HANDS-ON INTERVENTION:
Vehicle-Manipulator Systems

Lab 2: Resolved-rate motion control

Patryk Cieślak
patryk.cieslak@udg.edu

Exercise 1 (3 points)

The goal of the exercise is to implement a kinematic simulation of a planar robotic manipulator, based on the skeleton code (attached to the instruction) and the knowledge obtained during Lab 1. The missing parts of the program include a function implementing the Denavit-Hartenberg formulation and the function used to update the position of the robot's links, based on the transforms computed by the former.
A screen capture of an exemplary simulation output is presented in Fig. 1.

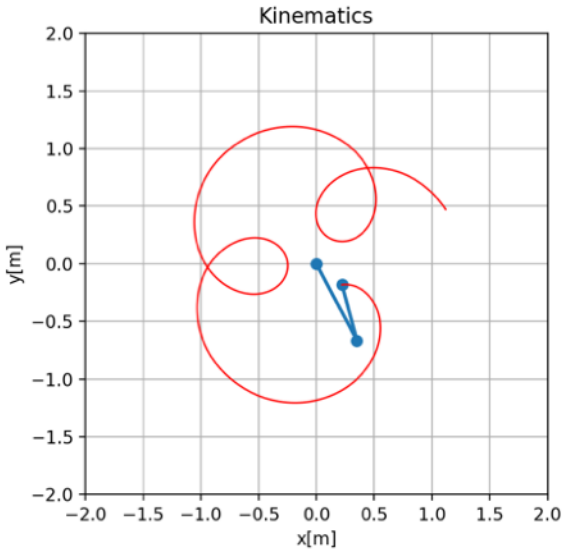

Fig 1. Simulation of a planar robot.

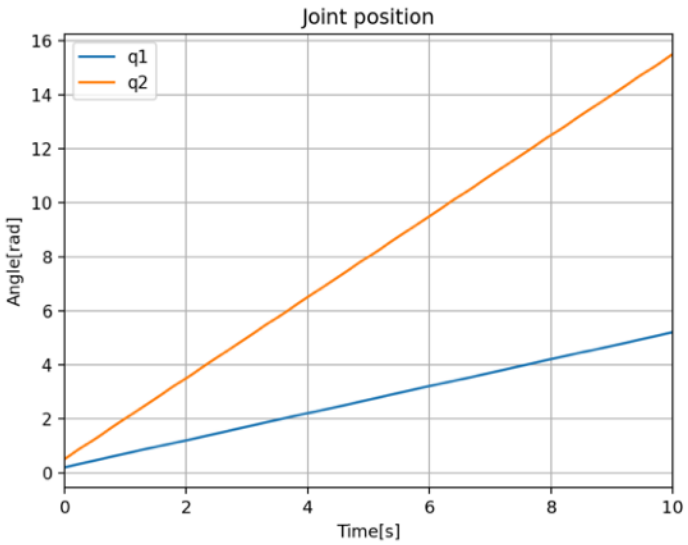

Fig 2. Positions of robot's joints during simulation.

The following elements have to be included in the simulation program:

1) Import of necessary libraries.
2) Definition of Denavit-Hartenberg parameters of a planar manipulator and its initial state.
3) Definition of parameters of the simulation.
4) Implementation of a function generating a chain of transformation matrices according to the Denavit-Hartenberg notation. Input to the function are vectors of DH parameters.
5) Visualisation of the robot structure using the animation functionality of *Matplotlib*.
6) Visualisation of end-effector path on the plane.
7) Positions of the robot's joints changing in realtime, according to arbitrary functions.
8) Second, separate plot, displayed after the simulation is finished, presenting evolution of joint positions over time.

9) All plots with titles, labeled axes, proper axis limits, grid.
10) Simulation without repeats.

The following elements have to be included in the report:

1) Drawing of the robot model, including DH parameters and coordinate systems.
2) Code of the simulation program (well formatted and commented in detail).
3) Plot presenting the visualisation of the robot structure in motion (see Fig. 1).
4) Plot presenting the evolution of robot's joints positions over time (see Fig. 2).

Exercise 2 (5 points)

The goal of this exercise is to implement the resolved-rate motion control algorithm, to control the robot simulated in Exercise 1. The main tasks of this exercise include implementations of the recursive computation of geometrical Jacobian and the control feedback loop. The base for this exercise is the code of Exercise 1.

The following new elements have to be included in the program (added to the code of Exercise 1):

1) Definition of the desired Cartesian position of the robot's end-effector.
2) Implementation of a function computing the geometrical Jacobian of the robot's end-effector, for the given chain of kinematic transformations (DH) and a vector of flags defining types of robot's joints.
3) Realtime update of robot's joints positions based on the resolved-rate motion control algorithm.
4) Control problem solution implemented using 3 different methods: transpose, pseudoinverse, DLS.
5) Separate plot, displayed at the end of the simulation, presenting the evolution of the norm of the control error over time.

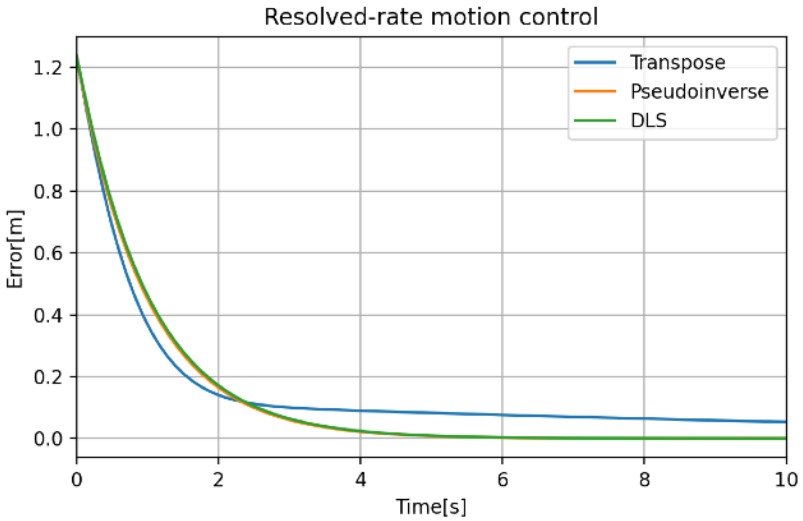

Fig 3. Evolution of the control error.

The following elements have to be included in the report:

1) Code of the program (well formatted and commented in detail).
2) Plot presenting the visualisation of the robot structure in motion (see Fig. 1).
3) Plot presenting the evolution of the control error norm over time, for all three methods used to solve the control problem (on one plot!), see Fig. 3. It can be achieved by saving the output of three simulations (saving the Numpy arrays to files), all with the same initial conditions and desired end-effector position, using different solution methods. The files can then be loaded by a simple program and the data plotted.

Questions (2 points)

1. What are the advantages and disadvantages of using kinematic control in robotic systems?
2. Give examples of control algorithms that may be used in the robot's hardware to follow the desired velocities of the robot's joints, being the output of the resolved-rate motion control algorithm.