This assignment is concerned with differential kinematics of serial link robotic arms.

The following exercises are solved with MATLAB and utilize the Robotics System Toolbox from Mathworks. Some tasks involve the use of ROS on a Linux (Ubuntu) system. Note, Matlab Linux uses different keyboard short-cuts per default (copy, paste, etc.). You might change it at *MATLAB - Preferences - Keyboard - Shortcuts - Active Settings: Windows Default Set*.

Prior to the assignment, study the labs description and complete the quiz on Differential Kinematics in the Moodle workspace of the course. In case you are unable to answer the questions in the quiz correctly go back to the lecture slides and carefully study the chapters 3.1 and 3.2 on differential kinematics in the book by Siciliano et al [1]. Further recommended reading is the chapter on kinematics in the Springer Handbook of Robotics [2] in particular section 1.8 on forward instantaneous kinematics.

Questions

1) What is the objective of differential kinematics?

2) What is the relation between joint and task space velocities (formula)?

3) The task space velocity $\mathbf{v}$ is a 6×1 vector. What do the first and last three components describe?

4) Which of the following statements about the differential kinematics are true.
   a) The linear velocity part of geometric and analytical Jacobian are identical.
   b) A prismatic joint does not contribute to the linear end effector velocity.
   c) A revolute joint does not contribute to the linear end effector velocity.
   d) The Jacobian looses full rank at a kinematic singularity.

5) What is the difference between geometric and analytical Jacobin w.r.t. to the rotational velocity part?

6) What is the contribution of a revolute joint to the angular velocity part $\mathbf{J}_{Oi}$ of the geometric Jacobian $\mathbf{J}$ (formula)?

7) What characterizes the null space of a Jacobian? Give an example.

8) What is the relationship between the matrix $\mathbf{S}(\boldsymbol{\omega})$ and the angular velocity $\boldsymbol{\omega}$?

9) What happens if the robot manipulator is at a kinematic singularity?

10) In case of a redundant manipulator there are an infinite number of joint space velocities $\dot{\mathbf{q}}$ that generate a desired end effector velocity $\mathbf{v}_e$. How do you choose $\dot{\mathbf{q}}$ in that case?

11) Which method is used to solve (equality) constrained optimization problems?

Differential Kinematics

Consider the forward kinematics of a robot arm described by the homogeneous transformation

$$\mathbf{T}_e(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_e(\mathbf{q}) & \mathbf{p}_e(\mathbf{q}) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{q} = [q_1, \ldots, q_n]^T$ is the vector of joint variables. Differential kinematics is concerned with the problem to determine the translational ($\dot{\mathbf{p}}_e$) and angular ($\boldsymbol{\omega}_e$) velocity of the end effector given the joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$. The goal is to express the end effector linear velocity $\dot{\mathbf{p}}_e$ and angular velocity $\boldsymbol{\omega}_e$ as a function of the joint velocities $\dot{\mathbf{q}}$.

$$\dot{\mathbf{p}}_e = \mathbf{J}_p(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$
$$\boldsymbol{\omega}_e = \mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}} \tag{3}$$

$\mathbf{J}_p(\mathbf{q})$ is the $3 \times n$ matrix that describes the effect of the joint velocity $\dot{\mathbf{q}}$ on the end effector linear velocity $\dot{\mathbf{p}}_e$. $\mathbf{J}_o(\mathbf{q})$ is the $3 \times n$ matrix that describes the effect of the joint velocity $\dot{\mathbf{q}}$ on end effector angular velocity $\boldsymbol{\omega}_e$.

In a compact representation the linear and angular velocity are combined into the 6×1-vector of total velocity $\boldsymbol{\nu}_e$. The relationship is provided by the $6 \times n$-Jacobian matrix $\mathbf{J}(\mathbf{q})$ which depends on the joint configuration $(q_1, \ldots, q_n)$.

$$\boldsymbol{\nu}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{4}$$

The matrix $\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix}$ denotes the manipulators **geometric** Jacobian, which depends on the joint configuration $\mathbf{q}$.

