ME449 Assignment 6

Author: Conor Hayes

Introduction

Contents

1. PDF file (this file) -- see sections below
2. commented code is contained in `code` directory
3. output CSV's + plots are in the top-level submission zip, and also are placed in `code/a6-output/*` when the code is run
4. videos are in the top-level submission zip

Running the code

using `uv` (no virtualenv needed)

```
uv sync
uv run assignment_6.py
```

using `pip` (should use virtualenv)

```
pip install -r requirements.txt
python assignment_6.py
```

Code Overview

This code simultates a controlled trajectory between two end effector poses. In order to do so, it takes input data from `config/a6_demo1.toml`, where the EE poses and other configuration data is specified. It then places the resulting data inside the `a6-output` folder under a subfolder named after the trajectory type (e.g. `SCREW_CUBIC`), overwriting any data that's already there. The contents of that output are:

- 2 figures (`*.png`) plotting the results of the run
- `coppellia.csv` - a csv file suitable for pasting into Scene 2
- `errors.csv`, `joint_angles.csv`, `joint_torques.csv` - error, joint angle, and joint torque vs. time data outputted by the simulation.
- a copy of the config used to generate that data.

The controller used to follow these trajectories, in all cases, is a computed torque controller configured by PID gains operating in joint space, as provided by the Modern Robotics library.

This means that it receives as input a desired position, velocity, and acceleration in joint space, and outputs joint torques to apply to maintain that trajectory at each timestep.

3 Example Runs

All runs share the following configuration:

- move the EE from T_A to T_B (same poses used) for all
- same start EE pose T_s is used for all
- use joint-space computed torque control with the ground-truth UR5 dynamics model.
- 5s trajectory duration, .01s timestep, 8 euler steps per timestep

The runs vary control gains, joint torque limit (modeled here as a single scalar applied to all joints), and trajectory type.

1 - Screw Cubic (limited torque, balanced gains)

A first pass was taken using a screw cubic path and balanced PID gains, with torque limited to 120N*m, like so:

```
Kp = 20
Ki = 10
Kd = 18
torque_limit = 120.0
```

Time series plot 3

Controller performance

As can be seen above, when the output saturates to the 120N*m torque limit, the controller can deviate significantly from the desired trajectory. Performance in this case is fine after the initial convergence to the desired trajectory, but still contains the occasional sudden jerk away due to saturation (see about 4.1 seconds in).

2 - Cartesian Quintic (limited torque, balanced gains)

Using the same gains & torque limits as the above with a cartesian trajectory plan results in the following:

Time series plot 3

Controller performance

This performance is very similar to example 1 above, except following a cartesian path. At the start, there is a sudden jerk dragging the EE to the desired trajectory, after which the control is somewhat smooth, though a little stuttery. The controller experiences a similar jerk to the above at 4.1s

3 - Screw Cubic (unlimited torque, high P gain)

This trajectory follows a screw path and raises the torque limit to 400 N*m, and makes the P gain dominate the controller:

```
Kp = 50
Ki = 0
Kd = 10
torque_limit = 400.0
```

I expected this to take advantage of the increased torque range, but actually it performed extremely well and stayed within a realistic 150N*m torque, as can be seen in the plot here: Time series plot 3

Controller performance

My takeaway from this is that PD control with P term dominating is actually a very viable tuning for this problem.