Robot Dynamics Quiz 1

Prof. Marco Hutter
Teaching Assistants: Victor Klemm, Mayank Mittal,
Nikita Rudin, Clemens Schwarke, Lorenz Wellhausen

October 19, 2022

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_2022.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. 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`
   Submit this zip-file through Moodle under **Midterm Exam 1 Submission**. You should receive a confirmation email.
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`
7. **Important:**
   (a) Implementations outside the provided templates will not be graded and receive 0 points.
   (b) Helper functions included in the `solutions/pcode` directory are specifically for Question 4. Using these functions in the solutions for other questions is prohibited and will receive 0 points.

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

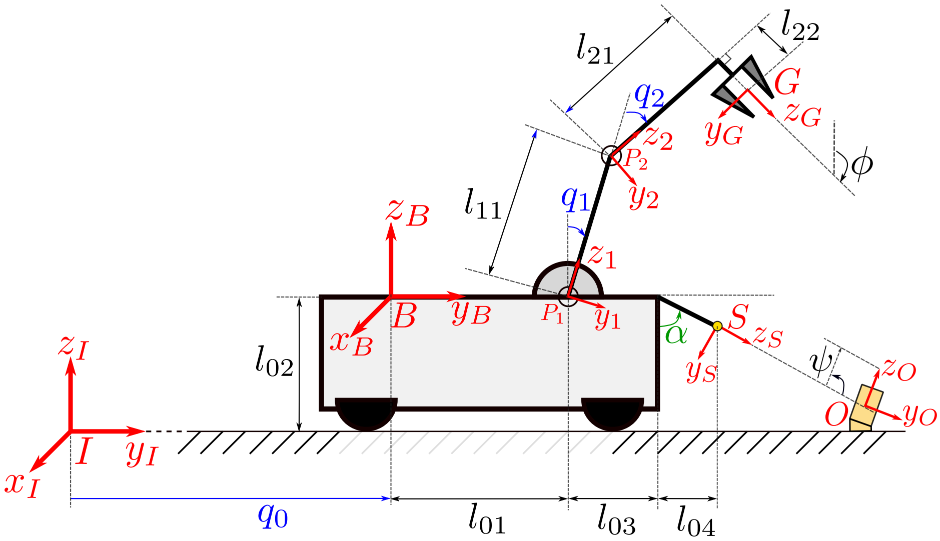

Figure 1: Schematic of a robot mobile manipulator with a two degrees of freedom robotic arm attached to a mobile base. The base can only move along y_I direction, while all joints on the arm can rotate around the positive x_I axis. The x axis of the frames $\{1\}, \{2\}$ is parallel to the x_I axis.

2 Questions

In this quiz, you will model the forward and differential kinematics for the robotic manipulator shown in Fig. 1. It is a 2 Degrees-of-Freedom (DoF) arm connected to a mobile base. The robot has a gripper attached to the last link of the arm, and a camera sensor attached to the base.

The base can only move linearly along the y_I axis. The reference frame attached to the base is denoted as $\{B\}$. The arm is composed of two links. The reference frames attached to each link are denoted as $\{P_1\}, \{P_2\}$. The frame attached to the gripper is denoted as $\{G\}$. *Note:* The transformation between $\{P_2\}$ and $\{G\}$ is fixed, i.e. there is a fixed 90 deg rotation between the gripper and the second link of the arm. As shown in Fig. 1, a camera sensor is rigidly mounted at point S on the base, rotated by a constant angle α. The corresponding reference frame is denoted as $\{S\}$.

Additionally, a target object is located at point O, with the corresponding frame denoted as $\{O\}$. The task is to grasp the object using the robot manipulator.

The generalized coordinates are defined as

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix} \in \mathbb{R}^3 \, . \tag{1}$$

Clarification 1: The angles q_1, q_2, ϕ, ψ are measured using right-hand thumb rule. For the state of the scene shown in Fig. 1, q_1, q_2 and ϕ would have negative values, while ψ would have a positive value.
Clarification 2: The angle α is measured with respect to negative z-axis in frame B. An angle of zero would mean that the sensor is facing towards the ground.

