ARC Assignment 03: Optimal Control Control

Andrea Carollo [206343] - Matteo Tomasi [206334]
Advanced Optimization-based Robot Control
University of Trento

June 11, 2020

Introduction

The aim of this assignment is to explore the world of the optimal control problem. However, since the force control was an interesting topic for us we choose to see how an optimal control can be used in order to include also a control of the force at the end-effector of a robot. To accommodate our preferences, we choose to control a robot that has to brush a surface with a certain force. In this way we can see how the optimal control minimizes some given cost functions, making the end-effector follows a desired trajectory.
For this purpose, we use the *Single Shooting* Optimal Control method coupled with a reactive *Force Control*. This allows to generate the trajectory by using the OCP and control use it as input reference for the reactive control. In addition, to the only Force Control, the reactive control is needed also to follow the trajectory with some errors in the model and to stabilize the robot in presence of contacts. This is because the contact dynamics is very hard to manage and makes the equations of motion stiff.

Procedure

In order to formulate the OCP problem in the correct way, we decided to proceed by steps introducing a new type of cost function at a time. This allows to check the correctness of the cost function formulation and to tune the weights and their impact on the motion. Moreover, to be sure that the cost gradients implementations are correct, we also check them by using the *sanity_check()* function that compares the gradients of the cost function computed by us with the estimated ones using the finite differences technique.

$$L = \sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N) \tag{1}$$

In the form 1, is reported the expression of the total cost. We can easily distinguish the running cost - inside the summation - with the final cost - at the end -. This form explains quite well what are these two costs and what the OCP aims to minimize.
Starting from joint space, we apply to the problem the following cost functions progressively:

- quadratic final cost on the final position error - in joint space - and on the final velocity;
- quadratic running cost on the input torques;

- running cost on a desired posture;
- running cost to follow a trajectory in joint space - from one robot configuration to another -;

Having good results in joint space, we move our analysis in task space, using the same approach:

- quadratic final cost on the final position error - in task space - and on the final velocity;
- quadratic running cost on the input torques;
- running cost to follow a trajectory in task space;

We notice immediately that the robot redundancy makes it fall down. This happens because the initial guess for the solver is zero. To avoid this behaviour and also to help the solver in the following steps, we choose to impose the gravity compensation term as initial guess for the controls. This means that the solver has not to compensate it starting from zero and, consequently, the convergence is faster. In addition, to avoid the "falling", we add a postural cost function giving the initial condition as reference.
We note that there are some overshoots in the trajectory tracking and, for this reason, we decide to run over the trajectory with a velocity profile that starts from zero, reaches a maximum speed and goes to zero again at the final time. This type of velocity profile helps the solver to track the trajectory.

Once we are able to follow the trajectory also in task space, we move our attention to the contact dynamics. We modify the ODE of the robot considering also the contacts. To make that, we use a similar version of the already implemented class *ContactSurface* and *ContactPoint* that are called by a new version of the *ODERobot* class to take care also of the contacts. The two classes that manage the contacts use the soft contacts theory that associates to the contact surface a stiffness - K - and a damping - B -. The contact force can be written as follow where p is the contact point of the robot and p_0 is the anchor point on the surface.

$$F_{contact} = K\ (p_0 - p) - B\ \dot{p} \tag{2}$$

In order to integrate the problem dynamics, we use small value of the contact stiffness and put to zero the contact damping.

For the reactive control, we decide to take the states from the optimal control solution as reference to track. For this reason we implement a simple PD controller on the joint position in order follow the OCP output. In order to control the force acting on the contact surface, we take a constant force as reference - value of 50N - and choose to track that force with a feed forward plus a feedback proportional loop.
The overall control torques have a very simple formulation.

$$f^* = F_{des} + K_p\ (F_{des} - F) \quad \text{and} \quad \tau_f = J^T f^* \tag{3}$$

Where F_{des} is the desired force, F is the sense contact force and J is the jacobian matrix of the contact point.

$$\ddot{q}_0 = K_{p,j}\, e - K_{d,j}\, \dot{e} = K_{p,j}\, (q_{ref} - q) - K_{d,j}\, (\dot{q}_{ref} - \dot{q}) \quad \text{and} \quad \tau_e = M\, \ddot{q}_0 \tag{4}$$

Where q_{ref} and $\dot{q}_{ref}$ are the reference states to track from the OCP, q and $\dot{q}$ are the real states and M is the system mass matrix.

$$\tau = \tau_e + \tau_f \tag{5}$$

The total control torque τ is the sum of the two contribute, force and tracking.
The optimal control works with a value of time step that is higher with respect to the classical controller. This is due to the fact that the OCP requires a lot of computation and decreasing too much the time step, the computation time will increase exponentially without giving a significant improvement in term of precision. For this reason, to have a good tracking performance of the reactive control, we use an higher frequency for this part, interpolating the solution of the OCP in linear way inside the Optimal Control time step. To further improve the reactive control tracking, we could tune properly the PD gains and, eventually add a friction compensation term.

