Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)

Constrained numerical optimization for estimation and control

Offline estimation of the human force acting on the end-effector of a collaborative robot and online optimal control of the torque

Luca Brolese, Andrea Ghezzi, Rodrigo Senofieni

July 18, 2019

Contents

1 Introduction to the problem

Nowadays we are living an important revolution in the industry driven by new technologies such as augmented reality, internet of things, big data and advanced manufacturing solutions. We point our attention to collaborative robotics and one of the most common and basic tasks in industry: loads transportation and manipulation. In a lot of activities human workers have to move a load between two points and in many cases the load is too heavy for a human to transport. Therefore, we had the idea to use a robot to aid the operator without pushing the human "out of the loop". In fact, the human intelligence is fundamental for the position task (trajectory generation and control) and for the dynamic sensing of the environment. Thus, the solution we imagine is to let the human and the robot work together to achieve the positioning goal. The main problem we have faced was the measure of the force applied by the human on the robot, because we assumed to have a force-sensor-less robot. The main reasons of this choice are firstly that mounting force sensor on the end-effector is not feasible because the human interacts directly with the load (and not with the end-effector), secondly force sensors are expensive.

It is possible to divide our work in two main activities:

1. The developing of a filtering algorithm to obtain a virtual force sensor for the robot;
2. We imagine a possible application: an optimal control problem that computes the robot joints torques to achieve the positioning goal, treating the human action as a disturbance acting on the system.

Finally, we would like to point out the baseline behind our modelling choices. For sure, the two main blocks are the human and the robot and both are modelled from us in the simplest way as possible (as we will see in the section 3), because we want to focus our attention on the optimization part of the project rather than on detailed structure of the two systems.

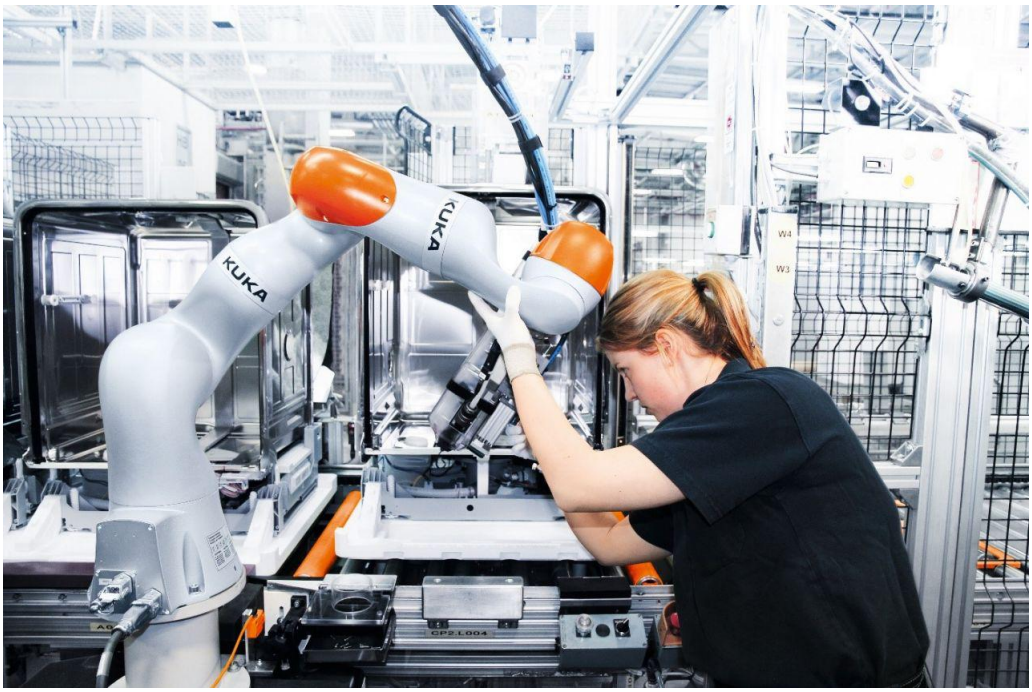

Figure 1: Example of collaborative robotics

2 Problem modelling

Aim of this section is to give a more detailed description of the model that has been used for the simulation and data acquisition during the optimization routine. As already said, the objective is to obtain a good estimation of the human force along XZ direction. Thus, in the modelling phase three important aspect were considered: 1. The model of the human is relatively simple, but more than enough for our optimization task; 2. The model of the robot is a 2 DOF manipulator along XZ axis; 3. The load has been considered as a solid mass attached to the end effector of the robot.