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

```
l01 = params.l01;
l02 = params.l02;
l03 = params.l03;
l04 = params.l04;
l11 = params.l11;
l21 = params.l21;
l22 = params.l22;
alpha = params.alpha;
```

Question 1. 6 P.

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

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

Solution 1.

```
function [ T_IG ] = jointToGripperPose( q, params )
  % Inputs:
  %     q: a 3x1 vector of generalized coordinates (3, 1).
  %     params: a struct of parameters.
  %
  % Outputs:
  %     T_IG: Homogenous transform from inertia to gripper frame ...
      (4, 4)

  % link lengths (meters)
  l01 = params.l01;
  l02 = params.l02;
  l03 = params.l03;
  l04 = params.l04;
  l11 = params.l11;
  l21 = params.l21;
  l22 = params.l22;
  % angle (radians)
  alpha = params.alpha;

  % Joint positions
  q0 = q(1);
  q1 = q(2);
  q2 = q(3);

  % Implement your solution here ...

  % inertia to base frame
  p_IB_I = [0; q0; l02];
  C_IB = eye(3);
  T_IB = [C_IB p_IB_I;
          zeros(1,3), 1];

  % base frame to link 1
  p_B1_B = [0; l01; 0];
  C_B1 = [1, 0, 0;
          0, cos(q1), -sin(q1);
          0, sin(q1), cos(q1)];
  T_B1 = [C_B1 p_B1_B;
          zeros(1,3), 1];

  % link 1 to link 2
  p_12_1 = [0;0; l11];
```

```
  C_12 = [1, 0, 0;
          0, cos(q2), -sin(q2);
          0, sin(q2), cos(q2)];
  T_12 = [C_12 p_12_1;
          zeros(1,3), 1];

  % link 2 to gripper
  p_2G_2 = [0; l22; l21];
  C_2G = [1, 0, 0;
          0, 0, 1;
          0, -1, 0];
  T_2G = [C_2G, p_2G_2;
          zeros(1,3), 1];

  % final: inertia to gripper
  T_IG = T_IB * T_B1 * T_12 * T_2G;
end
```

Question 2. 4 P.

Compute the position Jacobian ${}_I\mathbf{J}_P \in \mathbb{R}^{3\times3}$, that fulfills:

$$ {}_I\boldsymbol{v}_{IG} = {}_I\mathbf{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2} $$

where ${}_I\boldsymbol{v}_{IG} \in \mathbb{R}^3$ is the linear velocity of point G (the gripper) with respect to a fixed point expressed in frame $\{I\}$.
You should implement your solution in the function `jointToPositionJacobian.m`

Solution 2.

```
function [ J_IG_p ] = jointToPositionJacobian(q, params)
  % Inputs:
  %     q: a 3x1 vector of generalized coordinates
  %     params: a struct of parameters
  %
  % Outputs:
  %     J_IG_p: position Jacobian of gripper in inertia frame (3,3)

  % link lengths (meters)
  l01 = params.l01;
  l02 = params.l02;
  l03 = params.l03;
  l04 = params.l04;
  l11 = params.l11;
  l21 = params.l21;
  l22 = params.l22;
  % angle (radians)
  alpha = params.alpha;

  % Joint positions
  q0 = q(1);
  q1 = q(2);
  q2 = q(3);

  % Implement your solution here...

  % Note: Formula in the script, eq. 2.155, is only applicable
  %     for rotating joints, which are on the arm. For the base that
  %     only translates, we can account for its component directly.
  %
  % We apply eq. 2.154: J_IG_I = J_IB_I + J_BG_I
  %
  % Since, B is not a rotating frame, J_BG_B = J_BG_I

  % joint vectors (arm)
```

```
  n1_B = [1; 0; 0];
  n2_B = [1; 0; 0];

  % compute position vector from P1 to P2 in frame B
  p_12_1 = [0; 0; l11];
  C_B1 = [1, 0, 0;
          0, cos(q1), -sin(q1);
          0, sin(q1), cos(q1)];
  p_12_B = C_B1 * p_12_1;

  % compute position vector from P2 to G in frame B
  p_2G_2 = [0; l22; l21];
  C_12 = [1, 0, 0;
          0, cos(q2), -sin(q2);
          0, sin(q2), cos(q2)];
  C_B2 = C_B1 * C_12;
  p_2G_B = C_B2 * p_2G_2;

  % compute position vector from P1 to G in frame B
  p_1G_B = p_12_B + p_2G_B;

  % linear Jacobian (from base to gripper)
  J_BG_p = [zeros(3, 1), cross(n1_B, p_1G_B), cross(n2_B, p_2G_B)];
  % linear Jacobian (from inertia to base)
  J_IB_p = [[0; 1; 0], zeros(3, 1), zeros(3, 1)];

  % linear Jacobian (from inertia to gripper)
  J_IG_p = J_IB_p + J_BG_p;

end
```