For prismatic joints the geometric Jacobian is defined by

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix} \tag{5}$$

For revolute joints the geometric Jacobian is defined by

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \\ \mathbf{z}_i \end{bmatrix} \tag{6}$$

in which $\mathbf{z}_i$ denotes the i-th joint axis, $\mathbf{p}_e$ the translation vector of the end effector frame and $\mathbf{p}_i$ the vector to the origin of the i-th joint frame. Notice, that $\mathbf{z}_i$, $\mathbf{p}_e$, $\mathbf{p}_i$ are extracted from the homogeneous transforms of joint frames $\mathbf{H}_i$ and end effector frame $\mathbf{H}_e$. In particular $\mathbf{z}_i = \mathbf{H}_i(1:3, 3)$ is given by the third column of the rotation matrix part of $\mathbf{H}_i$, $\mathbf{p}_i = \mathbf{H}_i(1:3, 4)$ is given by the first three elements of the fourth column of $\mathbf{H}_i$, $\mathbf{p}_e = \mathbf{H}_e(1:3, 4)$ is given by the first three elements of the fourth column of $\mathbf{H}_e$.

Be aware that the Matlab Robotics System Toolbox reverts the order of the translational and rotational part of the geometric Jacobian and follows the convention

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_o(\mathbf{q}) \\ \mathbf{J}_p(\mathbf{q}) \end{bmatrix} \tag{7}$$

such that

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{8}$$

Jacobian Calculation with Symbolic Toolbox

`jacobian(f,v)` computes the Jacobian matrix of the symbolic function `f` with respect to symbolic variables `v`. The (i,j) element of the result is $\frac{\partial f_i}{\partial v_j}$. The Jacobian of a vector function is a matrix of the partial derivatives of that function. This code compute the Jacobian matrix of the vector function $[xyz, y^2, x+z]$ with respect to the variables $[x, y, z]$.

```
syms x y z
jacobian([x*y*z, y^2, x + z], [x, y, z])
ans =
[ y*z, x*z, x*y]
[   0, 2*y,   0]
[   1,   0,   1]
```

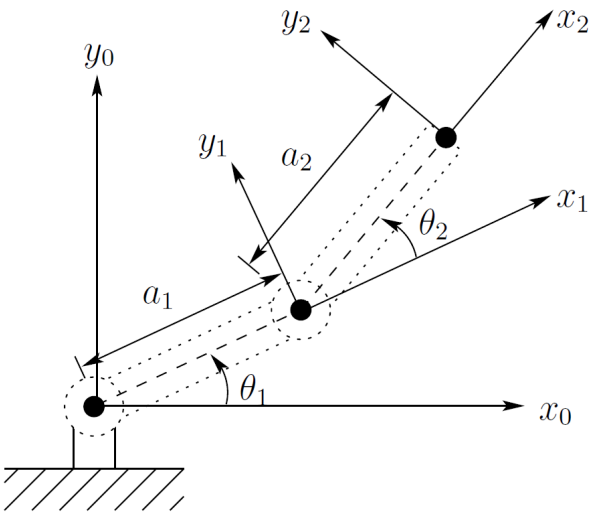

Figure 1: Two link planar arm

Fig. 1 shows a two link planar arm. The forward kinematics of the end effector $\{x_2, y_2, z_2\}$

w.r.t. robot base frame $\{x_0, y_0, z_0\}$ is given by $\mathbf{T}_2^0(\mathbf{q})$:

$$\mathbf{T}_2^0(\mathbf{q}) = \mathbf{A}_1^0(\mathbf{q})\mathbf{A}_2^1(\mathbf{q}) = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with

$$\mathbf{A}_i^{i-1}(\mathbf{q}) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i & 0 & a_i\sin\theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad i = 1, 2$$

Now, please complete tasks 1 to 3 in the template.