2.1 Model of the human

Figure 2: Human block

The modelling phase of the human behaviour is a critical point. As well known, in optimization and control tasks the first objective is always to have a model that fully describe the dynamics of our system and simultaneously is as easy as possible, trying to keep the complexity low in order to have good performances during the optimization routine. If we consider the human body, it can be modelled as a very complicated mathematical system, involving deep relationship between its apparatuses (neurological - intention to perform an action -, nervous - transmission of information - and muscle - fulfillment of the action). This approach would lead us inevitably to an overcomplicated system, which is something undesired for the aim of the task. A possible – and simple – approach is to try to extract the most important "feature" of the human body:

1. Capability to "predict an action": can be modelled as a zero with a specific frequency;
2. Time delay due to the neurological and nervous transmission of the signals: can be implemented with a pole, which can be placed depending on the human frequency bandwidth (explained below);
3. Ability of amplify an action (like a force): implemented as the gain of the transfer function.

The problem now is to understand how the frequency of the poles and zeros should be selected. Lots of interesting studies as been conducted on understanding the human behaviour -and its mathematical model- while performing some tasks. A very interesting study was as shown in [1] that the human can be modelled via a transfer function like:

$$F_{(s)} = \frac{Y_{(s)}}{X_{(s)}} = K\frac{T_3 s + 1}{(T_1 s + 1)(T_2 s + 1)}e^{-\tau s}$$

Where the coefficients are:

- T_1: Lag Time Constant – describes the time delay necessary for giving the correct input (in the problem case the force);
- T_2: Neuromuscular Lag Time Constant – represents the time constant associated with contraction of the muscles through which the control input is applied by the human;
- T_3: Lead Time Constant – reflecting the human's ability to predict a control input;

- K: Human Gain – representing the human's ability to respond to an error in the magnitude of a controlled variable;
- τ: Represents a pure time delay describing the period between the decision to change a control input and the change starting to occur.

In the formulation of our problem, always having in mind to keep the formulation as simple as possible, we neglected the pole associated to the neuromuscular lag time constant and the pure time delay τ. Thus, the final model is a first order transfer function with 1 pole, 1 zero and a gain K. The pole of the transfer function is set considering the human response time, which is attestable to be around 8 to 10 Hz. The other 2 parameter (K and T_3) have been set with some trial and error procedure, in order to obtain a satisfactory and stable open loop response of the system. We used the following transfer function:

$$H(s) = 100\frac{1+s}{1+0.1s}$$

2.2 Model of the robot

First, we have designed a possible trajectory that the full system, human-robot-load, must follow during its movement. We decided, in order to have the easiest trajectory to handle during the optimization routine, to design our path only along XZ direction. The trajectory can be divided in two part:

1. Starting from the origin, the trajectory end in (x=1, z=1) and reach this position in 4 seconds;
2. Starting in (x=1, z=1) we keep it constant for the next 6 seconds.

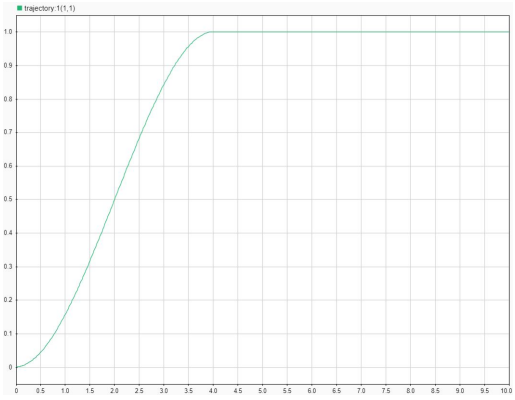

Figure 3: Trajectory

We generated the trajectory with a simple 3rd order polynomial function time dependant (time dependence is set by the ramp in input of the MATLAB function). As output the MATLAB function computes the trajectory and the speed reference that are fundamental for the robot kinematics. After the trajectory planning, we could focus on the core of the second modelling phase, the one concerning the robot. We decided to implement a classical 2 DOF manipulator, in particular considering the load directly attached to the extremity of the second arm.

As we can see in the picture the manipulator is composed by two connected joints with their respective masses and moments of inertia. The joint fixed on the floor is positioned 0.5 meters on the right of the operator. For the sake of simplicity, we have assumed that: the robot is frictionless, the joints are rigids and not deformable. To obtain the dynamical equations of the robot we have followed the "classical" approach solving the Lagrange equations starting from the kinetic and potential energy

θ_i i-th joint variable;

m_i i-th link mass ;

$\tilde{I}_i$ i-th link inertia, about an axis through the CoM and parallel to **z**;

a_i i-th link length;

a_{Ci} distance between joint i and the CoM of the i-th link;

τ_i torque on joint i;

g gravity force along **y**;

P_i, K_i potential and kinetic energy of the i-th link.