Question 3. 2 P.

Compute the rotation Jacobian ${}_I\mathbf{J}_R \in \mathbb{R}^{3\times3}$, that fulfills:

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

where ${}_I\boldsymbol{\omega}_{IG} \in \mathbb{R}^3$ is the angular velocity of frame $\{G\}$ with respect to the inertial frame $\{I\}$, expressed in frame $\{I\}$.
You should implement your solution in the function `jointToRotationJacobian.m`

Solution 3.

```
function [ J_IG_r ] = jointToRotationJacobian(q, params)
  % Inputs:
  %     q: a 3x1 vector of generalized coordinates
  %     params: a struct of parameters
  %
  % Outputs:
  %     J_IG_r: rotation Jacobian of gripper in inertia frame (3,3)

  % link lengths (meters)
  l01 = params.l01;
  l02 = params.l02;
  l03 = params.l03;
  l04 = params.l04;
  l11 = params.l11;
  l21 = params.l21;
  l22 = params.l22;
  % angle (radians)
  alpha = params.alpha;

  % Joint positions
  q0 = q(1);
  q1 = q(2);
```

```
    q2 = q(3);

    % Implement your solution here...

    % joint vectors
    n1_I = [1; 0; 0];
    n2_I = [1; 0; 0];

    % rotation Jacobian
    J_IG_r = [zeros(3,1), n1_I, n2_I];

end
```

Question 4. 3 P.

Given a desired gripper pose ${}_I\boldsymbol{p}^*$, use inverse kinematics formulation to compute the generalized coordinates required to have the gripper frame $\{G\}$ coinciding and aligned with the desired pose.

We indicate with ${}_I\boldsymbol{p}^* \in \mathbb{R}^3$ the following vector:

$$
{}_I\boldsymbol{p}^* = \begin{bmatrix} {}_I y_G^* \\ {}_I z_G^* \\ \phi^* \end{bmatrix}, \tag{4}
$$

where the angle ϕ is indicated in Fig. 1.

For this question, we provide:

- a function to calculate the current gripper position in the plane ${}_I yz$:

$$
{}_I\boldsymbol{p}_{yz} = \begin{bmatrix} {}_I y_G \\ {}_I z_G \end{bmatrix} \in \mathbb{R}^2. \tag{5}
$$

  You can call it with `jointTo2DGripperPosition_solution(q, params)`;

- the analytical Jacobian $\mathbf{J}_A \in \mathbb{R}^{3\times3}$, that fulfills:

$$
{}_I\boldsymbol{w} = \mathbf{J}_A \dot{\boldsymbol{q}}. \tag{6}
$$

  You can call it with `jointToGripperAnalyticalJacobian_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 `inverseKinematics.m`.

