Control Problems in Robotics
Course Project 2019/2020
Optimization-Based Iterative Learning Control for Trajectory Tracking

Emanuele Caruso, Syed Saad Saif, Alessandro Santopaolo

22 July 2020

SAPIENZA
UNIVERSITÀ DI ROMA

This report is submitted in fulfilment of the requirements for the course "Control Problems in Robotics" by Prof. Lanari and Oriolo, Sapienza University of Rome

1 Introduction

Robotics operations require sophisticated and intelligent control methods that can guarantee high performance under the presence of unmodeled dynamics and disturbances. Very often the task required is repetitive in time. This allows for the use of an iterative learning approach in the tuning of the controller. The experience gained from past trials can be used to learn more about un-modelled dynamics and parametric uncertainties, the feed-forward signal can then be modified accordingly to achieve better performance. This approach is called ILC-iterative learning control technique. The approach we have used for our work is taken by [1] and consists of two steps as we can see in 5. The novelty given by this approach lies in the combination of optimal filtering method with convex optimization techniques.

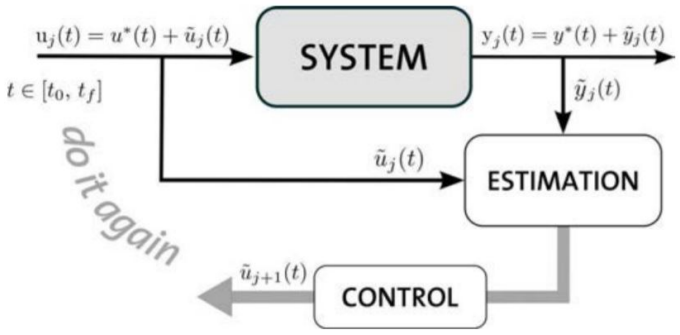

Figure 1: General ILC framework

The structure of the algorithm is the following. A dynamical model of the system is derived and a desired optimal reference trajectory is computed based on the model. The system is linearized around this trajectory and we compute the lifted domain representation of the linearized system. An initial trial is performed by actuating the desired reference input trajectory to the real system. Based on the output error with respect to the ideal one, we estimate the modelling error though the use of the Kalman filter. With this updated error information the control signal for the next iteration is adapted to better follow the reference trajectory, this is done by solving a constrained optimization problem. In this report we apply this strategy to swing up an inverted pendulum in a very small numbers of trials.

2 Experiment Setup - System, Model, Constraints

The goal of our project is the application of the ILC approach to swing up an inverted pendulum arranged on a cart. We will show that applying a purely open-loop input, the ILC algorithm learns how to do the swinging operation in few trials. Below we have a schematic drawing of the system:

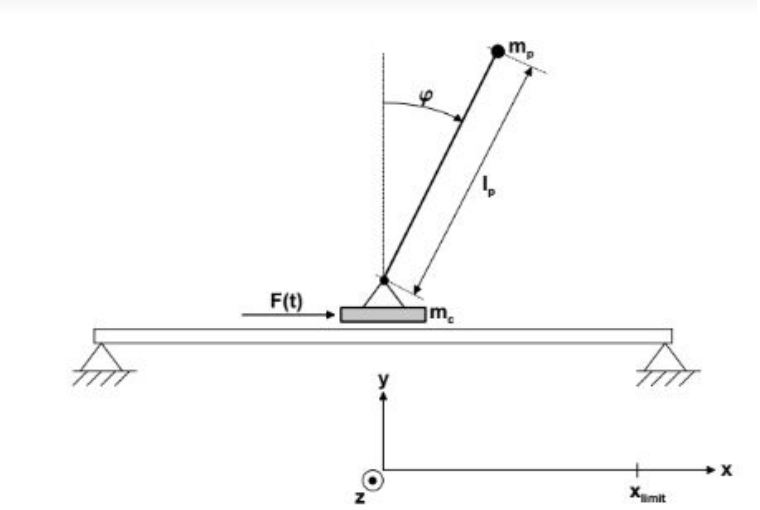

Figure 2: Scheme of the cart-pendulum system

The equations of motion of pendulum and cart used to describe the physical system are:

