Closed-loop force control/proprioception of robots with embedded sensors

Organizers: Thomas George Thuruthel, Arsen Abdulali (**Correspondence: aa2335@cam.ac.uk**), Kieran Gilday, and Fumiya Iida

Planning of Tutorial

Winter school Dec 13-17

This tutorial: Dec 13 & 16

Tutorial length: 2.5 hours

Participants: max 20 students

Group activity plan:

We create 4 groups, 5 students each, and each group should have one robot platform (incl. one robot control laptop, soft finger, force sensor, etc). Each robot platform consists of UR5 with soft finger (sensors, servo, integrated).

1. Introduction

This instruction is to guide the participants of the workshop that covers setting-up hardware and installing required software and safety measures required to operate robotic devices. The workshop is divided into two parts. First, the participants will learn basic principles of robotic manipulation and sensing in a virtual environment implemented in the Matlab Simulink environment **before** the physical event. At this stage the participants will explore control methods in robotics, acquire the interaction data, and process the recorded data to remove various types of noise and the sensor drift. Second, the participants will apply their knowledge in an actual robotic system that they will later use for a competition.

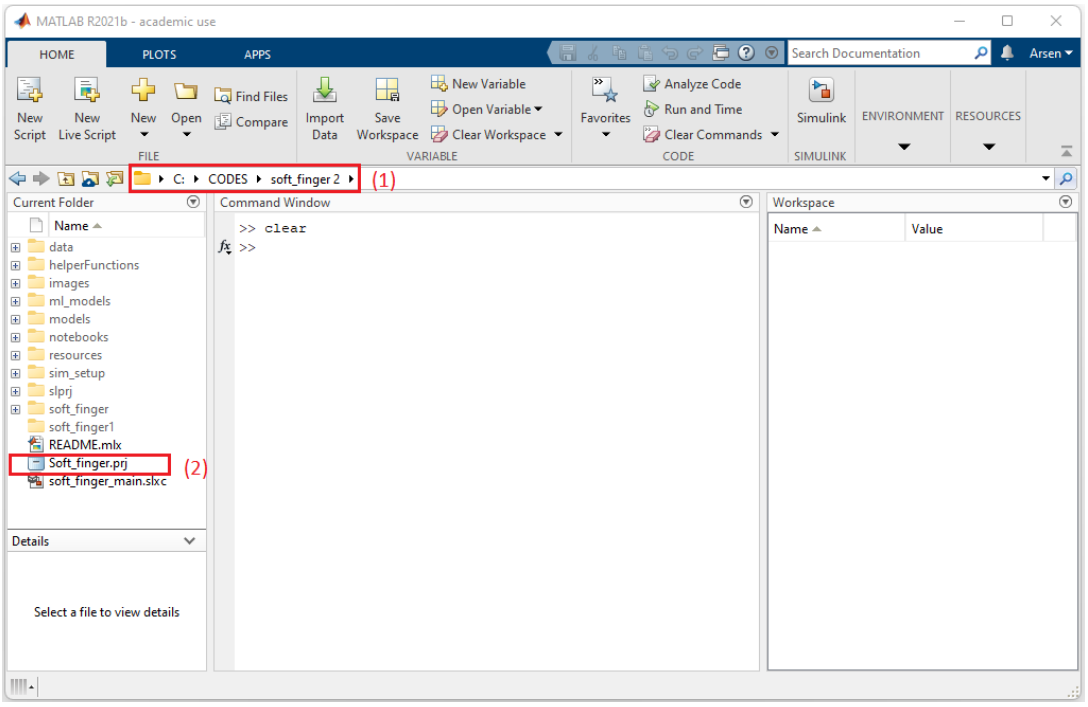
This instruction details most aspects required during the workshop. Taking into account the limited time, we encourage the participants to get familiar with this document in advance and inquire the additional information from the organizing committee or use search engines to learn related theory. The participants are also welcome to ask questions from the instructors during the workshop.

2. Part 1 (Simulation)

Note that the simulation model has only been tested on MATLAB version 2021a or higher.