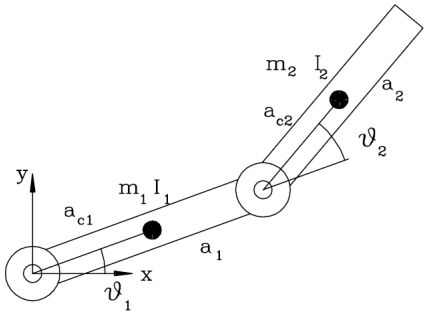

Figure 4: 2 DOF manipulator with its parameters

(neglecting dissipative terms).
[refer to the MATLAB function *robotdynamics* for the full model computation]

Moreover, to track our trajectory that is positioned in the XZ plane, this type of manipulator with the selected degrees of freedom: $q_1 = \theta_1$ and $q_2 = \theta_2$, and the respective length of the two joints satisfy perfectly our aim. After the definition of the models and the trajectory we can compute the joints angular positions and the respectively speeds through the inverse kinematics block.
[refer to the Simulink block $Modelsim/RobotControl/directkin$]

3 Problem abstraction and formulation of the optimization program

Before starting with the optimization routine, some preliminaries about the *Simulink* model and data acquisition has been performed.

3.1 Estimation phase

For what concern the *Simulink* models, the key point is to consider the human force as a disturbance acting on the end effector of the robot. The disturbance is actually giving an active contribute for moving the robot and the load, considering the fact that during the estimation phase the robot is assumed to be passive (no torque is produced). The human tries to follow the reference trajectory, while trying to move the load plus the robot. Of course, the human is not capable of bringing to reference the system, because the load is too heavy.
After we have set up all the model and the *Simulink* control schema, firstly we collect the reference data running the script $Modeldaq$; secondly we generate the reference data q_{ref} and qd_{ref} (respectively the angular positions and speed of the joints of the robot). These data will be used later in the computation of the cost function for the optimization routine. Furthermore, also the F_{human} reference has been stored for the final check after the optimization routine has estimated the human force.

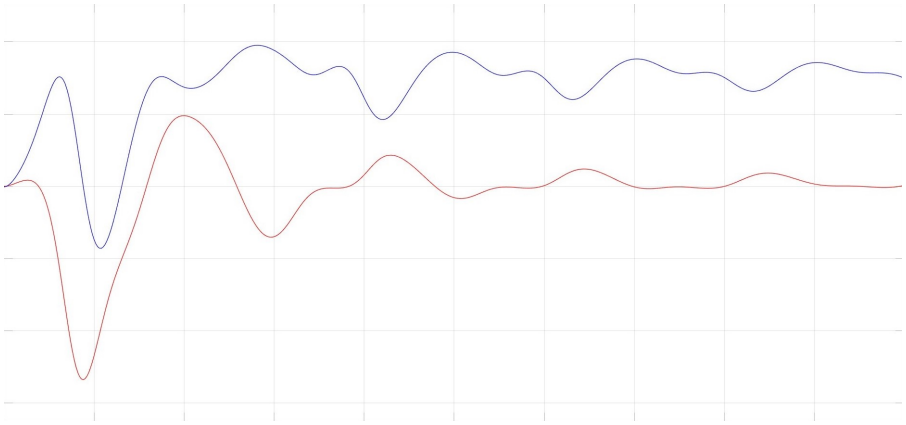

Figure 5: Reference human forces (red: Fh_x, blue: Fh_z)

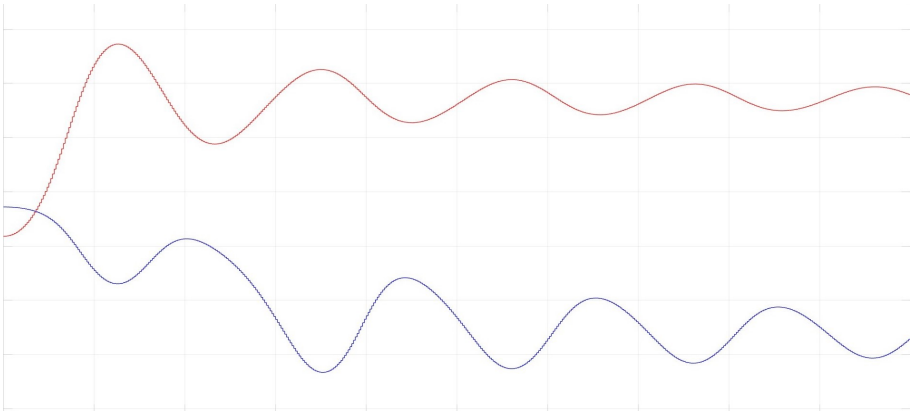

Figure 6: Reference angular positions (red: $x - axis$, blue: $z - axis$)