Challenges faced

The first problem that we addressed is the cost function weight tuning: this step has a strong impact on the OCP solution since it gives the relative importance to the cost functions that we want to minimize. Mainly in task space, this operation gives us some problems also because imagine the solution of the solver is hard considering the redundancy.
Another issue that we faced on is the contact dynamics. The solver is not able to solve directly the problem with contacts. For this reason, we solved two different types problem: firstly, we solve the problem without contacts and secondly, using the solution of the first OCP as initial guess, we solve the problem with the contacts. Since the solution has already optimized the second OCP performs few iterations. However, since the problem becomes stiff using the contacts, we try also to give directly the solution of the first problem to the reactive control and see the differences.
We also deal with the problem of end-effector orientation. This problem is harder then the simple position because the orientation is described by a matrix - not by a vector - and so manage it is harder. However, since the surface in which we have to brush is the floor, the problem is a bit simplified. To handle this problem we test three different types of cost functions. The first one is based on robot geometry property: to keep the end-effector orthogonal to the surface, the joint coordinates should satisfy $q_1 + q_2 = q_3$ and $q_4 = 3\,\pi/2$. The second one is related to the form of the rotation matrix: the columns of the rotation matrix are the unitary vectors that describe the new reference system in the first coordinates system. If we look at the vector of interest and put it equal to a unitary vector with the desired orientation we should be able to impose the vector orientation - the gradients of the rotation matrix elements are found analytically since they do not depends on the robot geometry -. The last method that we implement uses the *Pinocchio log3()* function to know

how the end-effector is oriented in the space. We try to handle with this function but the online documentation is not so clear so we are not sure to implement it in the correct way. However, despite our effort, none of these cost functions work properly. We leave them in the code to see the implementation.

Conclusion

For the testing phase the imposed trajectory on the floor is a 90° circumference arc centred on the basement of the robot with radius of 0.5m. Firstly, we focus on the case in which the OCP solution is directly given, as input to the reactive control.

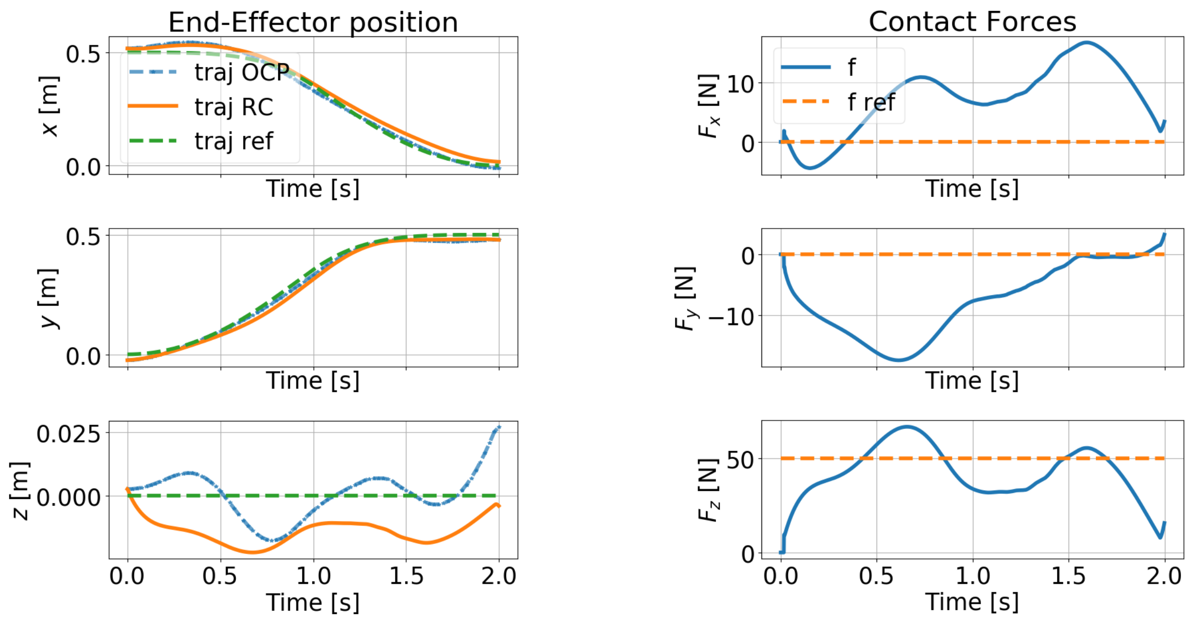

Figure 1: End-effector position and contact forces for the OCP wo contacts plus RC