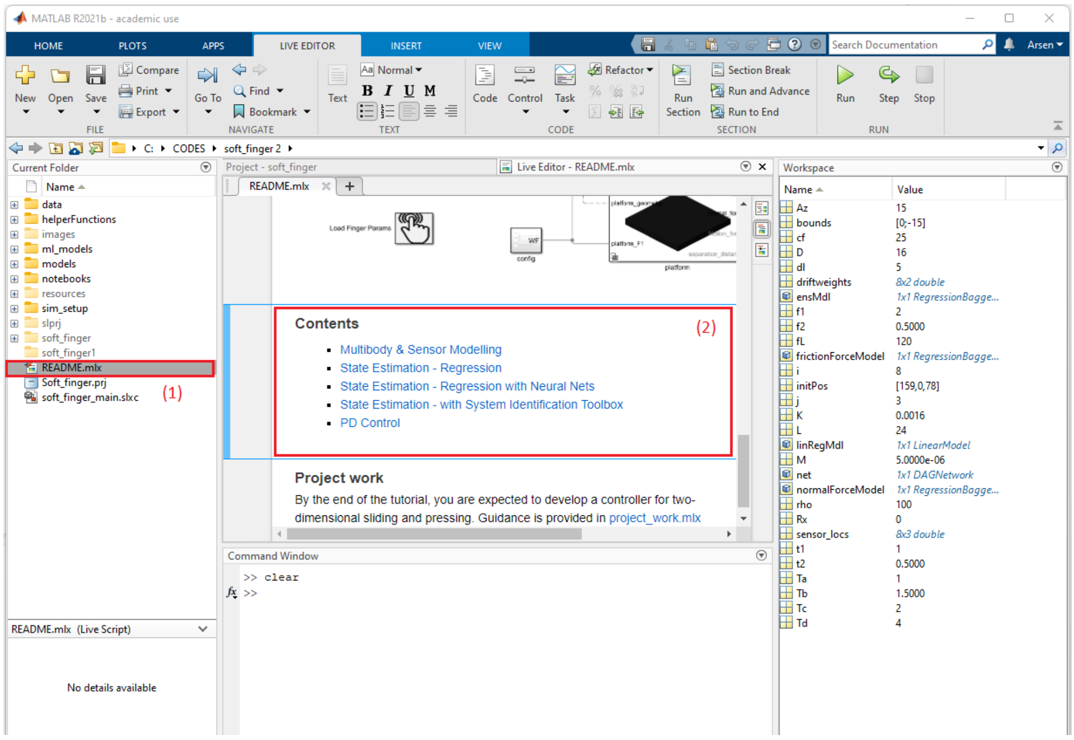
In the first part, the participants will utilize the Soft Finger simulation framework that covers all fundamentals steps needed in closed-loop controller design of sensorized fingers. In order to run the framework, the participants should open the project folder in Matlab as shown in Figure 1.. Then the environment variables of the project are needed to be set by running the “Soft_finger.prj” file as in Figure 1. Then the participant can open the “README.mlx” file and follow the guidelines provided in this file (see Figure 2).

In the section “Contents” of the “README.mlx”, the participants will find the links to subprojects that they need to perform sequentially. Each project contains the basic description of each step. Searching for additional theoretical explanations on the internet would also be beneficial and helpful during an implementation and deployment of the closed-loop controller on Part 2.



```
>> clear
>>
```

Figure 1. Setting-up the project variables of the Soft-Finger framework.



Name
- data
- helperFunctions
- images
- ml_models
- models
- notebooks
- resources
- sim_setup
- slprj
- soft_finger
- soft_finger1
- README.mlx
- Soft_finger.prj
- soft_finger_main.slxc

Contents

- Multibody & Sensor Modelling
- State Estimation - Regression
- State Estimation - Regression with Neural Nets
- State Estimation - with System Identification Toolbox
- PD Control

Project work

By the end of the tutorial, you are expected to develop a controller for two-dimensional sliding and pressing. Guidance is provided in project_work.mlx

```
>> clear
fx >>
```

