Learning for Adaptive and Reactive Robot Control
Instructions for Practical 3

Professor: Aude Billard

Spring Semester

Introduction

This practical takes place in the LASA robot room ME A3 455, using a Franka Emika Panda robot with 7 degrees of freedom. In this practical you will teach a task to a Panda robot using kinesthetic demonstrations and the Dynamical Systems formulation taught throughout the class. You will then control the robot using the learned DS, and play with simple obstacle avoidance.

The software structure for the practical is as follows: we are using ROS2, with a node in MATLAB sending cartesian velocity commands computed from a dynamical system, and a C++ node converting these commands into joint torques for the robot. You will work mainly with the MATLAB part of the application, but you will also use the terminal to access ROS functionalities.

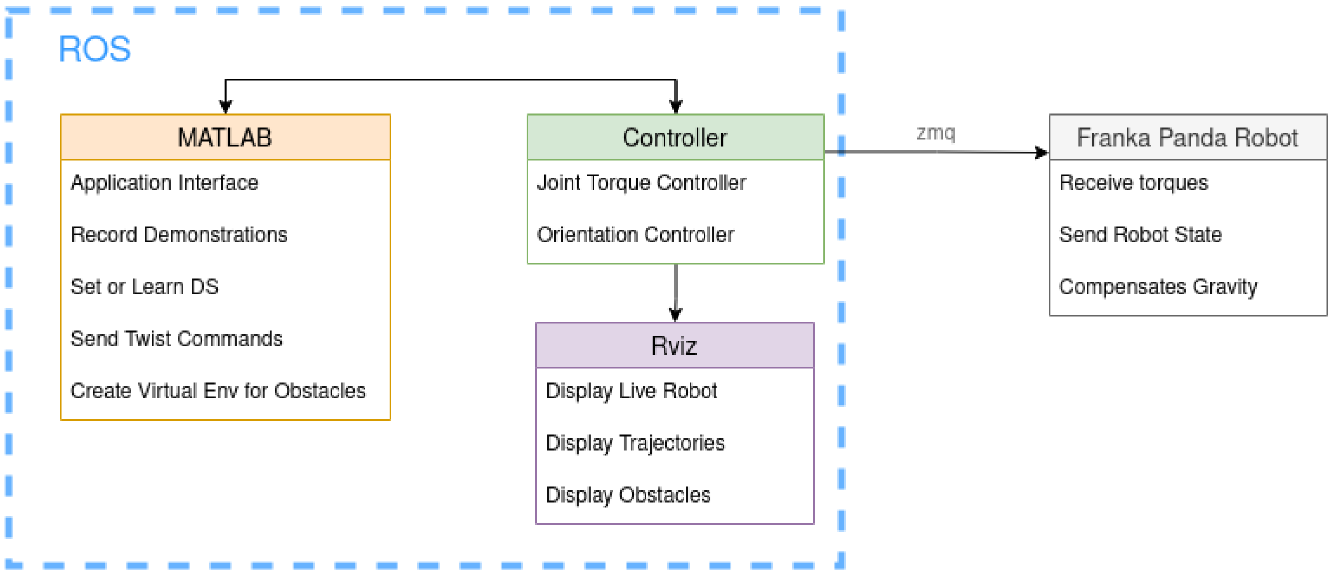

Figure 1: Software Structure

The robot is controlled by sending joint torques commands. Our controller tracks the desired twist (linear and angular velocity) by feedforward linearization and a correction on twist error. We also control the null space of the robot to stabilize the posture around the default configuration $\mathbf{q}_0$:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_f + \mathbf{G}(\mathbf{q}) + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{D}\dot{\mathbf{q}} + \boldsymbol{\tau}_0 \qquad \text{with} \quad \boldsymbol{\tau}_0 = k_0 * \mathbf{N}^{\#} * (\mathbf{q} - \mathbf{q}_0) \tag{1}$$

with the feedforward terms being $\boldsymbol{\tau}_f$ the estimated joint torques frictions, $\mathbf{G}(\mathbf{q})$ the gravity torques, and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ the Coriolis torques.