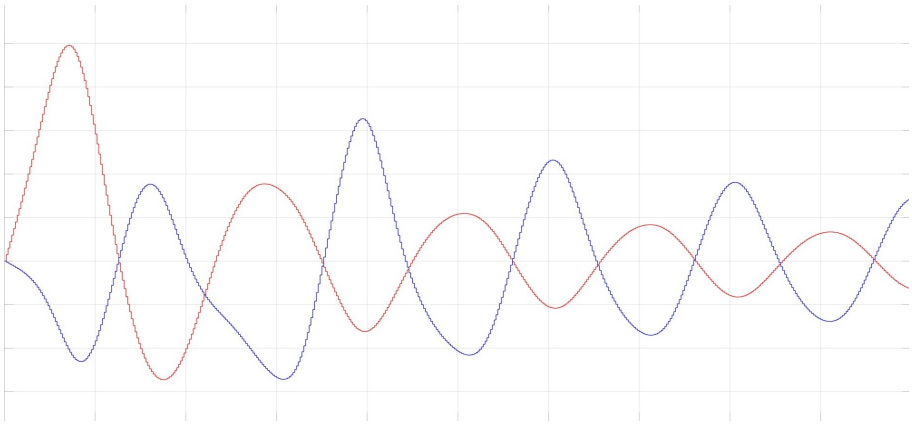

Figure 7: Reference angular speeds (red: $x-axis$, blue: $z-axis$)

Now, the task is to set up the cost function needed by the solver for the estimation phase. Since, from the point of view of the robot, the trajectory is unknown, the only measurable quantities we can work on are the measured angular positions q and speeds q_d of the joints. The cost function is evaluated considering the error between:

- The reference q_{ref} and qd_{ref}, taken from the data acquisition procedure;
- The simulated q_{sim} and qd_{sim}, computed using the estimated human forces that comes from the optimization routine;

At the beginning, the 2 forces are set to an initial guess (user defined e.g. 100 N), then the optimization routine will try to minimize the cost function by means of the 2 optimization variables (Fh_x and Fh_z).
The approach that we used for estimating the human force is based on a Moving Horizon Estimation, with a sliding window of dimension M (in our problem M=5). The algorithm, at time t go through the following steps:

1. Start the optimization routine at time i+M (i from 1 to the number of collected samples);
2. Collect the reference data (q_{ref}, qd_{ref}) corresponding to the current time window we are considering;
3. Update the initial conditions of the integrators in *Simulink* with the last positions of the joints at the previous optimization cycle;
4. Call the optimization function that oversees the research, at the current time window, of the optimal local minimizer Fh_x and Fh_z;
5. Store the optimal value of the current Fh guess;
6. Store the simulated trajectory (for final check between reference and simulated) computed using the *direct kin* function from the values q_{sim} and qd_{sim};
7. Repeat from point 2, until the cycle arrives to the end of the iterations;

3.2 Optimal control phase

This second part of the project consists in a more useful and "practical" application. The idea is that the robot should help the human to move the load. Therefore, it is now mandatory to consider the robot as an active element, that helps the worker during his task. Initially, we made some changes on the problem formulation. First of all, we changed the approach to a one more suitable for FHOCP. The main idea is to have a sliding window M steps ahead, from which we can get an estimation of the optimal torque to apply to the system, in order to help the human moving the load. From now on, we will not focus on the estimation of the human force, but rather on the estimation of the optimal control torque to apply to the system in order to help the human. The big difference is on the simulation of the model: now the optimization variables are the 2 torques (different from zeros), while the human forces are considered as disturbances acting on the robot dynamics. The human transfer function is working on the error, evaluated as the difference between the reference trajectory (the one designed in 2.1) and the simulated one (coming from the direct kinematic transformation of q_{sim} and qd_{sim}). The idea is that the optimization algorithm (which can be seen as an MPC with control horizon equal to 1, known as *mean level control*, see [2]) will try to find the optimal values of the torques such that the reference trajectory is followed. The human closes the trajectory loop and, as soon as the error gets smaller (basically when the robot is giving the right torque), the human force decreases to reasonable values.
So, we achieve our goal because the largest amount of force comes from the robot and the human only oversees the trajectory control. As we can see from the control schema, the rule played by the