Name	Value
Az	15
bounds	[0;-15]
cf	25
D	16
dl	5
driftweights	8x2 double
ensMdl	1x1 RegressionBagge...
f1	2
f2	0.5000
fL	120
frictionForceModel	1x1 RegressionBagge...
i	8
initPos	[159,0,78]
j	3
K	0.0016
L	24
linRegMdl	1x1 LinearModel
M	5.0000e-06
net	1x1 DAGNetwork
normalForceModel	1x1 RegressionBagge...
rho	100
Rx	0
sensor_locs	8x3 double
t1	1
t2	0.5000
Ta	1
Tb	1.5000
Tc	2
Td	4

Figure 2. Five sub-project to be studied during the first part of the tutorial.

3. Part 2 (Experimental) – 1.5 Hr

3.1. Hardware Overview

The hardware setup consists of three components (see Figure 3). The first component is the robotic arm, that we utilize to control the translation and orientation of the end-effector. The second component of the setup is the anthropomorphic soft finger with embedded tactile sensors. By controlling the position and orientation of the finger, the participant can interact with the environment which can be sensed by the sensors in the form of contact pressure. The raw data of contact pressure is non-linear and not scaled in physical units. Furthermore, the tactile signal can be considerably affected by the sensor drift and the noise. To solve these two problems, we use the LSTM-based model that can map the tactile signals to meaningful physical units and reduce the temporal distortions due to sensor drift and other noise. To train the LSTM-based model, the ground truth pressure measurements need to be collected during the interaction with the environment. In our case, we measure the ground truth contact forces using the FSR pressure sensor (see Figure 3), which is our third hardware component. Following, we describe each component as well as communication.

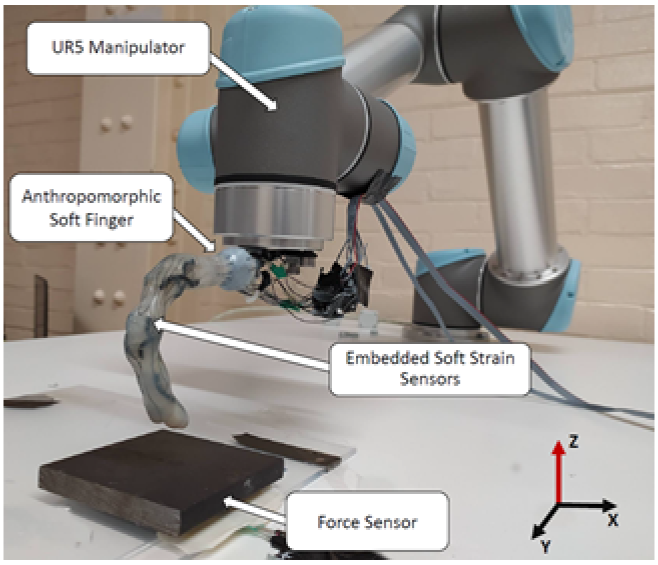


Figure 3 Representative experimental setup

3.2. Data Collection

In order to sample the data using the robotic system, the participants should open the file “batch_sampling.m” in matlab. First two lines in the script consist of the address of the serial port connecting the computer and the ip address of the robotic manipulator. In the system that participants use in the current tutorial, the correct addresses are already set up.
To run the script, the participants should click the “Run” button on the “Editor” tab of the Matlab main window (see Figure 3).

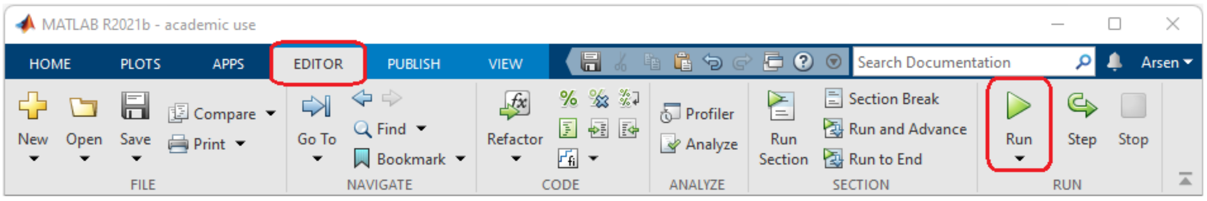


Figure 4. Running the script in Matlab Environment.