Geometric Jacobian Robotics System Toolbox

The Robotics System Toolbox provides functionalities for differential kinematics, in particular the calculation of the geometric Jacobian of arbitrary frames w.r.t. robot base frame. The command

```
jacobian = geometricJacobian(robot,configuration,endeffectorname)
```

computes the geometric Jacobian relative to the base for the specified end effector name and configuration for the robot model.

The resulting 6-by-n matrix is the geometric Jacobian of the end effector with the given configuration, where n is the number of degrees of freedom for the end effector. The Jacobian maps the joint-space velocity to the end effector velocity, relative to the base coordinate frame. The end effector velocity equals:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{9}$$

in which **omega** denotes the angular velocity and $\dot{\mathbf{p}}$ the translational velocity of the end effector frame (or any other frame specified by the string `endeffectorname`). $\mathbf{q} = [q_1, \ldots, q_n]$ denotes the joint-space velocity.

A kinematic singularity is a configuration within the robot's workspace where the Jacobian loses rank. Intuitively, this means that at a kinematic singularity the robot loses its ability to move the end effector in some direction no matter how the joints are actuated. Kinematic singularities often emerge when the robot's joints or links line up.

Now, please complete tasks 4 to 7 in the template.

The Jacobian matrix relates differential joint coordinate motion to differential Cartesian motion $d\mathbf{x} = \mathbf{J}(\mathbf{q})\, d\mathbf{q}$. The following tasks are based on the 6-DOF UR10 robot arm with a 6×6-Jacobian.

The geometric Jacobian is usually defined for translational and angular velocities expressed w.r.t. the base frame. In applications such as visual servoing one would like to express translational and angular velocities expressed w.r.t. a camera or end effector frame. If it is desired to represent the Jacobian in a different frame u, it is sufficient to know the relative rotation matrix $\mathbf{R}^u$. The relationship between velocities in the base frame and frame u is

$$\begin{bmatrix} \dot{\mathbf{p}}_e^u \\ \dot{\omega}_e^u \end{bmatrix} = \begin{bmatrix} \mathbf{R}^u & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^u \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}}^e \\ \dot{\omega}^e \end{bmatrix} \tag{10}$$

which gives

$$\mathbf{J}^u = \begin{bmatrix} \mathbf{R}^u & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^u \end{bmatrix} \mathbf{J} \tag{11}$$

Now, please complete tasks 8 to 10 in the template.

Inverse Kinematics Algorithms

The Jacobian inverse technique is a simple yet effective way of implementing inverse kinematics. A classic control technique to impose a desired end effector motion in task space is the resolved rate motion control

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}\dot{\mathbf{x}}, \tag{12}$$

which rests upon the inversion of the Jacobian matrix in the world frame.

The differential equation (12) can be converted into an equivalent difference equation

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{J}(\mathbf{q_k})^{-1}\dot{\mathbf{x}_k}\Delta t, \tag{13}$$

for numerical integration with Euler or Runge-Kutta methods. $\mathbf{q}_k$ denotes the joint configuration at time instance t_k, $\dot{\mathbf{x}_k}$ the task space velocity and $t_{k+1} = t_k + \Delta t$.

Euler Method for Ordinary Differential Equations (ODE)

Closed form analytical solutions of nonlinear ODE are often unavailable. Numerical integration approximates the solution of ODEs numerically with an accuracy that is usually sufficient for applications such as the simulation of dynamic systems. Without loss of generality, a first order differential equation of the form

$$\dot{x}(t) = f\left(t, x\left(t\right)\right), \; x\left(t_0\right) = x_0 \tag{14}$$

is considered in which time t denotes the independent variable. The condition $x\left(t_0\right) = x_0$ denotes the initial value of the function $x(t)$. These type of *initial value problems* are solved numerically using *linear multi-step methods* or *Runge-Kutta methods*.