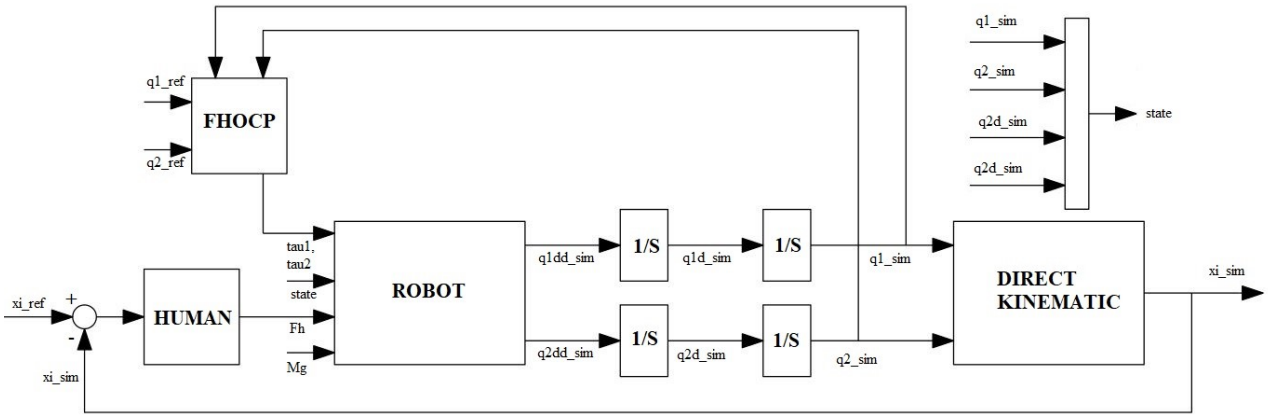

Figure 8: General control schema of FHOCP approach

optimization algorithm is to close the loop on the robot torque. In fact, from the point of view of the robot, the desired trajectory is unknown; the only dimensions we can sense are the joints angular positions and speeds. As a matter of fact, the optimization algorithm is still working on a cost function evaluating the error between the q_{ref}, qd_{ref} and q_{sim}, qd_{sim} (this time q_{ref} and qd_{ref} are the one that actually follow the desired trajectory designed in section 2.2), while the human loop is working on the error between the reference trajectory (the one he wants to follow) and the real trajectory (coming from the direct kinematic of q_{sim} and qd_{sim}). By doing so, as soon as the robot is giving the correct torque (notice that the human and the desired trajectory are still unknown entities from the point of view of the robot) the error, on which the human closed loop is working, is getting smaller and smaller, meaning that his force is decreasing. The results (see section 5) will show exactly this behaviour: as the robot gives more torque, the human is giving less force. Moreover, two saturation blocks have

been included, in order to limit the maximum force of the human (e.g. $\pm 100N$).

4 Numerical optimization solution

In this section we discuss the options for the optimization functions we used in the estimation problem (we have used the functions developed during the semester laboratories). Since this first task is a filtering/estimation problem of the human force, the best approach which grants optimal performance is $Gauss - Newton$ methods (mentioned from now on as "GN") for the computation, at each step k, of the descent direction p_k. As a matter of fact, in order to use this method, we express our cost function as:

$$f(x) = F(x)^T F(x)$$

where $F(x)$ is a long vector of stacked errors (concerning angular positions and speeds). Therefore, the descent direction p_k at step k is computed as:

$$p^k = -(H^k)^{-1} \nabla_x f(x^k)$$

where H^k, according to GN method is evaluated as:

$$H^k = 2\nabla_x F(x^k) \nabla_x F(x^k)^T$$

It is worth saying that, with GN approach, we have the advantage of H^k being always positive semidefinite and its computation requires only first derivatives of vector $F(x)$ (in our case vector of the stacked errors). Furthermore, GN guarantees linear convergence to a local minimizer when starting away from it, but quadratically when close to it.
Regarding the structure of the problem, we build an unconstrained program. This design choice is related to the main goal of the filtering phase: to have an estimation of the human force applied on the load and on the robot system. By adding constraints on the maximum force the human can give, it would have lead to incorrect results of the estimations. But, considering the fact that this algorithm can be used also for estimation of other disturbances (like other source of forces acting on the end effector), in the $MATLAB$ script we added the possibility of implementing equality and inequality constraints if required.
For the second phase of the project, things are slightly different. Being an optimal control problem, the best approach for the computation of the descent direction is still Gauss-Newton, for the same reasons stated above. Now, the optimization variables are the two optimal torques (tau_1 and tau_2) and the human is treated as an unknown entity acting on the system. The main changes were made on the settings of the optimization routine. First, we increase the tolerance on the minimum change of the optimization variable (from 10^{-13} to 10^{-8}) because being a torque produced by electric motors, such small variations does not produce relevant movements. Second, we try different values of tolerance on minimum change in the gradient and come to the conclusion that the best trade off between time/performances/results was to set it at 10^{-6}. Other settings were left unchanged.
In both methods, for the computation of the gradient, we used the central differences approach. Even if other methods are available for this aim (e.g. finite differences, automatic differentiation, etc.), central differences method is the one that gives the best results in terms of accuracy and precision, however implying higher computational cost.

5 Results and comments

Once, that we have defined the models and the optimization routines both for the filtering problem and for FHOCP problem, we obtain as outputs the estimation of the human force (Fh_x, Fh_z) and the optimal torques delivered by the joints (tau_1, tau_2) respectively. Now, we can proceed with the analysis and some considerations about the simulation results.

5.1 Estimation problem results

5.1.1 Force estimation

Our comparison is referred to a time horizon of 10 seconds where we can notice a different trend before and after the first 4 seconds. As already said, in the estimation phase, the optimization routine is working on the q_{ref} and qd_{ref}, coming from the robot dynamics. The reason of this choice is that, from a point of view of the robot, the $q1'_{ref}$ and $q2'_{ref}$ (coming from the direct kinematic of the desired trajectory of the human) are unknown. On the other hand, the robot can assume as reference signals the ones coming from the direct measurements of the joints. Here's why in this estimation phase, the obtained results do not match the desired trajectory (the one going from (0,0,0) to (1,0,1)), but rather the ones where the human is trying to move the load plus the robot, inevitably failing in his task. But, if we focus on the results obtained thanks to the optimization program, we can see from the Figure 9 a good estimation of the human forces. There is a clear offset along the time axis, mainly due to the MHE approach and the sampling time of the ZOHs in *Simulink*. Finally, there are some drifts probably due to integration problems in *Simulink*.