$$\ddot{x} = \frac{\frac{F}{m_p} - g\sin\varphi\cos\varphi + l_p\,\dot{\varphi}^2\sin\varphi}{\frac{m_c}{m_p} + \sin^2\varphi}$$

$$\ddot{\varphi} = \frac{-\cos\varphi\frac{F}{m_p} + \left(\frac{m_c+m_p}{m_p}g - l_p\,\dot{\varphi}^2\cos\varphi\right)\sin\varphi}{l_p\left(\frac{m_c}{m_p} + \sin^2\varphi\right)}$$

Where x represents the position of the cart, φ represents pendulum angle measured from the downward position and F is the force applied to the cart. The force is produced by a DC motor belt drive and can be modeled by :

$$F = \alpha_1 u + \alpha_2 \dot{x} \tag{1}$$

For the next steps we reformulated them as a set of first order differential equations with $x(t) = [x(t), \dot{x(t)}, \varphi(t), \dot{\varphi(t)}]$. Where u is the input signal

which is proportional to the voltage supplied to the motor.
In order to simulate and test the validity of the ILC algorithm we consider two different systems, the nominal and the real one.
The values of the nominal parameters are taken from [1] and are fully described in this table:

m_p	mass of the pendulum	175 g
m_c	mass of the cart	1.5 kg
l_p	distance from pivot to pendulum's center of mass	28 cm
α_1	motor constant 1 (voltage-to-force)	159 N
α_2	motor constant 2 (electrical resistance-to-force)	-22.5 Ns/m
g	gravitational constant	9.81 m/s^2

To simulate the behaviour of the real robot we have considered a model with slightly different parameter values with respect to the nominal one.

This system is perfectly suited for the application of the ILC algorithm. We pursue a repetitive task, the swinging up of the pendulum, under strong constraints of the input and the state due to the rail length and limitations of the cart actuator:

$$|x| \leq 0.5m$$

$$|\dot{x}| \leq 5m/s$$

$$|u| \leq 0.45m$$

In the next section we describe in detail the ILC algorithm.

3 Mathematical background

The Iterative Learning Algorithm is a powerful method for high performance reference tracking. The approach adopted by [1] is based on the combination of two steps, estimation and control. In the first step a time-varying Kalman-filter estimates the model error along the trajectory. In the second step, the control problem is formulated as a convex optimization problem. The combination of optimal filtering and convex optimization makes this method highly effective.

3.1 Dynamics and Lifted Representation

The starting point of the algorithm is a non linear time-varying model that in our case is represented by the the equations of motion of the pendulum and the cart. For further steps we reformulate the equations as first order differential equations with $x(t) = [x(t), \dot{x(t)}, \varphi(t), \dot{\varphi(t)}]$. So the dimension of the state vector is four, $x(t) \in R^{n_x}$ with $n_x = 4$. The dimension of the input vector is 1, $u(t) \in R^{n_u}$ with $n_u = 1$. Moreover we set $y(t) = x(t)$. We also set the constraints on the input and the state. In our case:

$$|x| \leq 0.5m$$

$$|\dot{x}| \leq 5m/s$$

$$|u| \leq 0.45m$$

The goal of the algorithm is to track an a pre-determined output trajectory y*(t) over a finite time interval that we have chosen as 1.86s. The desired trajectory is assumed to be feasible with respect to the model and the given constraints. The optimal trajectory is obtained by running NLMPC on the non-linear model. Furthermore, we suppose that the motion of the system stays close to the generated reference trajectory during the learning process. This hypothesis allows us to approximate the system's behaviour by a first-order Taylor expansions about the reference trajectory resulting in a linear, time-varying system.

