Assignment 02: Operational-Space Control VS Impedance Control

Andrea Del Prete*- andrea.delprete@unitn.it

April 28, 2020

1 Description

The goals of this assignment are:

- implementing a (Cartesian-space) impedance controller;
- implementing an Operational-space controller;
- comparing the performance of the two controllers with different feedback gains, Coulomb friction, and modeling errors.

2 Submission procedure

You are encouraged to work on the assignments in groups of 2 people. **If you have a good reason to work alone, then you can do it, but this has to be previously validated by one of the instructors.** Groups of more than 2 people are not allowed. The mark of each assignment contributes to 10% of you final mark for the class (i.e. 3 points out of 30).

When you are done with the assignment, please submit a single compressed file (e.g., zip). **The file name should contain the surnames of the group members**, and it must contain:

- A pdf file with the answers to the questions, the **names and ID** of the group members; you are encouraged to include plots and/or numerical values obtained through simulations to support your answers. **This pdf does not need to be long. One or two pages of text should be enough to answer the questions. You can then add other pages for plots and tables.**
- The complete *arc* folder containing all the python code that you have developed.

If you are working in a group (i.e., 2 people) only one of you has to submit.

Submitting the pdf file without the code is not allowed and would result in zero points. Your code should be consistent with your answers (i.e. it should be possible to produce the results that motivated your answers using the code that you submitted). If your code does not even run, then your mark will be zero, so make sure to submit a correct code.

*Advanced Optimization-based Robot Control, Industrial Engineering Department, University of Trento.

3 Operational Space Control (OSC)

Operational-Space control is a well-known approach for controlling the position of the end-effector of a robot manipulator. Given the standard joint-space dynamics of a robot manipulator:

$$M\ddot{q} + h = \tau \tag{1}$$

We can write the Operational-space dynamics:

$$\Lambda\ddot{x} + \mu = J^{\top\dagger}\tau = f, \tag{2}$$

where $\Lambda \triangleq (JM^{-1}J^{\top})^{-1}$ is the Operational space inertia matrix, $\mu \triangleq \Lambda(JM^{-1}h - \dot{J}\dot{q})$ are the bias forces (due to gravity, Coriolis and centrifugal forces), $J^{T\dagger} \triangleq \Lambda JM^{-1}$ is a pseudo-inverse of $J^{\top}$ using M^{-1} as weight matrix, and finally f is our new control input, defined via $\tau = J^{\top}f$.

Given a desired acceleration $\ddot{x}^d$ (typically computed with a linear PD control law), the corresponding value of f is simply computed as:

$$f^d = \Lambda\ddot{x}^d + \mu \tag{3}$$

The corresponding joint torques are given by $\tau = J^{\top}f^d$. Finally, we can add to τ another control signal to stabilize the joint space motion without affecting the Operational space acceleration:

$$\tau = J^{\top}f^d + (I - J^{\top}J^{\top\dagger})\tau_0, \tag{4}$$

where $\tau_0 \triangleq M\ddot{q}^d + h$, and $\ddot{q}^d \triangleq k_{pj}(q^d - q) - k_{dj}\dot{q}$.

4 Impedance Control (IC)

Impedance control tries to make the system behave as a linear impedance:

$$\Lambda\ddot{x} + B\dot{e} + Ke = f_{ext} \tag{5}$$

where $e \triangleq x^r - x$ is the tracking error, f_{ext} is the external force applied to the end-effector, and Λ, B and K are the Operational-space inertia, damping and stiffness. The following control law approximately achieves this dynamical behavior of the end-effector:

$$\tau = h + J^{\top}(B\dot{e} + Ke) \tag{6}$$

Similarly to OSC, an additional control signal can be added to stabilize the joint space motion, resulting in:

$$\tau = h + J^{\top}(B\dot{e} + Ke) + (I - J^{\top}J^{\top\dagger})\tau_0, \tag{7}$$

with the difference that this time τ_0 does not need to include the bias forces h, which are already compensated, so it is simply defined as $\tau_0 \triangleq M\ddot{q}^d$.