After running the script, the robotic arm performs a series of interactions with various levels of vertical translation and velocity. The measured data from both tactile and FSR sensors are then saved into a “data” variable in the Matlab workspace (see Figure 5).

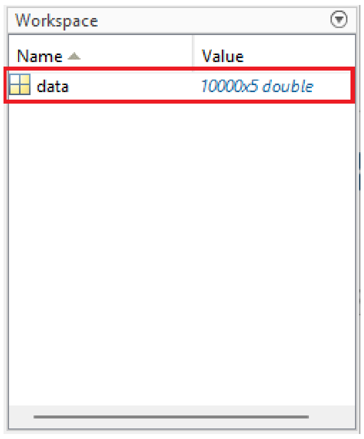


Figure 5. Interaction data collected during the experiment.

This data can be saved to the file or used directly from the Matlab workspace in the next step. In our case, we will load this data for training the lstm model in the following step.

3.3. LSTM Model Training

To train the LSTM model, the participants should open the script file "lstm_sec.m". By clicking the "Run" button on the "Editor" tab, as in the previous step, the participants can start the training process with default hyper parameters of the LSTM network. At the end of the training, the resultant LSTM model will be available in the Matlab workspace and can be used in the next step.

3.4. Closed-loop control

In this step, the participants can utilize the LSTM model from the previous task for closed-loop control. To this end, they can open the "closed_loop.m" script file. This experiment is mostly similar to the first one, except that the FSR sensor is not utilized. This script, on the other hand, uses the lstm model that we obtained in the previous step and converts the tactile signal from the pressure sensors directly to force values, meanwhile removing sensor drift and noise.

3.5. Benchmarking

When the participants got familiar with the code of each step, i.e., data-collection, LSTM network training and closed loop control of the desired force level. At this point, four groups of participants will compete with each other by optimizing the architecture of the LSTM model and tuning the closed-loop controller. Below, we present two benchmarks that will be used to score the performance of each group.

3.5.1. Optimizing architecture of the LSTM model

The main objective of this task is to improve the model of mapping tactile pressure signals to the corresponding ground truth forces. To this end, the participants can apply any changes into the "lstm_sec.m" file. One possible way to improve the model is changing the architecture of the LSTM model and other hyperparameters used for training. Another way could be to apply pre-processing to the raw data. The participants are also welcome to use any methods that they have learned in the Part 1 or found on the internet.