$$\dot{\tilde{x}}(t) = A(t)\tilde{x}(t) + B(t)\tilde{u}(t)$$
$$\tilde{y}(t) = C(t)\tilde{x}(t) + D(t)\tilde{u}(t)$$

where $t \in T = [t_0, t_f]$ with t_f finite.
Notice that all the terms expressed with tilde in the linearized system, refer to the deviations with respect to the nominal trajectory. Since we're dealing with digital system we have states available only at precise instant of time

so we have introduced a discrete-time representation of the obtained linear system.

$$\dot{\tilde{x}}(k+1) = A_D(k)\tilde{x}(k) + B_D(k)\tilde{u}(k)$$
$$\tilde{y}(k) = C_D(k)\tilde{x}(k) + D_D(k)\tilde{u}(k)$$

where $k \in K = 1, ..., N$. The desired trajectory is represented by N sample sequence. The next step in obtain a lifted representation of the system. At each trial the deviations of input, state and output are stored in a lifted vector:

$$u = [\,\tilde{u}(0),\, \tilde{u}(1),\, \ldots,\, \tilde{u}(N)\,]^T \;\in \mathbb{R}^{(N+1)n_u}$$
$$x = [\,\tilde{x}(0),\, \tilde{x}(1),\, \ldots,\, \tilde{x}(N)\,]^T \;\in \mathbb{R}^{(N+1)n_x}$$
$$y = [\,\tilde{y}(0),\, \tilde{y}(1),\, \ldots,\, \tilde{y}(N)\,]^T \;\in \mathbb{R}^{(N+1)n_y}$$

In dynamics relation of the system is captured by a static map:

$$x = Fu + d^0$$
$$y = Gx + Hu$$

The lifted matrix $F \in R^{(N+1)n_x \times (N+1)n_u}$ has this structure:

$$F = \begin{bmatrix} F_{(0,0)} & \ldots & F_{(0,N)} \\ \ldots & \ldots & \ldots \\ F_{(0,N)} & \ldots & F_{(N,N)} \end{bmatrix}$$

where:

$$F_{(l,m)} = \begin{cases} A_D(l-1)\ \ldots\ A_D(m+1)\ B_D(m) & \text{if } m < l-1 \\ B_D(m) & \text{if } m = l-1 \\ 0 & \text{if } m > l-1 \end{cases}$$

and matrices G and H are block-diagonal and analogously defined by

$$G = \begin{cases} C_D(l) & \text{if } l = m \\ 0 & otherwise \end{cases}$$

$$H = \begin{cases} \mathrm{D}_D(l) & \text{if } l = m \\ 0 & otherwise \end{cases}$$

This structure is well suited for the ILC algorithm because we have stored the entire dynamics of a single trial is represented as matrices and the states can be fond algebraically. The goal of ILC is to update the feed-forward signal u at each trial based on the data gathered during the last execution. Trial after trial, we build the dynamics of the learning that we specify with a subscript j that indicate the j-th task.
We can now discuss the insertion of noise and disturbances inside the system description that explains the background behind the estimation and control steps.

$$\begin{aligned} x_j = &\quad Fu_j + d_j &+ N_\xi \xi_j \\ y_j = &\quad Gx_j + Hu_j &+ N_\upsilon \upsilon_j \end{aligned}$$

