Source Code Guide

February 27, 2017

This document presents an explanation of the folder structure and a brief *How To* guide to run the Iterative Learning Control (ILC) based code that tracks a desired joint trajectory for a Soft Robot. Details on the algorithm can be found here [1, 2].

1 Source Directory Contents

The code is organized in folders as follows:

- directory *"Images"*: data and figures will be stored in this directory. Each trial has a specific directory named *"year_month_day_hour_minute"*;
- directory *"Parameters"*: it contains the .mat file storing the reference and its derivatives and all the configurations files:
  - *"cube_parameters"*: it defines some parameters of qbmove maker pro. For more information please check qbmoves maker pro datasheet at [3];
  - *"model_parameters"*: it defines the number of joints of the robot;
  - *"time_parameters"*: it defines the time intervals;
  - *"input_parameters"*: it defines the preset and it loads the reference trajectory;
  - *"ILC_parameters"*: it defines the input control weight;
  - *"step_parameters"*: it defines all the variables to identify the decoupled model.
- directory *"Functions"*: it contains all the utility functions used by the main scripts;
- directory *"Simulink"*: it contains all the Simulink schemes used;
- directory *"Teaching"*: it contains Matlab scripts and Simulink schemes used to manually generate a reference trajectory;

- script *"ILC_initialization.m"*: matlab script used to initialize the algorithm;
- script *"ILC_algorithm.m"*: matlab function used to run the algorithm. This function has three arguments:
  - number of iterations
  - reset flag: set it equal to 1 to start a new trial (a new will directory be created to store the data), set it equal to 0 to carry on the last learning
  - error threshold: if the threshold is passed, the algorithm stops. The default value of the threshold is 0.

2 How to Use the Algorithm

To use this code you need qbmoves libraries installed on your computer. For more information please visit [3]. To run the algorithm follow these steps:

1. Run Matlab in *"src"* directory.
2. Configure the parameters of the test you want to perform. In particular:
   (a) *"model_parameters"*: change *"N"* depending on the number of actuators of your setup;
   (b) *"time_parameters"*. Note that all time interval should be a multiple of the sample time *"T_sample_init"*. If this condition is not fulfilled, the code will try to approximate the intervals to the next multiple of *"T_sample_init"*. The intervals that need to be set are:
      i. change *"t_fin_init"*. This defines the tracking time, i.e. the time used to track the desired trajectory;
      ii. change *"pre_time_init"*. This is the time interval the robot uses to place himself on the initial position of the reference;
      iii. change *"post_time_init"*. This is a time interval that can be used to reposition the robot on the resting position (i.e. the position that the robot has when turned off);
      iv. change *"T_sample_init"* depending on the sample time you chose. This parameter may be influenced by the performance of the computer used and has to be increased if the QB Pacer and the Simulink clock differ too much (scope *"pacerTime"* inside the *"Robot"* subsystem in every Simulink scheme used). Usually a good choice is 0.001*N;
   (c) *"input_parameters"*: set the preset variable *"preset"*;

(d) *"ILC_parameters"*: set LQR input weight *"R"*;

(e) *"step_parameters"*: change the amplitude of the input step *"Amp_step"* used to identify inertia and damping. Change this parameter only if the identification phase does not converge;

3. Place a .mat file named *"reference"* in *"./Parameters"*. This file should contain the reference trajectory with its first and second derivative. These variables should be timeseries and should be named *"ref"* (°), *"dref"* and *"ddref"* . Note that the end time should be equal to *"t_fin_init"*. Furthermore a good reference trajectory should have its first derivative equal to zero at the starting time.

4. (Optional step) Another option is to generate the reference trajectory using the file in *"Teaching"* directory:

   (a) you should have already defined the variables in *"model_parameters"* and *"time_parameters"*;

   (b) open the Simulink scheme *"./Teaching/TrajectoryTeaching.slx"*;

   (c) open the script *"./Teaching/Reference_Generation.m"* and set *"cut_time"* as you prefer. This is the time interval you have at the begging and at the end of the teaching phase and it will be cut off the trajectory taught;

   (d) run *"./Teaching/Reference_Generation.m"*. When you are ready manually move your robot: you should use the time interval *"cut_time"* to place the robot in the initial position of your trajectory and move it after *"cut_time"* seconds. Repeat this process until you are satisfied with the trajectory. Use the scope in *"./Teaching/TrajectoryTeaching.slx"* to check your trajectory;

   (e) the script will generate the first and second derivative of the recorded trajectory, and it will save it in the *"Parameters"* directory;

   (f) check the generated reference and its derivatives.

5. Run the script *"ILC_initialization.m"*

6. If the initialization fails, check the value of variables J and b. If they are Nan, increase the parameter *"Amp_step"* in *"./Parameters/step_parameters"* and run *"ILC_initialization.m"* again.

7. (Optional step) Run the Simulink scheme *"./Simulink/Trajectory_tracking.slx"* to test the first control action estimate.

8. Run the function *"ILC_algorithm.m"*. Type *"ILC_algorithm(num_iterations, reset_flag)"* in the matlab command window, where *"num_iterations"* is the number of iterations you want to execute, while *"reset_flag"* is

equal to 0 if you want to continue the learning phase or start a new one. Optionally, you can use the command *"ILC_algorithm(num_iterations, reset_flag, error_threshold)"* where *"error_threshold"* is an optional argument with default value = 0; it defines a threshold under which the algorithm stops. The current iteration number is printed on the command window at each iterations.

9. If the robot does not return to the resting position after each iteration you should place it there yourself.

10. All data are stored in *"./Images/year_month_day_hour_minute/"*.

References

[1] Cosimo Della Santina, Matteo Bianchi, Giorgio Grioli, Franco Angelini, Manuel G Catalano, Manolo Garabini, and Antonio Bicchi. Feedback and feedforward control for soft robots. *IEEE Robotics & Automation Magazine (Accepted)*.

[2] Franco Angelini, Cosimo Della Santina, Manolo Garabini, Matteo Bianchi, Gian Maria Gasparri, Giorgio Grioli, Manuel G Catalano, and Antonio Bicchi. Decentralized trajectory tracking control for soft robots interacting with the environment. *IEEE Transactions on Robotics (Submitted)*.

[3] qbrobotics. `http://www.qbrobotics.com`.