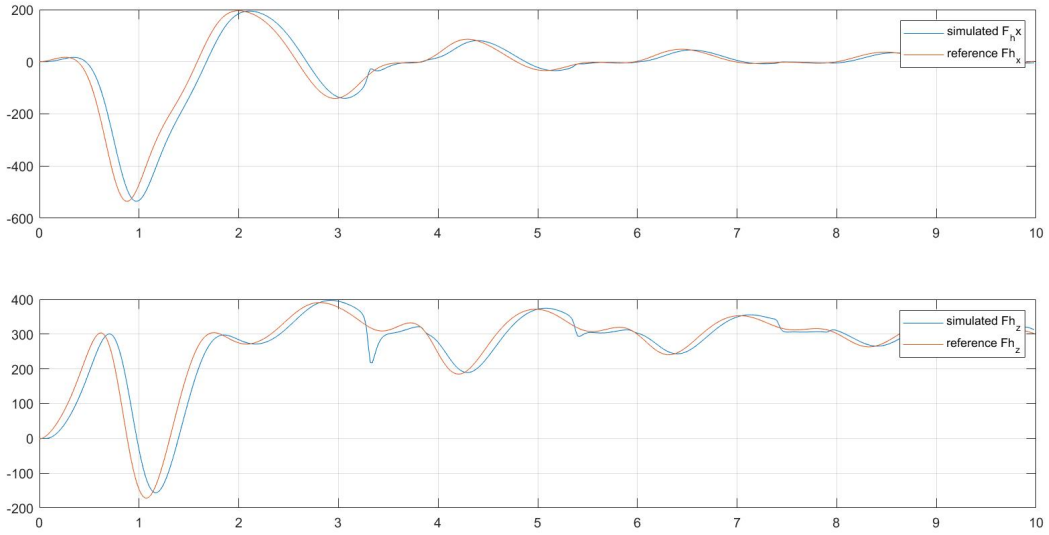

Figure 9: Estimated vs reference forces

5.1.2 Trajectory tracking

In the first row of Figure 10 we can see the plot of the reference trajectory vs. the simulated trajectory on X and Z axis. In the second row we can see the plot of the reference speed vs. the simulated speed. Notice that, as already said in the paragraph above, the reference trajectory is not the one desired,

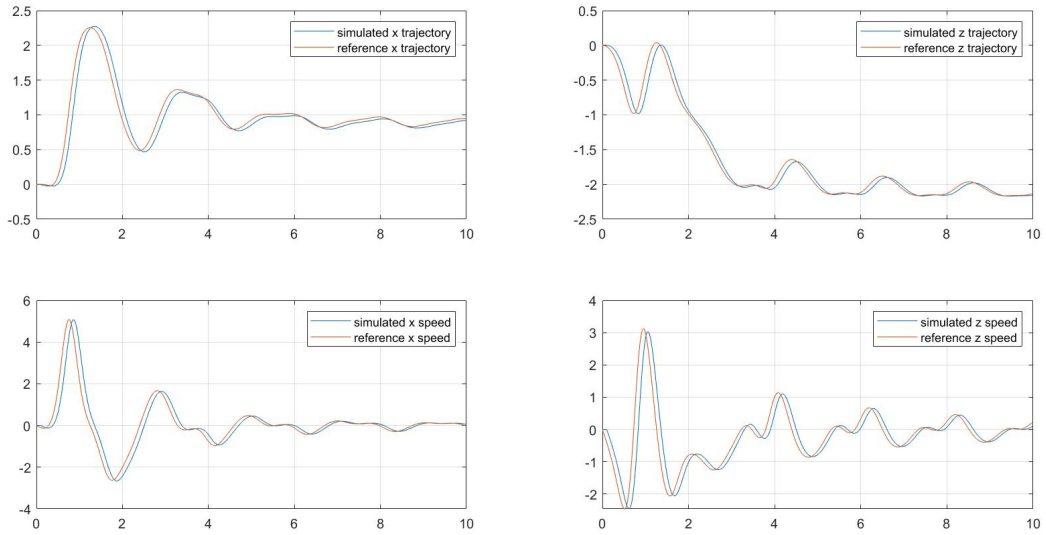

Figure 10: Position and speed, estimated vs reference

but the one the human is generating while trying to move the whole system. In both pairs of plots, we can see a very good tracking of the reference, but, as for the forces, there is a small time delay due to the ZOHs and the MHE approach.
The total computational time required by our estimation problem is approximately about 20 minutes; the bottleneck of our estimation problem is mainly due to the integrations, which are carried out in *Simulink*. Each time, during the optimization routine, in the $robot - sim - error$ function there is a call to the Simulink model $model - sim$. A possible alternative (not implemented neither tested) is to numerically integrate the acquired simulated data in the $robot - sim - error$ function.
With this filtering algorithm, possible applications can be performed. In this problem we focus on the estimation of the human disturbance acting on the end-effector, other possible solutions might be:

- Estimation of unknown time-varying forces acting on the end-effector of a robot (from any possible source);
- Any other application that requires a "virtual sensor" for the estimation of the force acting on the E-E, in any field of robotics.