The process disturbance is specified by ξ while υ is considered as sensor noise. Both random variables are considered trial-uncorrelated of zero-mean Gaussian white noise with covariance Ξ_j and Υ_j. The vector d_j represents the model error along the reference trajectory which slightly changes during each iteration. The nonzero initial conditions are captured also in d_j. The iteration dynamics of d_j is described as follow:

$$d_j = d_{j-1} + \omega_{j-1}$$

Where ω_j is another trial-uncorrelated sequence of zero-mean Gaussian white noise with covariance Ω_j. Having specified those sources of error, we can summarize the characteristics of the iteration-domain system dynamics. The state deviation along the desired trajectory is affected by two different noise sources, a trial uncorrelated zero-mean component Ξ_j and a 'random walk' component d_j. In real setting we can consider d_j as a vector representation of all unmodeled dynamics along the trajectory and the dynamic model can be simplified where d_j absorbs removed dynamics. The ultimate goal of ILC algorithm is to estimate and optimally compensate for the error d_j by updating the input trajectory appropriately. Assuming a convergent input $(u^* + u_j)$ for an increasing number j of trials, the sequence d_j converges too. Hence, we can define $\Omega_j = \epsilon_j I$ with $\epsilon_j < \epsilon_{j-1}$. With this definition the covariance supports a fast converge of the learning algorithm.

3.2 Estimation

The first step of the learning algorithm, provides an estimation of the modeling error d_j using optimal filtering techniques. We can represent the evolution of the output trajectories through iterations with this dynamic system:

$$\begin{cases} d_j = d_{j-1} + \omega_{j-1} \\ y_j = G\ d_j + (GF + H)u_j + \mu_j \end{cases}$$

where

$$\mu_j = \begin{bmatrix} GN_\xi & N_\upsilon \end{bmatrix} \begin{bmatrix} \xi_j \\ \upsilon_j \end{bmatrix}$$

With the previous noise definitions, the stochastic variable μ_j, is characterized by $\mu_j \sim N(0, M_j)$

where,

$$M_j = \begin{bmatrix} GN_\xi & N_\upsilon \end{bmatrix} \begin{bmatrix} \Xi_j & \Delta_j \\ \Delta_j^T & \Upsilon_j \end{bmatrix} \begin{bmatrix} (GN_\xi)^T \\ N_\upsilon^T \end{bmatrix}$$

Since the modeling error d_j is not fully observable, we can estimate $\hat{d}_j$ by using the standard Kalman filter approach, by looking at the obsevations of y_j and so minimizing the error variance :

$$P_{j|j} = E[(d_j - \hat{d}_{j|j})(d_j - \hat{d}_{j|j})^T]$$

Here are the Kalman filter equations for this specific problem:

$$\begin{cases} P_{0|0} = P_0 \\ P_{j|j-1} = P_{j-1|j-1} + \Omega_{j-1} \\ \Theta_j = G\ P_{j|j-1}\ G^T + M_j \\ K_j = P_{j|j-1}\ G^T\ \Theta_j^{-1} \\ P_{j|j} = (I - K_j G) P_{j|j-1} \end{cases}$$

Finally, with the optimal Kalman gain K_j, we compte the estimation of $\hat{d}_j$.

$$\begin{cases} \hat{d}_{0|0} = \hat{d}_0 \\ \hat{d}_{j|j} = \hat{d}_{j-1|j-1} + K_j(y_j - G\hat{d}_{j-1|j-1} - (GF + H)u_j) \end{cases}$$

where y_j is the measurement of the deviation of the output trajectory at iteration j from the nominal trajectory, and is given by the robot sensors. Usually the first guess of the modeling error $\hat{d}_0$ is set to a vector of 0, and the value of P_0 is chosen as a diagonal matrix with enough large positive elements.

3.3 Control

The algorithm is completed with the use of a convex optimization technique. Using the information gained from the estimated modeling error $\hat{d}_{j|j}$, the goal is to find the new input u_{j+1} that better tracks the desired trajectory. In our context this means minimize:

$$x_{j+1} \approx Fu_{j+1} + \hat{d}_{j|j}$$

over all feasible u_{j+1}, $u_{min} - u^* < u_{j+1} < u_{max} - u^*$. Taking in account the convergence of d_j we can approximate the error d_{j+1} by $\hat{d}_{j|j}$. The problem can be then expressed as the minimization of the euclidean norm below:

$$\min_{u_{j+1}} \left(Fu_{j+1} + \widehat{d}_{j\,|\,j}\right)^T \left(Fu_{j+1} + \widehat{d}_{j\,|\,j}\right)$$