By replacing the derivative in equation 14 by the finite difference approximation, one obtains:

$$\dot{x}\left(t\right) \simeq \frac{x\left(t + \Delta t\right) - x\left(t\right)}{\Delta t} \tag{15}$$

where Δt denotes a step size, by rearranging and choosing a step value and setting: $t_n = t_0 + n\Delta t$ and $x_n = x(t_n)$, yields that one step of the Euler method from t_n to t_{n+1} is

$$x_{n+1} \simeq x_n + f\left(t_n, x_n\right) \Delta t \tag{16}$$

The smaller the step size Δt the more accurate the approximation, however resulting in a higher number of iterations and an increase in the computational effort.

Resolved Rate Motion Control

Now, please complete tasks 11 to 19 in the template.

Notes:

- Figure 2 for tasks 14, 16,17

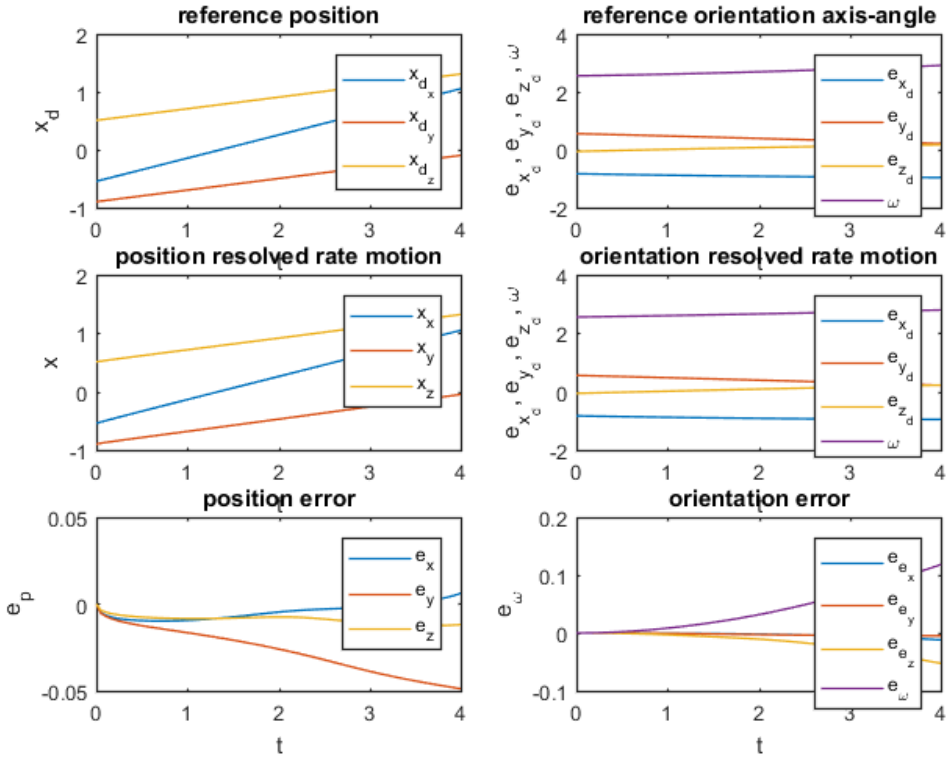

Figure 2: Reference trajectory in task space

Advanced Resolved Rate Motion

A more advanced resolved rate motion scheme (open-loop) considers the variation of the Jacobian with the pose and relies on the integration.

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{J}_w(\mathbf{q}_k)^{-1}\dot{\mathbf{x}}_d\Delta t \tag{17}$$

The key difference is that this scheme employs the Jacobian $\mathbf{J}_w(\mathbf{q}_k)$ of the current joint configuration $\mathbf{q}_k$

Now, please complete tasks 20 to 23 in the template.

References

[1] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[2] Kenneth Waldron and James P. Schmiedeler. Kinematics. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 9–33. Springer, 2008.