In the last part of the practical, we will use the Motion Capture cameras set up around the room to track the obstacles relative to each robot, using the reflective markers. **Please DO NOT MOVE the markers.** Your tracked osbtacle is already set up in the MATLAB environment, with its ObstacleNumber being 1. Be careful to not change its parameters until Part 3 of the practical.

Safety Protocol

- Always keep the emergency STOP button within reach and ready to use. Do not hesitate to use it for any reason. Once pressed, the lights of the robot turn white: Call an assistant to unlock the robot.
- Only touch and move the robot when the light is blue.
- If the light on the end effector starts flickering, immediately stop trying to to move the robot. Press the emergency STOP button and call an assistant.
- Whenever you are changing parameters, make sure to first press **"S"** to stop sending commands to the robot.

Application Interface

During this practical, you will be controlling everything using a MATLAB GUI which is created when launching the `practical3_main.m` script. With this, you can record demonstrations, learn a DS using those demonstrations and send commands from your DS to the robot, among other things. Important notes :

- To use the GUI correctly you must first select the interface window, called Figure 1. Make sure your last mouse click was inside this window before using the keyboard.
- Matlab will display information in the command window whenever a new action begins, use this to verify that your are using the GUI correctly.
- You only need to run the main script once, as all functions can be called directly from the command window and code can be edited without needing to close the app.
  All available functions and properties are listed at the end of the `practical3_main.m` script.
- Two ROS commands will also be used in the provided terminal. These commands are specified both in the instructions and the `practical3_main.m` script. You can use CTRL+Shift+V to paste them in the terminal, and CTRL+C to stop the command.

Part 1: Setting up a safe environment and a compliant robot controller

TASK 1 Program a linear DS in Matlab to reach a position in task space. Compare your results with what you achieved in simulation during the semester.

1. Press "**S**" in the MATLAB window to be sure the robot position controller is stopped. Move the robot by hand in the workspace. Which degrees of freedom are controlled, and which ones are free to move ?
2. Using your hand, move the robot end effector in the workspace to get a sense of its boundary. Move slowly the robot to the elbow joint limits. It should automatically stop and blink blue. Call an assistant to unlock it.
3. Using your hand, move the end effector to a target position. In the ROS terminal, you can type
   `ros2 run matlab_bridge print_robot_state`.
   This runs a small script to display the 3D end effector position in the terminal.
4. In MATLAB's command window, call the function:
   `myHub.myDS.setLinearDS([0.5, 0.0, 0.5])` with the desired attractor as a 3D position vector to program your linear DS.
5. Press the key "**P**" (Play) to start the position controller. The robot should now follow your DS to reach the specified target. Play around with the robot to answer the following questions:
   - Does this control method cover the entire workspace of the robot? How well does it respond to perturbations?
   - What happens if you place the end effector of your robot on the side opposite your attractor, then press "**P**"? How could you remedy this issue ?

Whenever you are changing parameters, make sure to first press **"S"** to stop sending commands to the robot.

TASK 2 Add fixed obstacles to your environment to prevent collisions. Test the precision and compliance of the implemented controller.

1. Measure the base of the robot arm. Using the MATLAB console, add a vertical cylinder centered on the robot. This will simulate the base of the robot a an obstacle to avoid for the end effector when planning trajectories.
   You can use the function `myHub.addCylinder(position, radius)` with `radius` the radius of the cylinder, and `position` the 3D vector of it's center position. The [0,0,0] position is at the base of the robot. You can call
   `myHub.updateObstacle(obstacleNumber, newPosition, newDimensions, newRho)`
   if you need to change the obstacle's position, dimension, or sensitivity (rho value). All values are in meters. `ObstacleNumber` is the id of each obstacle, **starting at 2**. Osbtacle number 1 is the tracked osbtacle.

```
% Create a vertical cylinder of radius 10cm with default rho value ...
    of 1.
myHub.addCylinder([0; 0 ;0], 0.1);

% Move the cylinder (obstacle 2) up by 50cm nad change its radius ...
    to 0.15
myHub.updateObstacle(2, [0; 0; 0.5], 0.15)
```

2. In case the modulation is not activated in your DS, you should call the function `myHub.myDS.activateModulation` in MATLAB's command window. Then, press **"P"** to command the robot with your modulated DS. Does the robot now avoid itself?

