Universitat de Girona

IRSLab MRS

HANDS-ON INTERVENTION:
Vehicle-Manipulator Systems

Lab 5: Task-Priority kinematic control (2A)

Patryk Cieślak
patryk.cieslak@udg.edu

Exercise 1 (5 points)

The goal of the exercise is to extend the implementation of the recursive Task-Priority algorithm to include support for set-based (inequality) tasks. An obstacle avoidance task should be developed and tested on a simulated 3-link planar manipulator, for a few obstacles defined in the workspace.

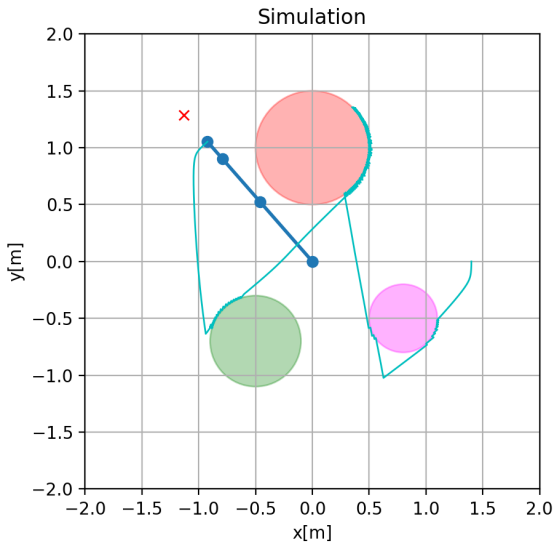

Fig 1. Simulation of the manipulator, including end-effector goal and obstacles.

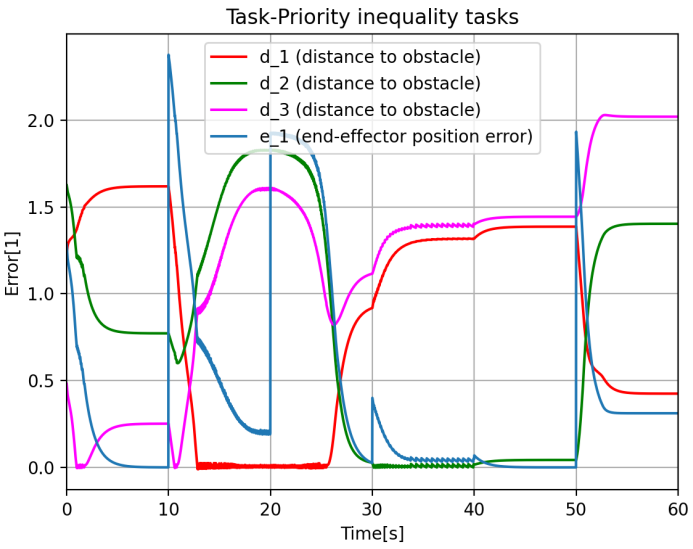

Fig 2. Evolution of the TP control error and distance to obstacles, over time.

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

1) Import of necessary libraries.
2) Definition of Denavit-Hartenberg parameters of a 3-link planar manipulator and its initial state.
3) Definition of parameters of the simulation.
4) Augmentation of the base Task class to include checking if task is active (method: *bool isActive()*).
5) Implementation of the Obstacle2D class, being a subclass of Task, representing a two-dimensional circular obstacle avoidance action (inequality task).
6) Augmentation of the recursive formulation of the Task-Priority algorithm to include inequality tasks.
7) Definition of a task hierarchy including few obstacle avoidance tasks and the end-effector position task.
8) Visualisation of the robot structure using the animation functionality of *Matplotlib*.
9) Visualisation of the desired end-effector position and the obstacle geometry on the plane.

10) Second, separate plot, displayed after the simulation is finished, presenting evolution of end-effector position task error and distances to all of the obstacles, over time.
11) All plots with titles, labeled axes, proper axis limits, grid.
12) Random desired end-effector position, initialised on simulation start (inside *init()* function).
13) Simulation repeating every 10s.

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 task error and obstacle distances over time (three obstacle avoidance tasks, see Fig. 2).

Exercise 2 (5 points)

The goal of this exercise is to implement another set-based task - the joint limits task. It should be tested using the same manipulator model.

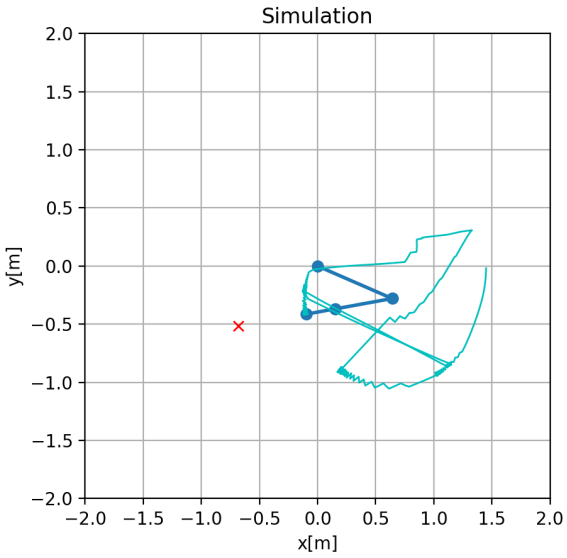

Fig 3. Simulation of the manipulator, including end-effector goal.

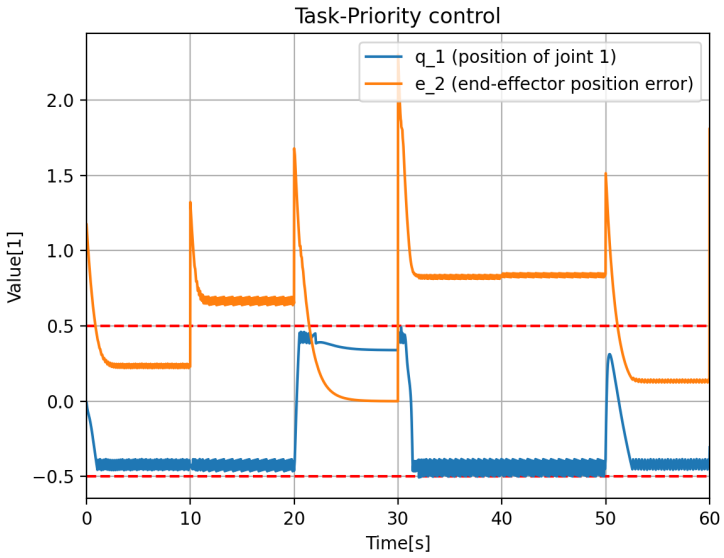

Fig 4. Evolution of the end-effector position error and the first joint position (including limits).

The following elements have to be added to the program:

1) Implementation of the joint limits task (subclass of class Task), as a set-based task (inequality).
2) Implementation of a task hierarchy including two tasks: joint limits for joint 1 and end-effector position.
3) Plot displayed after the simulation is finished, presenting evolution of end-effector position task error and position of the first joint (including limits), over time.

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. 3).
3) Plot presenting the evolution of end-effector position task error and position of the first joint over time, including joint limits (see Fig. 4).