4 Implementation details

In the project we have used Matlab software to simulate and test the efficacy of the ILC algorithm applied to the cart pendulum system. First, we created a function expressing the continuous time cart-pendulum system. The parameters values for the nominal model are the one chosen by [1], conversely, to simulate the real robot we have chosen slightly different parameters. A sampling time of 0.01s was chosen and the time horizon of 1 second was selected over which the performance of the feed-forward signal is considered. The first step of the algorithm was to compute the nominal trajectory. To do that we have applied a nonlinear Model predictive Control technique. A nonlinear model predictive controller computes optimal control signal across the prediction horizon using a nonlinear prediction model and a nonlinear cost function in addition the constraints are also taken into account. We have imposed a prediction horizon of 100 and a control Horizon of 5. Furthermore, constraints have been added to the states and input due to the limited rail length and given by the actuator constraints. In the picture below we can see the results obtained:

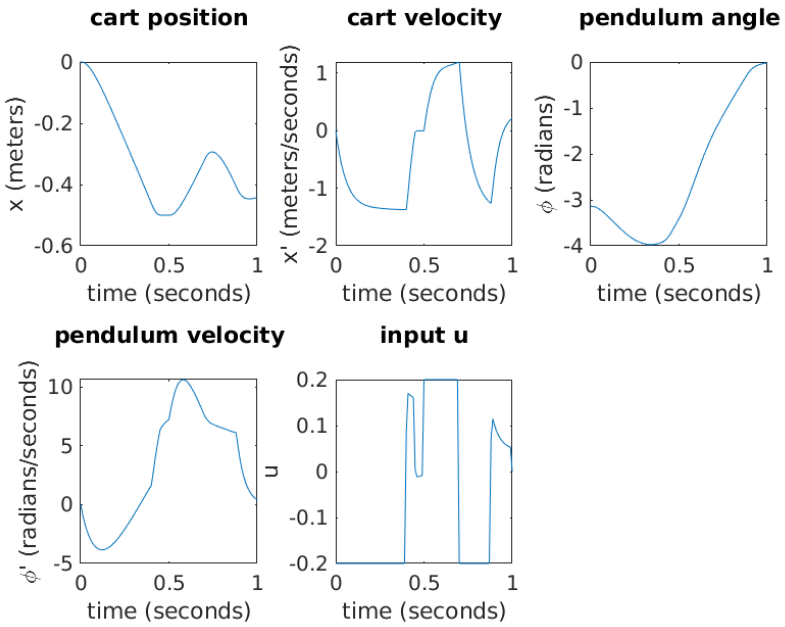

Figure 3: From left to right plots of cart position, cart velocity, pendulum angle, pendulum velocity and input with respect to time.

The MATLAB function used to perform this computation is named **get**

optimal trajectory. At this stage we start to apply the ILC algorithm.
First of all we have linearized the system dynamics around the nominal trajectory found a time varying linear system of the following form.

$$\begin{cases} \dot{\tilde{x}}(t) = A(t)\tilde{x}(t) + B(t)\tilde{u}(t) \\ \tilde{y}(t) = C(t)\tilde{x}(t) \end{cases}$$

where A,B matrices are the jacobians of the non-linear function of the cart-pendulum system $\dot{x}(t) = f(x(t), u(t), t)$ with respect to x and u, evaluated on the nominal trajectory. Those matrices are computed by using the MATLAB function **get Jacobians** inside the code. C is the identity matrix since we want $y(t) = x(t)$.
Then we have discretized the system by using Euler discretization [2] :

$$\begin{cases} \dot{\tilde{x}}(k+1) = A_d(k)\tilde{x}(k) + B_d(k)\tilde{u}(k) \\ \tilde{y}(k) = C_d(k)\tilde{x}(k) \end{cases}$$

where

$$\begin{cases} \mathrm{A}_d = (I + TA) \\ \mathrm{B}_d = TB \\ \mathrm{C}_d = C = I \end{cases}$$

with T = sampling time = 0.01.