4.1 Making the controllers comparable

In order to make OSC and IC comparable in our tests, we should make sure that the gains have a similar effect in the two control laws. By default, this is not the case because in OSC the gains (which act inside $\ddot{x}^d$) are multiplied by Λ before being multiplied by $J^{\top}$. Instead in IC the gains (K and B) are directly multiplied by $J^{\top}$. Since Λ has values greater than 1, this would result in "higher gains" for OSC. Therefore, to make the comparison fair, you must scale the gains of IC by a factor which is similar to the values of the matrix Λ. For the UR5 robot, in our tests, we have observed that the diagonal values of Λ are typically around 8, so we suggest you to add a scalar factor $\lambda = 8$ in your implementation of IC:

$$\tau = h + J^{\top}\lambda(B\dot{e} + Ke) + (I - J^{\top}J^{\top\dagger})\tau_0, \tag{8}$$

5 Code

The template code for the assignment is located in

`code_template/arc/02_basic_control/ex_04_impedance_control.py`

To work properly, this file needs to be used together with a new version of the simulator, which you can find in

`code/arc/utils/robot_simulator.py`

This file `ex_04_impedance_control.py` already contains all the code for comparing OSC and IC. The only part that needs to be implemented are the two control laws inside the for loop. Once the two controllers are implemented, you can run the script.

The two controllers are compared using three Coulomb friction values (0, 2 and 4% of the maximum torque at each joint), and two gains for each friction value. Therefore, at each run, 12 simulations are executed, as specified by these lines in the script:

```
tests = []
tests += [{'controller': 'OSC', 'kp': 50,  'friction': 0}]
tests += [{'controller': 'IC',  'kp': 50,  'friction': 0}]
tests += [{'controller': 'OSC', 'kp': 100, 'friction': 0}]
tests += [{'controller': 'IC',  'kp': 100, 'friction': 0}]

tests += [{'controller': 'OSC', 'kp': 100, 'friction': 2}]
tests += [{'controller': 'IC',  'kp': 100, 'friction': 2}]
tests += [{'controller': 'OSC', 'kp': 200, 'friction': 2}]
tests += [{'controller': 'IC',  'kp': 200, 'friction': 2}]

tests += [{'controller': 'OSC', 'kp': 200, 'friction': 4}]
tests += [{'controller': 'IC',  'kp': 200, 'friction': 4}]
tests += [{'controller': 'OSC', 'kp': 400, 'friction': 4}]
tests += [{'controller': 'IC',  'kp': 400, 'friction': 4}]
```

Only the proportional gain k_p is specified because the derivative gain is always chosen as $2\sqrt{k_p}$. Please do not change any parameter (such as gains and time step) in the script and in the configuration file `ex_4_conf.py`. You can comment/uncomment the lines defining the tests to run, in case you want to temporarily remove some of them. You can also enable/disable the viewer (see `use_viewer` and `simulate_real_time` in the configuration file) and the plots.

The script prints and plots the average tracking error obtained with each simulation, which can be used to compare the performance of the controllers.

Try to answer the following questions:

1. What can you say about the relative performance of OSC and IC in the different settings? Which controller performed best without friction? Which one performed best with friction? Report the values of the tracking errors that you got in simulation.

2. Set to 1 the flag `randomize_robot_model` in the configuration file and re-run the script. This flag enables the random modification of the inertial parameters of the simulated robot, making it different (at most 30%) from the model used by the controllers. How did the performance of OSC and IC change? Which of the two controllers was most robust to the introduced modeling errors?[1]

[1] Every time that you re-run the script you will get slightly different results because of the randomization of the robot model, so you should run the script a few times to avoid basing your answers on a particularly lucky/unlucky case.

3. The impedance control law (6) does not exactly achieve the desired impedance behavior (5). Why? How could you modify the control law to achieve exactly the desired behavior?

4. Try to modify the impedance control law that you implemented to include also the term you added in the previous question. Does this lead to an improvement in the tracking performance? Why?

5. (Optional) Which control law would you choose to implement on a real robot? Why? And how could you try to improve its performance? Discuss different ways, pros and cons, possibly showing plots and tracking errors that you obtained by testing your ideas in simulation.