Numerical Activity 1 (NA1): Simulation and Trajectory Planning

Numerical activity NA1 will be divided into two main parts, that are:

1. Simulation
2. Trajectory planning

You can consider the following nominal parameters:

- Wheels radius: $r = 0.03\ [m]$
- Wheels distance: $d = 0.165\ [m]$

Optional material must be delivered in a single zip file.

PART 1: SIMULATION

In this set of activities, you have to set up a Simulink model that simulates the TurtleBot and test elementary maneuvers.

Experiments:

1.1 Prepare a Simulink file that simulates the TurtleBot in continuous time, using as input the wheels' speed. The model is composed of these main blocks:
- "DDR model", input: wheels speed, output: kinematic inputs (v and ω);
- "Unicycle model", input: kinematic inputs + current $\boldsymbol{q}$, output: $\dot{\boldsymbol{q}}$;
- "Visualization", blocks needed to visualize what the robot is doing.

Prepare a MATLAB script to set all necessary variables, such as the initial position and velocity of the robot, input parameters, and duration of the simulation. Simulate the following trajectories using the step function as input:
- $w_L = \omega_R = a$ with $a \neq 0$
- $w_L = a = -\omega_R\ a \neq 0$
- $w_L = a$ and $w_R = b$ with $a \neq b$ (try both $a > b$ and $b < a$)

Try these trajectories by variating initial conditions and inputs parameters, and verify that you obtain what you expect. In the case $w_L = \omega_R = a$ plot the unicycle orientation using both unwrapped and wrapped angles.

1.2 Prepare a Simulink file that simulates the TurtleBot in continuous time, using as input the kinematic inputs (v and ω). The model is composed of these main blocks:
- "Unicycle model", input: kinematic inputs + current $\boldsymbol{q}$, output: $\dot{\boldsymbol{q}}$;
- "DDR model", input: wheels speed, output: kinematic inputs (v and ω);
- "Visualization", blocks needed to visualize what the robot is doing (including wheel angles).

Prepare a MATLAB script to set all necessary variables, such as the initial position and

velocity of the robot, input parameters, and duration of the simulation. Simulate the following trajectories using the step function as input:
- $v = a$ and $\omega = 0$ with $a \neq 0$
- $v = 0$ and $\omega = a$ with $a \neq 0$
- $v = a$ and $\omega = b$ with $a \neq 0$ and $b \neq 0$

Try these trajectories by variating initial conditions and inputs parameters, and verify that you obtain what you expect.

1.3 Add to the model developed for the previous exercise (EX 1.2) blocks that translate $\boldsymbol{q}\ \dot{\boldsymbol{q}}\ \boldsymbol{u}$ in the chained form state/input $\boldsymbol{z}\ \dot{\boldsymbol{z}}\ \boldsymbol{v}$. Test the model with the same input trajectories used in EX 1.2 and verify that you obtain what you expect.

Suggestion

For each experiment, organize the program flow in the following way:

- Write a script that setup parameters
- In the Simulink file, exploit MATLAB functions

See, for example, the template files uploaded for Ex. 1.1

Optional material to deliver

Run the model of EX 1.3 applying the input $v = 0$ and $\omega = 2$ for $T_s = 1[sec]$, with initial condition $\boldsymbol{q}(0) = [1\ 1\ 0]^T$. Save the data and write a script that plots on the same figure the evolution of $\boldsymbol{q}$ and $\boldsymbol{u}$ and $\boldsymbol{z}$ and $\boldsymbol{v}$ (use subplot and report on the left $\boldsymbol{q}$ and $\boldsymbol{u}$ and on the right $\boldsymbol{z}$ and $\boldsymbol{v}$). Name the script EX_opt_1.

PART 2: MOTION PLANNING

In this set of activities, you have to write a set of MATLAB functions that, based on differential flatness, implement the following functionalities:

- Given a trajectory of cartesian flat output obtain the correspondent kinematic state/input $\boldsymbol{q}(t)\ \ \boldsymbol{u}(t)$
- Given a trajectory of chained form flat output obtain the correspondent kinematic state/input $\boldsymbol{q}(t)\ \ \boldsymbol{u}(t)$
- Given an initial and final configuration $\boldsymbol{q_i}\ \boldsymbol{q_f}$, white a function that plans a trajectory form $\boldsymbol{q_i}$ to $\boldsymbol{q_f}$ using the cartesian polynomials strategy explained.
- Given an initial and final configuration $\boldsymbol{q_i}\ \boldsymbol{q_f}$, white a function that plans a trajectory form $\boldsymbol{q_i}$ to $\boldsymbol{q_f}$ using the chained form flat output strategy explained.

- Timing law: constant velocity and trapezoidal velocity

Experiments:

2.1 Use the functions implemented to plan the following trajectories with cartesian polynomials (plot the trajectories both w.r.t. time and in the 2D space):

2.1.1 Line change, $\boldsymbol{q_i} = \left[-1\ -1\ \frac{\pi}{2}\right]^T$ to $\boldsymbol{q_f} = \left[1\ 1\ \frac{\pi}{2}\right]^T$ variating k_i and k_f

2.1.2 Re-orientation, both in the origin and other configurations

2.2 Use the functions implemented to plan the following trajectories with chained form flat output (plot the trajectories both w.r.t. time and in the 2D space):

2.2.1 Movement from $\boldsymbol{q_i} = \left[0\ 0\ -\frac{\pi}{3}\right]^T$ to $\boldsymbol{q_f} = \left[1\ 1\ \ \frac{\pi}{3}\right]^T$

2.2.2 Movement from $\boldsymbol{q_i} = \left[0\ 0\ \ \pi + \frac{2\pi}{3}\right]^T$ to $\boldsymbol{q_f} = \left[1\ 1\ \ \frac{\pi}{3}\right]^T$

2.2.3 Movement from $\boldsymbol{q_i} = \left[1\ 1\ -\frac{\pi}{3}\right]^T$ to $\boldsymbol{q_f} = \left[2\ 2\ \ \frac{\pi}{3}\right]^T$ (compare the input with the ones of EX 2.2.1)

2.2.4 Re-orientation: in the origin and in another configuration (with the same rotation)

2.2.5 Same trajectories of EX 2.2.3 and EX 2.2.4, but planning w.r.t. to a reference frame translated in the in the initial position.

2.3 Prepare a Simulink file with the unicycle model. Feed the model with the inputs computed by planning functions in EX 2.1 and EX 2.2 and verify that the robot follows the planned trajectories.

2.4 Differential flatness for feedforward: design an eight-shaped trajectory, and use differential flatness to compute the ideal inputs necessary to follow the trajectory. Use the Simulink model to verify the effectiveness of the planned inputs.

Suggestion

Solve the tasks 2.1 and 2.2 using script and functions. Save functions on an "utils" folder. You can use visualization files uploaded on stem to plot results.
For task 2.3 use the Simulink template provided on stem.

Optional material to deliver

- Run EX 2.1.1 using a constant velocity timing law with $T_s = 1[sec]$, and $k_i = k_f = 10$. Adapt k_i so that the planned trajectory does not collide with an obstacle centered in the origin, with square shape and side length $1[m]$. Check the obtained wheels speed, and modify T_s to obtain a wheels speed lower than $15[rad/sec]$. Save the data and write a script that plots
    1. The evolution of $\boldsymbol{q}$, $\boldsymbol{u}$, ω_L and ω_R
    2. The 2D trajectory performed by the robot.

  Name the script EX_opt_2.1.