After that, using the MATLAB function **get lifted** which exploits the algorithm proposed by Prof. D'andrea we have obtained the lifted domain representation. In this way we have computed matrix F and G that we will use in the estimation procedure. This two matrix remain the same during each trial.
Then, the error given the disturbances ξ, the noise of the sensor υ and the "random walk error" ω for the initial condition, are each simulated inside the real model by sampling from a random value taken from a zero-mean normal distribution with diagonal covariance matrix with all elements equal to 0.001.
Then the covariances assumed inside the estimation step are the same of the ones used in the paper:

- P_0 diagonal with entries 2
- M_j diagonal with entries 0.1

- Ω_j diagonal with entries 0.3

So, once the estimation step is done, the control step provides a solution for the optimization problem expressed in chapter 3.3. Instead of using CPLEX, the MATLAB tool fmincon has been exploited to find the input vector. Differently from the paper, the error to be minimize is depends only on the pendulum angle, while in the paper also the error on the cart position is included. In this way, the solution of the minimization problem is totally focused to swing up the pendulum which is the real goal of the project. The constraints given to fmincon refer both to $u, x, \dot{x}$. While for u we have only to specify the lower and the upper bounds, for $x, \dot{x}$ we have chosen to simulate the trajectory by considering the nominal model embedding the current estimation of d for the unmodeled dynamics, and then imposing the limits on x and $\dot{x}$. This constraints are specified inside the MATLAB function **nonlinearconstraints**. Notice that neither scaling nor weighting has been used for the control step. Now let's show the results obtained:

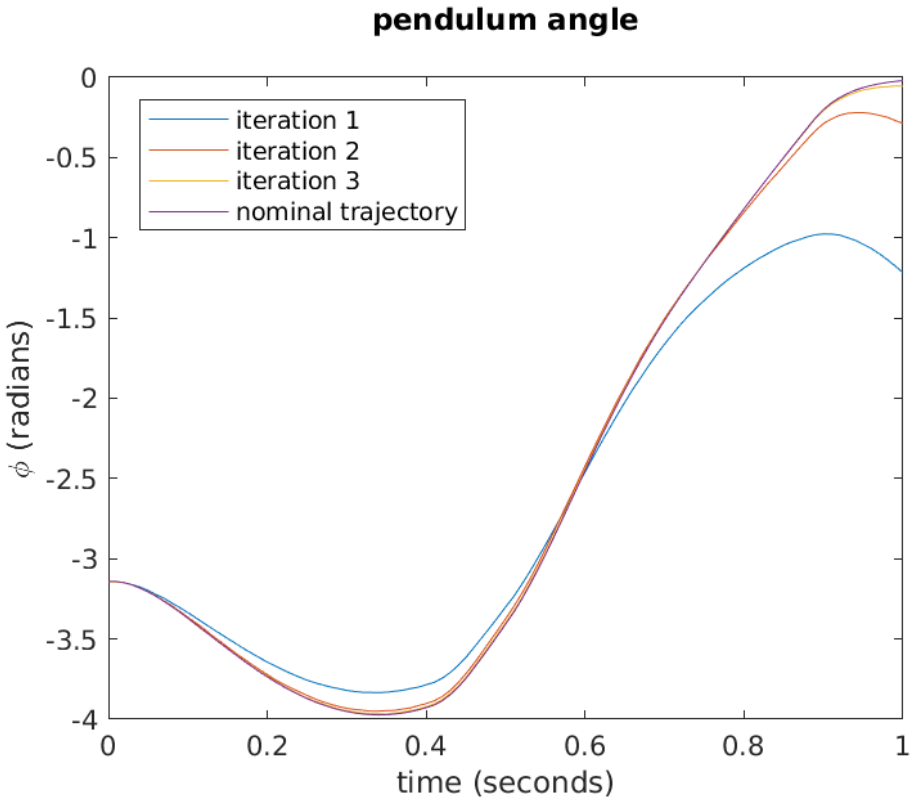

Figure 4: Pendulum angle trajectory for each iteration.

