Robot Dynamics Quiz 1

Prof. Marco Hutter, Victor Klemm
Teaching Assistants: Mike Zhang, Clemens Schwarke,
Filip Bjelonic, René Zurbrügg

October 16, 2024

Duration: 1h 15min
Permitted Aids: The exam is open book, which means you can use the script, slides, exercises, etc; The use of internet (besides for licenses) is forbidden; no communication among students during the test is allowed.

1 Instructions

1. Download the ZIP file `RobotDynamics_Quiz1_2024.zip` from Moodle. 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. Run `test_robot_viz` in the MATLAB command line to check that the included robot visualization runs.
4. All problem files that you need to complete are located in the `problems` folder.
5. 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).
6. When the time is up, zip the entire folder and name it
   `ETHStudentID_StudentName.zip`
   Submit this zip-file through Moodle under **Midterm Exam 1 Submission**. You should receive a confirmation email.
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. **Important:**
   (a) Implementations outside the provided templates will not be graded and receive 0 points.
   (b) The use of functions in MATLAB Toolboxes outside of the base MATLAB installation is not allowed.
   (c) Helper functions included in the `solutions/pcode` directory are specifically for Questions 4 and 5. Using these functions in the solutions for other questions is prohibited and will result in 0 points.

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

Figure 1: Start-up *Zürich Kinematics'* first product: a robot waiter.

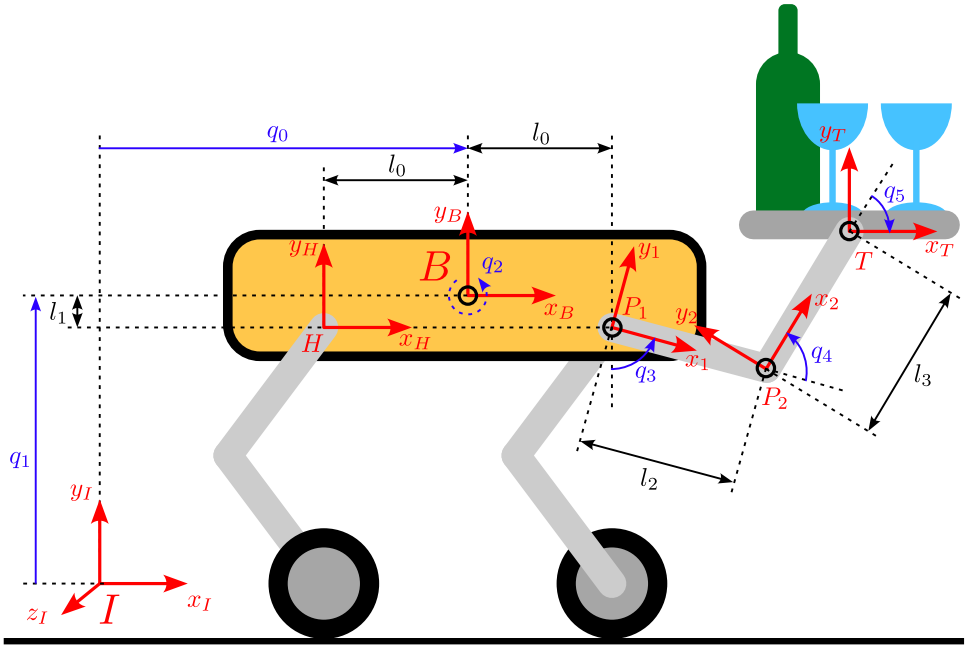

Figure 2: Simplified diagram of the robot waiter. The wheel on one of the front legs was swapped with a serving tray. Thus, the tray is actuated and can rotate around the z_T axis. All four rotational joints, indicated with hollow circles, rotate around the positive z axis (For clarity, the z axis is only drawn for the I frame).

2 Questions

In this quiz, you will model the forward and differential kinematics for the robot waiter shown in Fig. 2. The system has 6 Degrees of Freedom (DoF) in total, 3 for the base and 3 for the leg holding a tray.

