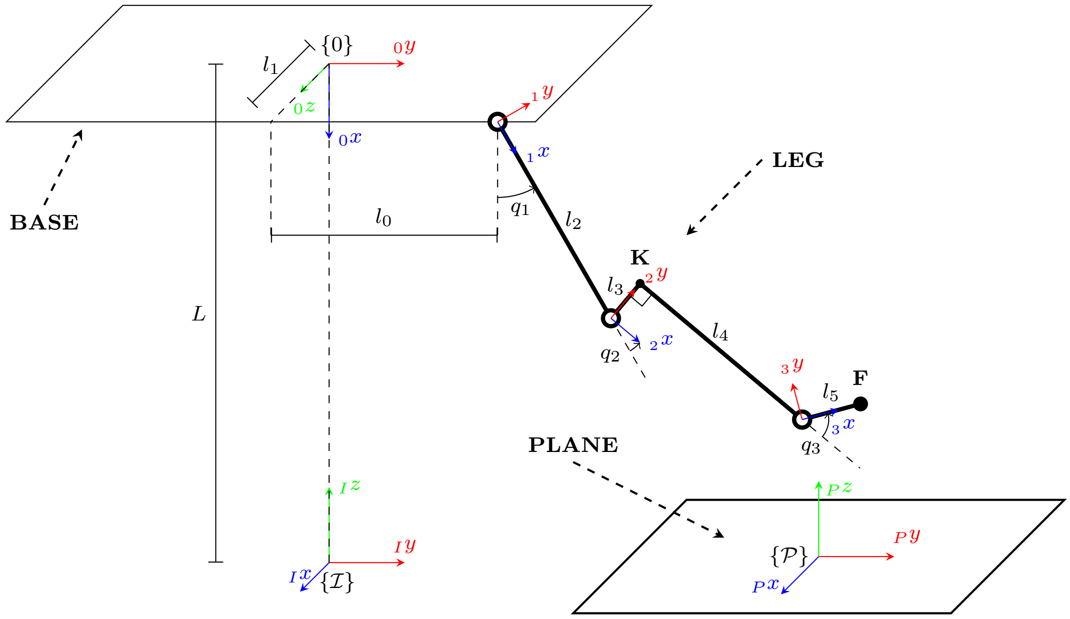

Figure 1: Three degree of freedom robotic leg attached to a fixed base. All joints rotate around their local positive z axis. The z axis of the frames $\{1\}$, $\{2\}$, $\{3\}$ is parallel to the ${}_0z$ axis.

2 Questions

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

Let $\{0\}$ be the base frame, which is displaced by L from the inertial frame $\{I\}$ along the ${}_Iz$ axis. The leg is composed of three links and is displaced by l_0 and l_1 from the base frame $\{0\}$ along the ${}_0y$, and ${}_0z$ axis, respectively. The links' segments have lengths l_2, l_3, l_4, l_5.
The generalized coordinates, with all angles in radians, are defined as

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

In the following questions, we have already provided the kinematics (transforms, Jacobians, Jacobian time derivatives) and controller gains (kP, kD) for you. You may access them as follows:

```
params.m{1}; % mass of link 1
params.k_r_ks{1}; % position of com of link 1 in same frmae
params.k_I_s{1}; % intertia tensor of link 1 in link 1 frame
I_Jr{1}; % rotational Jacobian of link 1 in I frame
I_dJr{1}; % time derivative of I_Jr{1}
I_Jp_s{1}(q); % positional Jacobian of com of link 1 in I frame
I_dJp_s{1}(q,dq); % time derivative of I_Jp_s{1}
I_Jp_F(q); % positional Jacobian of the foot in I frame
I_dJp_F(q,dq); % time derivative of I_Jp_F

params.kp_j; % P gain matrix for joints (3x3 diagonal matrix)
params.kd_j; % D gain matrix for joints (3x3 diagonal matrix)
params.kp_F; % P gain matrix for the end-effector (6x6 diagonal matrix)
params.kd_F; % D gain matrix for the end-effector (6x6 diagonal matrix)

```

```
% If generalized coordinate, gc (2X1 cell) are given,
gc.q;  % joint position.
gc.dq;  % joint velocity
```

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 $\boldsymbol{g}(\boldsymbol{q})$.
The solution should be implemented in `Q1_generate_eom.m`.

Question 2. 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 `Q2_forward_simulator.m`.

Question 3. 2 P.

Implement a joint-level PD controller that compensates for the gravitational terms and tracks a 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. You should obtain the gravitational terms in this question through the provided `g_fun_solution(q)`.
The PD gains are provided in the script.

You should implement your solution in `Q3_gravity_compensation.m`.

Question 4. 2 P.

Implement a controller that uses an 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 position and orientation of the end-effector (foot) as well as the current joint position $\boldsymbol{q}$ and joint velocities $\dot{\boldsymbol{q}}$:

- desired foot position ${}_I\boldsymbol{r}_{Fd} \in \mathbb{R}^3$
- desired foot orientation in euler angles ${}_I\boldsymbol{\chi}_{Fd} \in \mathbb{R}^3$
- desired foot linear velocity ${}_I\dot{\boldsymbol{r}}_{Fd} \in \mathbb{R}^3$
- desired foot angular velocity ${}_I\omega_{Fd} \in \mathbb{R}^3$

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)`. The PD gains are provided in the script.

You should implement your solution in `Q4_task_space_inverse_dynamics.m`.

Hint: To get a rotation matrix from euler angle, `eulAngXyzToRotMat(eul)` is used, and, to transform the rotation error matrix to rotation vector, `rotMatToRotVec(C_err)` is used. These are already implemented in the problem file. When calculating the inverse, use `pseudoInverseMat(A, damping factor)`.

Question 5. 3 P.

As shown in Fig. 1, there is a plane which is placed in some distance from the inertial frame. We want to perform a scratching motion on the place. The foot should scratch the plane in $-_Py$ direction while maintaining the desired perpendicular force in $_Pz$ direction. The foot should also keep the desired orientation. The desired perpendicular force is given and you have to define a tangent force to make the foot slip on the plane.
In this question, you should implement,

- end-effector force in inertial frame $_I\boldsymbol{F}_F \in \mathbb{R}^6$
- selection matrices $\boldsymbol{S}_M \in \mathbb{R}^{6\times 6}$, $\boldsymbol{S}_F \in \mathbb{R}^{6\times 6}$
- hybrid operational space controller that outputs control torque $\tau \in \mathbb{R}^3$

The inputs to the controller are

- desired foot position $_I\boldsymbol{r}_{Fd} \in \mathbb{R}^3$
- desired foot orientation in euler angles $_I\boldsymbol{\chi}_{Fd} \in \mathbb{R}^3$
- desired perpendicular force $_IF_{Fz} \in \mathbb{R}^1$

The friction coefficients are provided as follows.

- static friction coefficient between the foot and the plane μ_s
- kinetic friction coefficient between the foot and the plane μ_k

The PD gains are provided in the script.
You should implement your solution in `Q5_hybrid_operational_space_control.m`

Hint: To realize the slipping motion, the $-_Py$ direction force has to overcome friction. However, it should not be too large to avoid excessive acceleration that makes the simulation unstable. Check the desirable behavior from `run_solutions`.