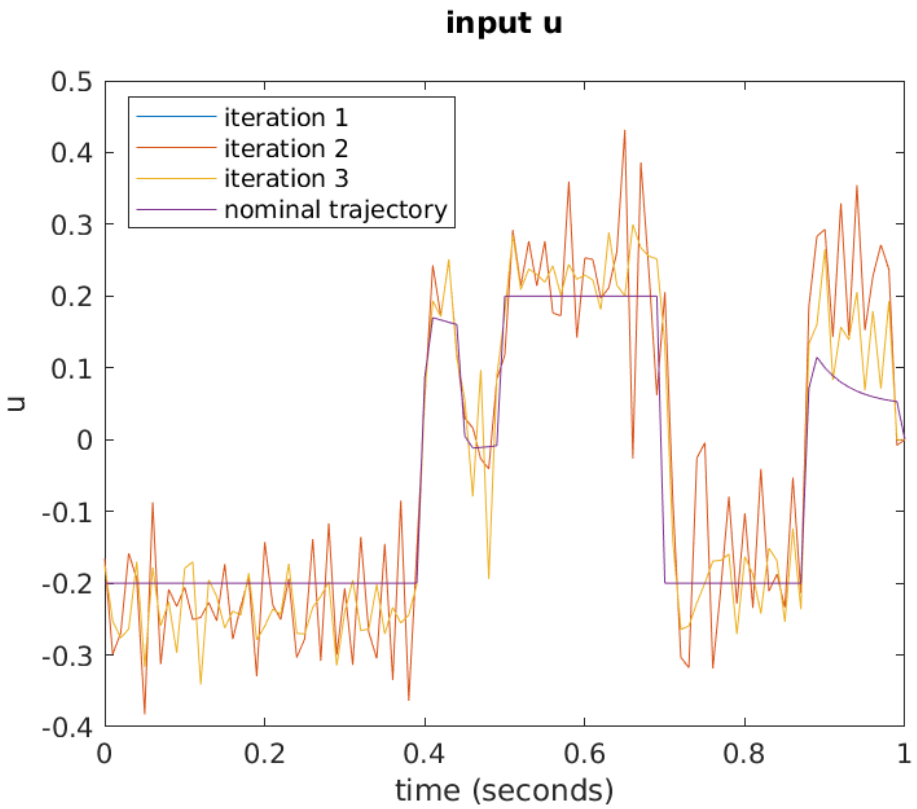

Figure 5: Input trajectory for each iteration.

As can be seen by the images, after 3 iterations, the trajectory of the pendulum angle is very close to the original one. Notice that for the figure which shows the input, the trajectory of the first iteration coincide with the nominal input.

5 Conclusion

In this project we have exploited an optimization-based ILC approach swinging up an inverted pendulum cart-based. We have analyzed and explained in detail the algorithm and then the implementation details of our application of it. The optimality of the method is achieved in both the estimation of the modeling error and the following control step, which optimally compensates for the error by an updated input trajectory. This approach can be very useful in situation in which there is a repetitive motion, that is usual in industrial scenarios. According on the particular problem, the learning procedure can be changed changing the optimization objective tuning the weights for different states.

References

[1] Angela Schöllig and Raffaello D'Andrea. Optimization-based iterative learning control for trajectory tracking. In *2009 European Control Conference (ECC)*, pages 1505–1510, 2009.

[2] Zoran Gajic. Linear dynamic systems and signals. pages 407–410, 2003.