The reference frame attached to the base is denoted as $\{B\}$, and is modeled as two prismatic joints q_0 and q_1, and one revolute joint q_2. The base moves with respect to the inertial frame $\{I\}$. The reference frames attached to each link of the robot's tray carrying leg are denoted as $\{P_1\}$ and $\{P_2\}$ while the reference frame attached to the tray is denoted as $\{T\}$.

The reference frame $\{H\}$ is centered at the hip joint of the hind leg and rigidly attached to the base of the robot (i.e. the rotation matrix between frames $\{H\}$ and $\{B\}$ is always $\mathbb{I}_{3\times3}$).

The generalized coordinates are defined as

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} \in \mathbb{R}^6 \, . \tag{1}$$

Clarification: The angles q_2, q_3, q_4, q_5 are measured using the right-hand thumb rule. For the state of the scene shown in Fig. 2, q_2 would be zero, q_3 and q_4 would be positive, and q_5 would be negative.

In the following questions, all required parameters are passed to your functions in a structure called `params`. You can access it as follows:

```
l0 = params.l0;
l1 = params.l1;
l2 = params.l2;
l3 = params.l3;
```

Question 1. 6 P.

Find the homogeneous transformation between the inertial frame $\{I\}$ and the tray frame $\{T\}$, i.e., the matrix $\mathbf{T}_{IT}$ as a function of the generalized coordinates $\mathbf{q}$.

Hint: You might need to reuse the rotation matrices and translation vectors for question 2.

You should implement your solution in the function `jointToTrayPose.m`

Question 2. 4 P.

Compute the position Jacobian ${}_I\mathbf{J}_{IT,P} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\mathbf{v}_{IT} = {}_I\mathbf{J}_{IT,P}(\mathbf{q})\dot{\mathbf{q}}, \tag{2}$$

where ${}_I\mathbf{v}_{IT} \in \mathbb{R}^3$ is the linear velocity of point T (the tray frame origin) with respect to a fixed point expressed in frame $\{I\}$.

You should implement your solution in the function `jointToPositionJacobian.m`

Question 3. 2 P.

Compute the rotation Jacobian ${}_I\mathbf{J}_{IT,R} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\boldsymbol{\omega}_{IT} = {}_I\mathbf{J}_{IT,R}(\mathbf{q})\dot{\mathbf{q}}, \tag{3}$$

where ${}_I\boldsymbol{\omega}_{IT} \in \mathbb{R}^3$ is the angular velocity of the tray frame $\{T\}$ with respect to the inertial frame $\{I\}$ expressed in frame $\{I\}$.

You should implement your solution in the function `jointToRotationJacobian.m`

Question 4. 4 P.

You recently launched a startup called *Zürich Kinematics*, specializing in selling your latest flagship robot to restaurants as an automated waiter. The initial deployment of these robots has been successful so far. However, some restaurant owners have recently seen impressive videos of robots performing dance routines on the video portal ClickClack.

They observed that your robots often stand idle while picking up drinks and suggested it would be a great opportunity to entertain customers by incorporating some engaging dance moves during these moments.

Hence, your task is now to implement a tracking controller that is able to track a desired base pose ${}_I\mathbf{p}^*_{IB,2D} = \left[{}_Ix^*_{IB}, {}_Iy^*_{IB}, {}_I\phi^*_{IB}\right]^T$ while trying to keep the tray stationary (i.e. ensure zero angular and linear velocity at the tray frame $\{T\}$).

The desired base pose as well as the current generalized coordinates $\mathbf{q}$, are passed as inputs to your MATLAB function.

For this question, we provide the following:

- the 2D base Jacobian ${}_I\mathbf{J}_{IB,2D} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\mathbf{w}_{IB,2D} = \begin{bmatrix} {}_I\dot{x}_{IB} \\ {}_I\dot{y}_{IB} \\ {}_I\dot{\phi}_{IB} \end{bmatrix} = {}_I\mathbf{J}_{IB,2D}\dot{\mathbf{q}}. \tag{4}$$

  You can call it with `jointToBaseJacobian_solution(q, params);`

- the 2D tray Jacobian ${}_I\mathbf{J}_{IT,2D} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\mathbf{w}_{IT,2D} = \begin{bmatrix} {}_I\dot{x}_{IT} \\ {}_I\dot{y}_{IT} \\ {}_I\dot{\phi}_{IT} \end{bmatrix} = {}_I\mathbf{J}_{IT,2D}\dot{\mathbf{q}}. \tag{5} $$

  You can call it with `jointToTrayJacobian_solution(q, params);`

- A function to compute the damped pseudo-inverse of a matrix `A`. You can call it with `pseudoInverseMat_solution(A, lambda)`.