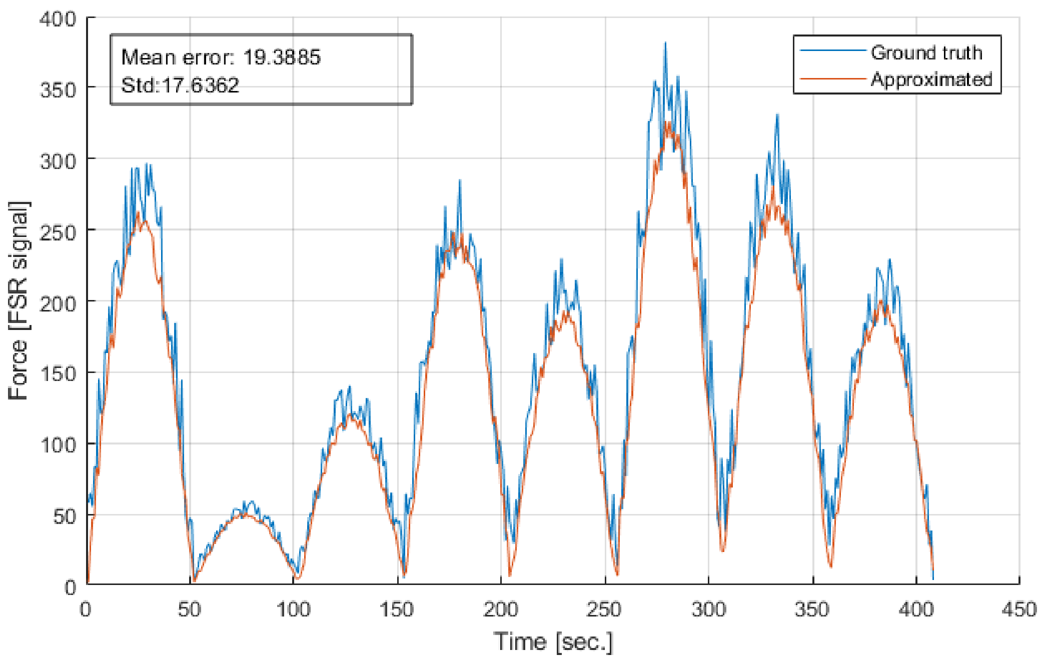


Figure 6: Comparison of ground truth and approximated forces.

To evaluate the performance of the resultant model, the approximated signal will be compared to the ground truth one as in Figure 6. By reducing the mean error at this point will also improve the performance of the resultant closed-loop controller.

3.5.2. Tuning closed-loop controller

At this point, the participant will try to improve the closed-loop controller that is implemented in the file “closed_loop.m”. The participants can start by tuning the gain of the basic proportional controller. Alternatively, the participants can improve the design of the controller, e.g., implementing the PID controller.

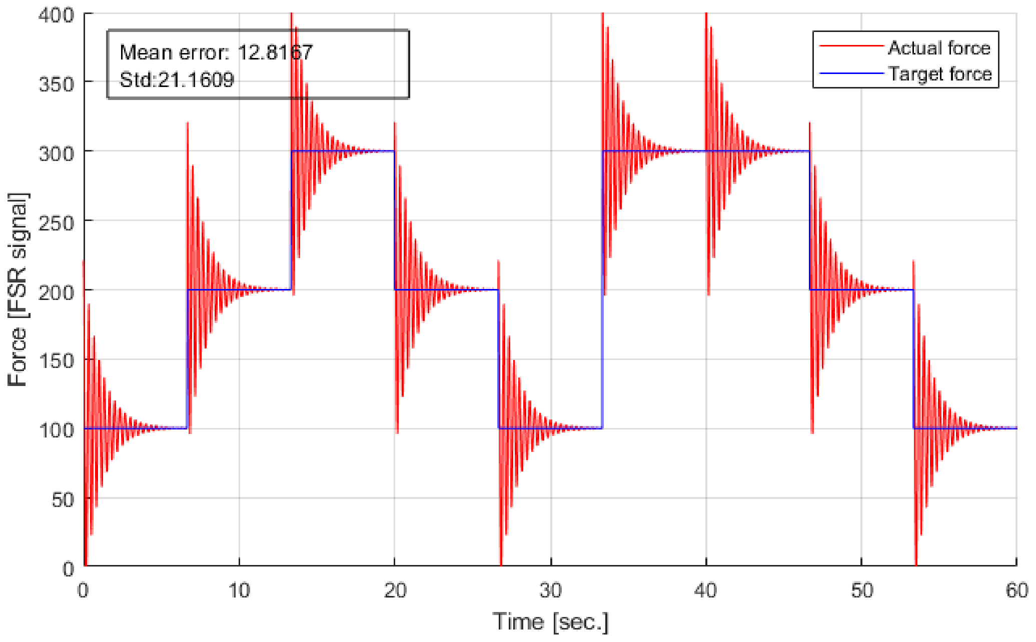


Figure 7: Benchmark of the closed-loop controller performance.

To evaluate the performance of the closed-loop controller, the robotic manipulator needs to reach a series of target force levels as in Figure 7. The resultant force capture by the force sensors will then be compared with the target forces. The error metric that will be used in competition is the mean force error that captures the performance of both the LSTM model and the closed-loop controller. The mean force error drops is that the LSTM predicts the force values accurately so the closed-loop controller reaches the right target. Additionally, the error is reduced when the closed-loop controller reaches the target faster. The team having the least error by the end of the tutorial will be announced to be a winner.