Solution 4.

```
function [ q_iter ] = inverseKinematics( p_des, params )
    % Inputs:
    %  p_des          : desired gripper pose (3x1)
    %  params         : a struct of parameters

    % Output:
    %  q_iter         : joint position command (3x1)

    % Choose a pseudo_inverse damping coefficient
    lambda = 1e-2;
    % Choose a convergence threshold
    epsilon = 1e-3;
    % Maximum number of iterations
    N_max = 100;
```

```
    % initialize the IK
    q_iter = [1; 1; 1];

    % Implement your solution here...

    N = 0;
    % current gripper pose
    p_curr = [jointTo2DGripperPosition_solution(q_iter, params); ...
        sum(q_iter(2:3)) - pi / 2];
    % computer error
    p_error = norm(p_des - p_curr);

    % iteratively find solution
    while p_error > epsilon
        % incremenet counter
        N = N + 1;
        if N ≥ N_max
            error('Maximum IK iterations reached...');
        end
        % analytic Jacobian
        Ja = jointToGripperAnalyticalJacobian_solution(q_iter, params);
        % pseudo-inverse of the analytic Jacobian
        Ja_pinv = pseudoInverseMat_solution(Ja, lambda);
        % incremental update of joint angles
        q_iter = q_iter + Ja_pinv * (p_des - p_curr);
        % current gripper pose
        p_curr = [jointTo2DGripperPosition_solution(q_iter, ...
            params); sum(q_iter(2:3)) - pi / 2];
        % computer error
        p_error = norm(p_des - p_curr);
    end
end
```

Question 5. 3 P.

The sensor S on the mobile manipulator now detects the pose of a target object O. The sensor provides the position of this point O in sensor frame and the angle, ψ, between the z-axis of the sensor frame and the object frame.

We indicate this sensor measurement with ${}_{S}\boldsymbol{p}_{SO} \in \mathbb{R}^3$ the following vector:

$$
{}_{S}\boldsymbol{p}_{SO} = \begin{bmatrix} {}_{S}y_O \\ {}_{S}z_O \\ \psi \end{bmatrix}, \tag{7}
$$

where the angle ψ is indicated in Fig. 1.

Given as input the pose of the object in the sensor frame, compute the desired gripper pose (Eq. (4)) to execute grasping of the object.

Hint: For proper grasping, the z-axis of the gripper frame needs to be aligned opposite to the z-axis of the object frame. What will be the relative rotation between $\{O\}$ and $\{G_{des}\}$ in that case?

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

Solution 5.

```
function [p_des] = gripperToObject( q, pO, params )
    % Inputs:
    %  q             : current joint angles (3x1)
    %  pO            : detected object pose in sensor frame (3x1)
```

```
    %  params       : a struct of parameters

    % Output:
    %  p_des        : desired gripper pose in inertia frame (3x1)

    % link lengths (meters)
    l01 = params.l01;
    l02 = params.l02;
    l03 = params.l03;
    l04 = params.l04;
    l11 = params.l11;
    l21 = params.l21;
    l22 = params.l22;
    % angle (radians)
    alpha = params.alpha;

    % Joint positions
    q0 = q(1);
    q1 = q(2);
    q2 = q(3);

    % Object pose (in sensor frame)
    pO_y = pO(1);
    pO_z = pO(2);
    pO_psi = pO(3);

    % Implement your solution here ...

    % inertia to base frame
    p_IB_I = [0; q0; l02];
    C_IB = eye(3);
    T_IB = [C_IB p_IB_I;
            zeros(1,3), 1];

    % base frame to sensor frame
    p_BS_B = [0; l01 + l03 + l04; - l04 * cot(alpha)];
    C_BS = [1, 0, 0;
            0, cos(- pi + alpha), -sin(- pi + alpha);
            0, sin(- pi + alpha), cos(- pi + alpha)];
    T_BS = [C_BS p_BS_B;
            zeros(1,3), 1];

    % sensor frame to object frame
    p_SO_S = [0; pO_y; pO_z];
    C_SO = [1, 0, 0;
            0, cos(pO_psi), -sin(pO_psi);
            0, sin(pO_psi), cos(pO_psi)];
    T_SO = [C_SO p_SO_S;
            zeros(1,3), 1];

    % object frame to grasp frame
    p_OG_O = [0; 0; 0];
    C_OG = [1, 0, 0;
            0, -1, 0;
            0, 0, -1];
    T_OG = [C_OG p_OG_O;
            zeros(1,3), 1];

    % inertia frame to grasp frame
    T_IG = T_IB * T_BS * T_SO * T_OG;

    % convert into minimal represetation
    p_des = zeros(3, 1);
    p_des(1:2) = T_IG(2:3, 4);
    p_des(3) = atan2(T_IG(3, 2), T_IG(2, 2));

end
```