5.2 Optimal control results

5.2.1 Optimal torques

In this second section, we are analysing the results concerning the optimal control problem. In this case, the problem is shifted from estimating the human force to finding the optimal values of the torque to be applied from the joints' motors. Now, the human is considered as an unknown disturbance acting on the robot. We can notice that, now the outer loop (which can be interpreted as a sensing loop of the human, trying to reach the desired trajectory and adjusting the position) is a closed one. The human transfer function is working on the error between the reference trajectory and the simulated one (coming from the direct kinematic block in Simulink after the robot dynamics, as shown in Figure 8). The inner loop, which is not explicitly designed in *Simulink*, is still working on the error between the q_{ref} and q_{sim} (refer to figure 8). The main idea was to substitute a classical PID approach, trying

to stabilize the system, with a more modern MPC approach. In Figure 11 we can see the optimal torques produced by the optimization program:

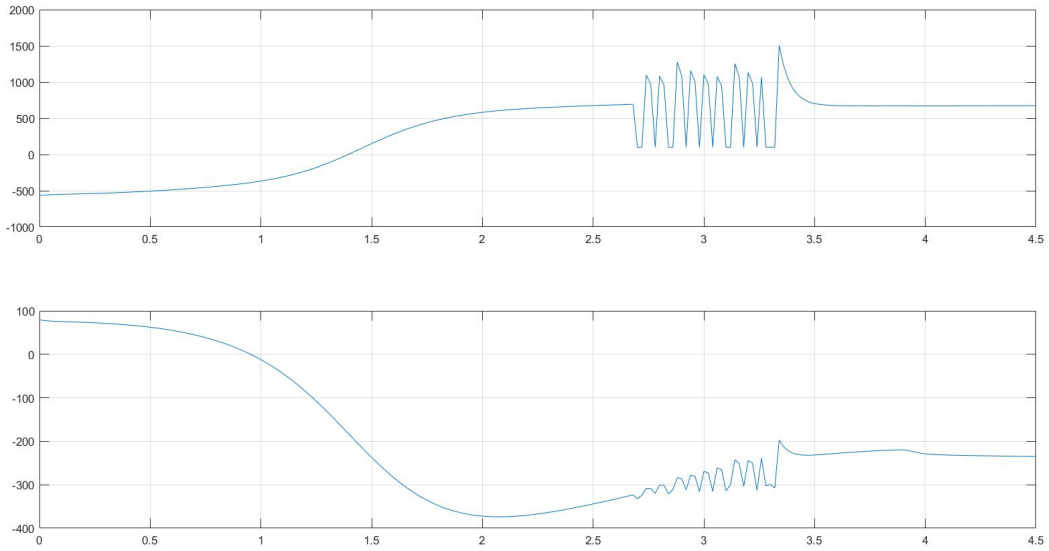

Figure 11: Optimal torques

We can see that there are some high frequencies oscillations between 2.5 and 3.5 seconds. These effects are probably due to integration carried out in *Simulink*. Moreover, if we plot the human forces, we can see that we manage to achieve our goal of helping the human. By putting saturation blocks of $\pm 100N$ on the human outer loop, we can clearly see that:

- Along x direction, the force starts with small values, saturating then at $-100N$ after about 3.1 seconds;
- Along z direction we never reach none of the boundaries, meaning that the robot is correctly helping him.

In Figure 12 we can see the plots of the forces.

5.2.2 Trajectory tracking

For what concern the trajectory tracking, we struggle a little bit, not fully obtaining the results we hoped for. Even if we tried tuning the different parameters of the optimization program (tolerances on the minimum variation of the gradient, minimum variation on the optimization variables etc.) we obtain a non-perfect tracking of the trajectory and big oscillations on the speed profiles. As we can see in the Figure 13, the results are acceptable, but not the best we were looking for.

As a final remark, notice that optimization routine as been stopped at 210 iterations, meaning that we have passed the 4 seconds transient. Therefore, all the plots have meaningful results in the interval 0-4 seconds. This optimal control scheme has worse performance than the one used in the estimation phase, because the human outer loop is in feedback. The time requested to complete the routine is about 2 hours and 30 minutes.