3. What other fixed obstacle can easily be added to prevent collisions in the workspace? Add it using the MATLAB console. You can see the available commands to do this in the `practical3_main.m` file

```
        % Create a plane centered in 0 with a vertical normal ...
            vector with default rho value of 1.
        myHub.addPlane([0; 0; 0], [0; 0; 1], 1);
```

4. What happens now if you place your attractor close to the table? Does the robot correctly avoid the table?

TASK 3 Observe the impact of different gain values for the joint torque controller.

The low level controller implemented here is a simple torque controller without inertia compensation, as seen in previous exercises :

$$\boldsymbol{\tau} = \mathbf{G}(\mathbf{q}) - (\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\tilde{\mathbf{q}}) \tag{2}$$

1. Now that your environment is setup to be safe, move the robot around in control mode and answer the following questions :
   - What steady-state error does the robot achieve with this controller? Where does this error come from?
   - How compliant is the robot when you perturb it? Can it still reach the target?
2. You can change the scaling value of the gains of the impedance controller using `ros2 param set /matlab_bridge gain_scale 1.0` in the terminal. The initial value is 1.0, the range is [0.1, 2.0].
   - What happens when you set values lower than 1.0? Why?
   - What happens if you set values higher than 1.0? What can happen if the your gains are too high?
   - Set the value back to 1.0 before moving on.

   Warning : If robot starts shaking, press the emergency stop button immediately.

Part 2: Learning from Kinesthetic Demonstrations

TASK 1 Record pick-and-place trajectories so that the robot end effector reaches the top of the square foam coming from above.
You can record a trajectory by pressing the key "**R**" (record) when starting, then "**S**" (stop) when you finish you demonstration.

1. Record two trajectories starting from close initial positions. When you're done, you can press the "**L**" key (Learn) to start learning with SEDS. If you need to modify SEDS parameters, you can do so in the function `learnSEDS()` of the file `DS.m`, and press again the "**L**" key to learn again until you're satisfied with the results.
   - Does SEDS fits well your demonstration?
   - Does it generalizes well on the whole workspace? What would happen if your start from another starting position?