In the figure 1, we can see that the trajectory imposed at the end-effector is well followed by the OCP and also the OCP solution is followed in a good way from the reactive control. The z coordinate from the OCP tends to goes up at the end of the trajectory but the contribute of the force control keeps the height of the end effector on the floor plane. Looking at the force, we can see that the F_x and F_y does not follow the reference. This is because of the friction in the contact surface. However, the real desired force contribute - F_z - is followed in a quite good way. If we would have a better performance in the force tracking we could increase the force proportional gain but we notice some strong oscillations if we increase it too much.
Secondly, we move to the results obtained with the two OCP - with and without contacts - plus the reactive control.

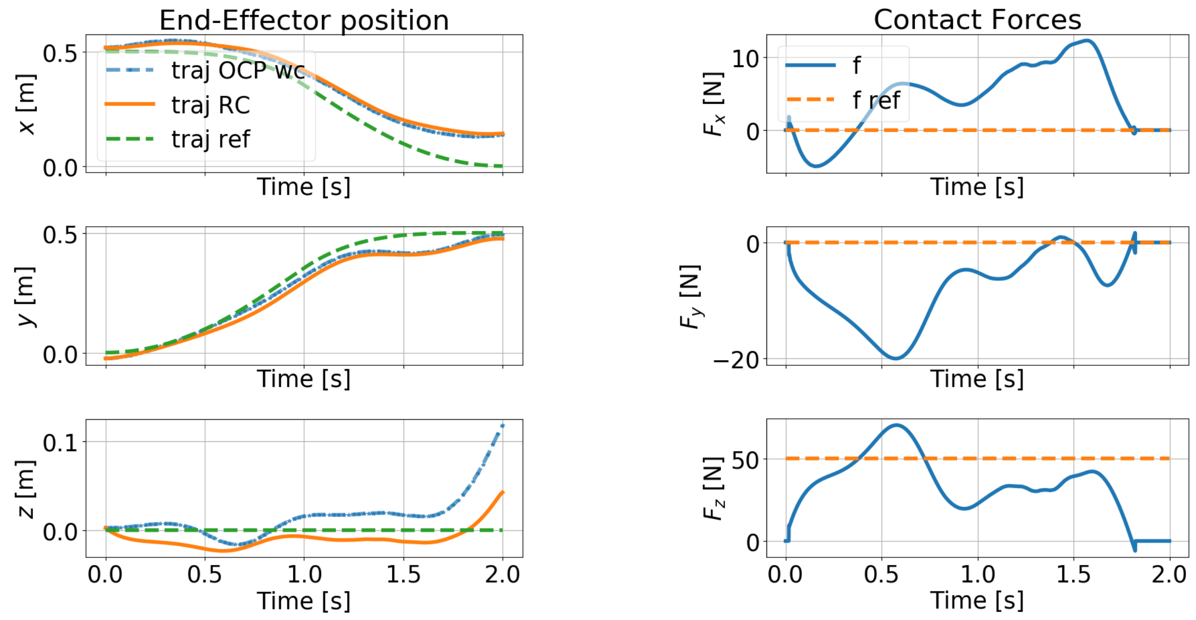

Figure 2: End-effector position and contact forces for the OCP w contacts plus RC

In this case - figure 2 -, we can see that the presence of the contact creates an error in the tracking of the reference trajectory in the second OCP. This is due to the fact that the surface presents an elastic contribute that makes the robot bump on it. The tracking performance of the reactive control is still good - remember that it tracks the OCP states - and the force control tends to push the z coordinates downward. This works until the last part of the trajectory where the error in z becomes too large and the tracking term becomes the dominant part of the controls. Also in this case, the desired force is well followed.
We test the code also with model randomization and coulomb friction. Notice that these terms do not affect the OCP solution since the OCP makes its computation on the ideal ODE function. The effects of coulomb friction and randomization are visible in a decrease of the reactive control performance both for tracking and for force control. However, this decreasing is low and it can be partially compensated by using a friction compensation strategy.
To conclude our analysis, we can say that the Optimal Control Problem works better without contacts in general. The reactive control can be a very useful technique to handle with model randomization and coulomb friction since they are not considered in the OCP.
To proceed with this project could be interesting to correctly implement the part that takes the end-effector orthogonal to the surface since the links closer to it penetrates the floor now. Moreover, control the force using the Optimal Control is still interesting so we could increase the OCP control size in order to consider the end-effector force as a control variable of our problem.
Finally, we notice that, at the end at the trajectory, the OCP solution tends to diverge. To correct this we could move to a more robust Optimal Control strategy like *Multiple-Shooting* or *Collocation*.