You should implement your solution in the function `kinematicTrajectoryControl.m`.

Your implementation is judged based on how well the robot base tracks a predefined trajectory with minimal linear and angular velocities of the tray. *Do not* modify the tunable parameters of the controller, namely K_p and λ.

A kinematics-simulator is implemented for you in `entertain_control_loop.m`, where the variable `use_solution` is set. You can run this file to visualize the result. If `use_solution` is set to 1, executing this script will show you how the solution should look like. If `use_solution` is set to 0, at each time-step the simulator will use the velocities returned by your own implementation.

Question 5. 2 P.

Today, you find yourself in a tricky situation: Soland Riegwart, a distinguished guest who happens to strongly dislike dogs, has arrived. To make things worse the tray joint at frame $\{T\}$, is malfunctioning and you want to minimize how much it moves unless you enjoy the sound of mechanical failure (which you don't!).

But you, being a sharp ETH student, quickly develop a clever solution:

- You let the robot stand up on its hind legs and convincingly pretend to be a human.
- You minimize any unnecessary movement in the tray joint, hoping to avoid a mechanical fiasco until you can repair it.

Now, because you took Robot Dynamics, you decide to implement a multi-task controller. The tasks are then to stand the robot up, keeping the tray level (no one likes spilled coffee), and to minimize the velocity at the tray joint.

To stand the robot up, it suffices to track a reference pose of the hip joint frame $\{H\}$ denoted by ${}_I\mathbf{p}^*_{IH,2D} = \left[{}_Ix^*_{IH}, {}_Iy^*_{IH}, {}_I\phi^*_{IH}\right]^T$

Hint: Can you convert the reference pose of the hip joint frame $\{H\}$ to a reference pose of the base frame $\{B\}$?

The tray is level at the initial state. Therefore, to keep the tray level, it suffices to enforce that the angular velocity of the tray frame $\{T\}$ is zero.

The reference hip joint frame pose as well as the current generalized coordinates $\mathbf{q}$, are passed as inputs to your MATLAB function.

For this question, we provide the following:

- the 2D base Jacobian ${}_I\mathbf{J}_{IB,2D} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\mathbf{w}_{IB,2D} = \begin{bmatrix} {}_I\dot{x}_{IB} \\ {}_I\dot{y}_{IB} \\ {}_I\dot{\phi}_{IB} \end{bmatrix} = {}_I\mathbf{J}_{IB,2D}\dot{\mathbf{q}}. \tag{6} $$

  You can call it with `jointToBaseJacobian_solution(q, params);`

- the 2D tray Jacobian ${}_I\mathbf{J}_{IT,2D} \in \mathbb{R}^{3\times 6}$, that fulfills:

$$ {}_I\mathbf{w}_{IT,2D} = \begin{bmatrix} {}_I\dot{x}_{IT} \\ {}_I\dot{y}_{IT} \\ {}_I\dot{\phi}_{IT} \end{bmatrix} = {}_I\mathbf{J}_{IT,2D}\dot{\mathbf{q}}. \tag{7} $$

  You can call it with `jointToTrayJacobian_solution(q, params);`

- A function to compute the damped pseudo-inverse of a matrix `A`. You can call it with `pseudoInverseMat_solution(A, lambda)`.

You should implement your solution in the function `kinematicStandupControl.m`.

Your implementation is judged based on how well the robot stands up, that the tray stays level, and that movement of the tray joint is minimized. *Do not* modify the tunable parameters of the controller, namely K_p and λ.

A kinematics-simulator is implemented for you in `standup_control_loop.m`, where the variable `use_solution` is set. You can run this file to visualize the result. If `use_solution` is set to 1, executing this script will show you how the solution should look like. If `use_solution` is set to 0, at each time-step the simulator will use the velocities returned by your own implementation.