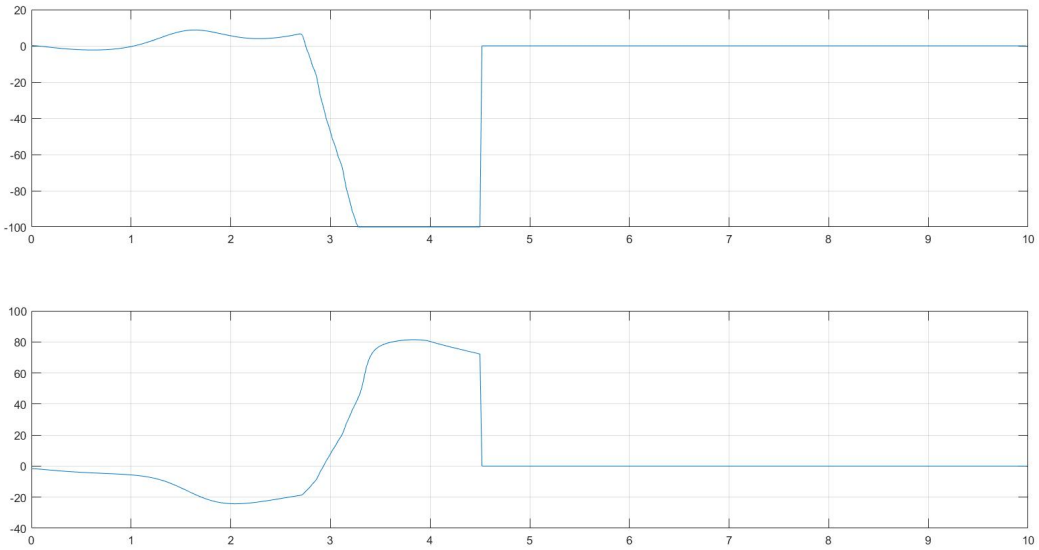

Figure 12: Human forces extracted from Simulink during the optimization routine

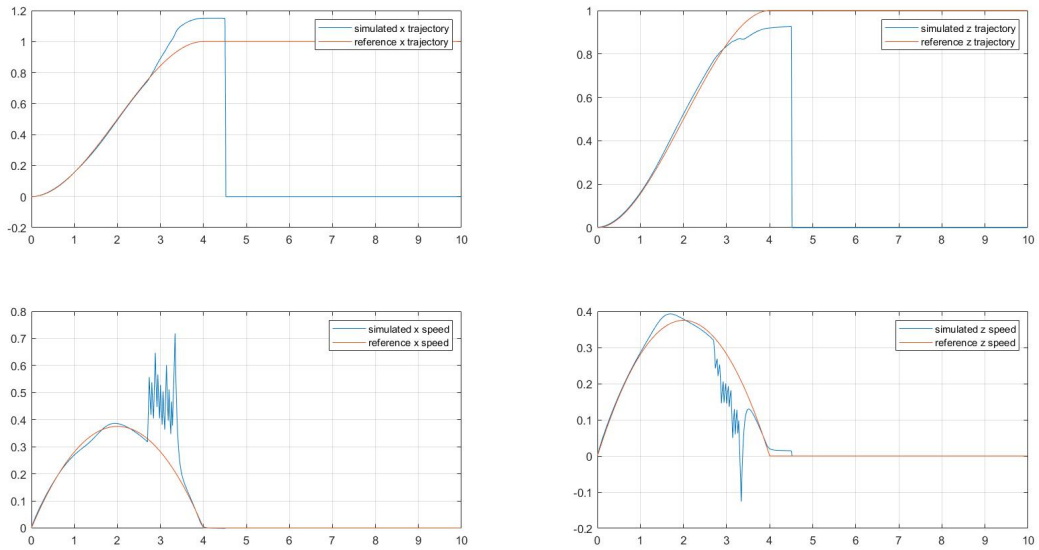

Figure 13: Position and speed trajectory

6 Final considerations

In this section we would like to point out some considerations that we have done to carry out our project.

1. We have considered noise free measurements from the ZOHs sampling. This is a quite strong assumption, but they can be introduced in the problem as we have seen in laboratory session B. Furthermore, the high frequencies contribution of the sensors' noise can be easily managed with classical approaches;

2. Both the two problems are unconstrained because the constraints are related only to the dynamics of the model, so they are embedded in the cost function. But we have used a minimizer function for constrained problem because it is easier to further implement features that require constraints;

3. In our model we considered a mass of 10 kg but our routines guarantee the robustness in case of mass variation.

In conclusion, we would like to highlight that the optimal control section of our project is an additional part that we develop in order to explore a more interesting application for us (hoping that the approach we follow is meaningful and correct!)

References

[1] Jan Boril, Rudolf Jalovecky. *Parameter estimation of transfer function of pilot behaviour model from measured data*

[2] Lalo Magni, Riccardo Scattolini. *Advanced and Multivariable Control* (12.2.4 Industrial MPC/Control Horizon)

[3] Lorenzo Fagiano. *Constrained Numerical for Estimation and Control - Lecture Notes*