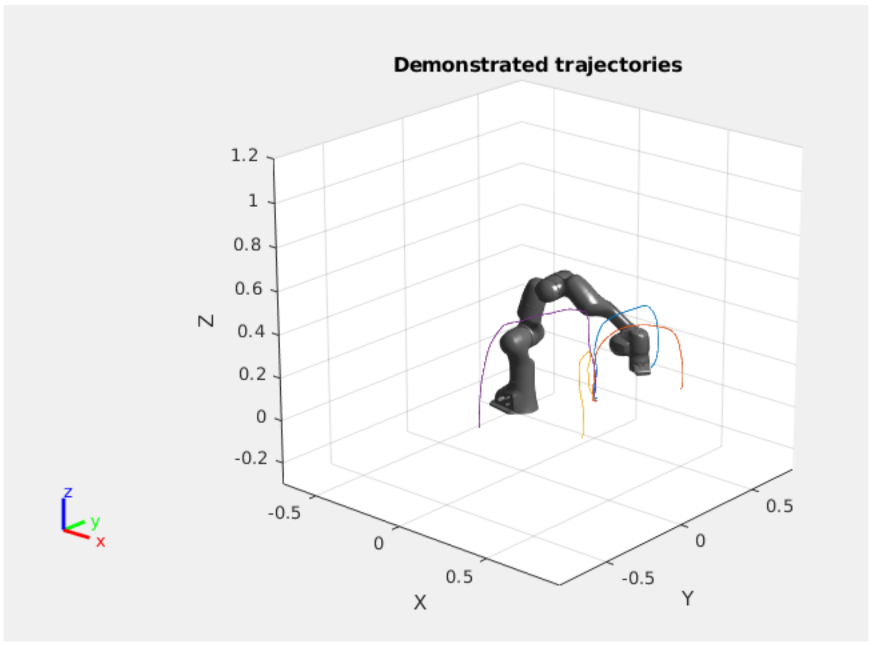

Figure 2: Examples of pick-and-place trajectories

2. Press the "**P**" key to start following your DS and observe the result. In Rviz, you can see in red the open loop trajectory from your starting point and in green the robot's actual path. Do not hesitate to press the emergency stop button before the robot hits anything.
3. Without deleting your previous dataset, record new trajectories starting from different initial conditions to span the whole workspace (see figure 2). Change the learning algorithm by typing `myHub.myDS.algoName = 'LPVDS'`, then start the learning again. Similarly, you can change the LPVDS parameters in the function `learnLPVDS()` in the file `DS.m`.
4. Press the "**P**" key to start following your DS. Answer the following questions:
   - How well can the robot reproduce your demonstrations? What is the main challenge?
   - How many trajectories are needed for generalization? What other factor affects generalization?
   - How does the quality of your DS affect the tracking performance? What could happen if the motion is not generalized over the whole workspace?

TASK 2 Learn a DS to follow a nonlinear path on the table and reach the target without collision.

1. Use the **"X"** key to clear your previous dataset. Record a few trajectories starting at similar position and following the predefined path.
2. Learn the motion using LPVDS and the Physically Consistent Non-Parametric Initialization (`est_options.type = 0`). The optimal learning parameters should already be set.
3. Start the controller to see how well it tracks your DS. Can the robot follow the path well and avoid all the obstacles at the same time?

4. What happens now if you deactivate the modulation?
   You can use the following command in the MATLAB command Window :
   `myHub.myDS.deactivateModulation`

Part 3: Moving Obstacle Avoidance

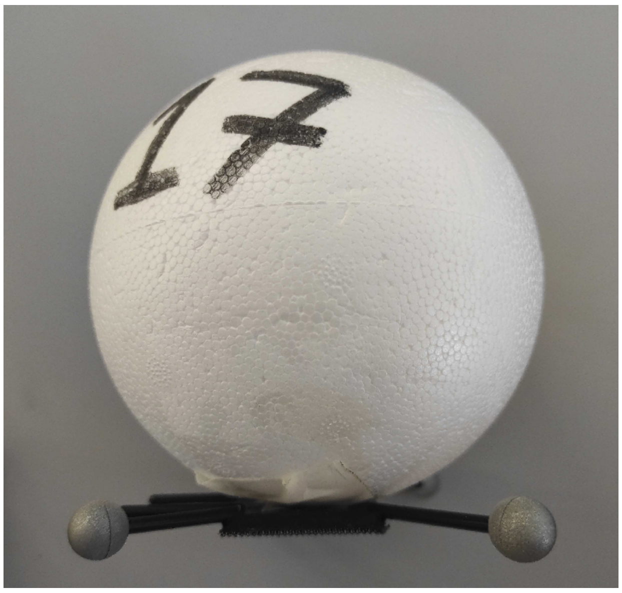

Figure 3: Example of tracked obstacle

TASK 1 Add the tracked obstacle and activate its modulation. It is easier to see the modulation when using a linear DS, as in Part 1.

1. Place the obstacle on the linear DS trajectory. What happens?
2. Now activate its modulation. What changes occur in the robot behavior?
3. You can change the sensitivity `rho` of the tracked obstacle using the command shown below. How does this parameter affect the behavior of the robot?

```
    % Set the rho value of the first obstacle to 0.5
    myDS.myWorld.listOfObstacles(1).rho = 0.5
```

4. What happens if you place the obstacle on the attractor ?

TASK 2 Move the obstacle around while the robot is following the linear DS.

1. How does the robot react to the moving obstacle?
2. What happens if you place your obstacle on the arm of the robot? Does it attempt to avoid it? Why?
3. What happens if you place the obstacle on the robot's trajectory and move it towards the end effector? Why?

References

[1] Aude Billard, Sina Mirrazavi, and Nadia Figueroa. *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach.* MIT press, 2022.