Optimization of a Pendulum System using Optimica and Modelica

Pontus Giselsson[a] Johan Åkesson[a,b] Anders Robertsson[a]
[a)] Dept. of Automatic Control, Lund University, Sweden
[b)] Modelon AB, Sweden

Abstract

In this paper Modelica and Optimica are used to solve two different optimal control problems for a system consisting of a pendulum and a cart. These optimizations demonstrates that Optimica is easy to use and powerful when optimizing systems with highly non-linear dynamics. The optimal control trajectories are applied to a real pendulum and cart system, in open loop as well as in closed loop with an MPC-controller. The experiments show that optimal trajectories from Optimica together with MPC feedback is a suitable control structure when optimal transitions through non-linear dynamics are desired.

Keywords: Optimal control, Optimica, Modelica

1 Introduction

Optimal control problems for dynamical systems with non-linear dynamics often lead to non-convex optimization problems. These problems are usually difficult to solve and lots of time and effort is usually spent on transforming the optimal control problem into a numerical optimization problem. In this paper we use the high-level languages Modelica together with Optimica to solve two different optimal control problems for a pendulum and cart system. The Modelica and Optimica combination allows the user to concentrate on how to formulate the optimal control problem, rather than on how to transform it into a numerical optimization problem. The pendulum dynamics are highly non-linear which makes this an appropriate application to show the efficiency of the Optimica and Modelica combination. The first optimization problem considered in the paper is to swing up the pendulum from its downward position to the inverted position in as short time as possible. The second problem is to move the cart from one position of the track to another in as short time as possible, while the pendulum end point, i.e., the end of the pendulum that is not attached to the cart, must avoiding an elliptical obstacle.

We also present experimental results where the optimal control trajectories are applied to a real pendulum on a cart system. There is a close match between the optimal trajectories and the real system trajectories when no or small disturbances are present. This demonstrates that optimal control is applicable to the real process. Of course, we get good experimental results when the process is accurately described by the model. When larger disturbances are present, e.g., in the initial conditions, the optimal control trajectories applied to the real process result, as expected, in state trajectories that are far from the optimal ones. A Model Predictive Controller (MPC) is introduced to minimize the influence of these disturbances. Experimental results show that the combination of optimal control feed-forward and MPC-feedback is a suitable control structure for these problems where optimal transitions through non-linear dynamics are desired.

The remainder of the paper is organized as follows. In Section 2 an introduction to the Modelica extension Optimica is given. Section 3 describes the cart and pendulum process used in the paper. In Section 4 we state and solve two optimization problems using Optimica and Modelica. Results from the optimizations are applied to the real pendulum, in open loop as well as in closed loop with an MPC-controller, in Section 5. Section 6 describes how Optimica and this particular application is used in the education at the Dept. of Automatic Control, Lund University. Finally in Section 7 we give some conclusions.

2 Optimica and JModelica.org

Modelica does not offer explicit support for formulation of dynamic optimization problems. In particular, means to express quantities such as cost function, con-

straints, optimization interval, and optimization parameters are lacking. In an effort to extend Modelica to also include high-level formulation of dynamic optimization problems, the Optimica extension was proposed [1]. The Optimica extension is supported by the novel Modelica-based open source platform JModelica.org [8].

2.1 JModelica.org

JModelica.org is a novel Modelica-based open source project targeted at dynamic optimization [2], [3]. JModelica.org features compilers supporting code generation of Modelica models to C, a C API for evaluating model equations and their derivatives and optimization algorithms. The compilers and the model C API has also been interfaced with Python [6] in order to enable scripting and custom application development. In order to support formulation of dynamic optimization of Modelica models, JModelica.org supports the Optimica extension [1]. Optimica offers constructs for encoding of cost functions, constraints, the optimization interval with fixed or free end points as well as specification of transcription scheme.

The JModelica.org platform contains an implementation of a simultaneous optimization method based on orthogonal collocation on finite elements [5]. Using this method, state and input profiles are parametrized by Lagrange polynomials, of order three and four respectively, based on Radau points. This method corresponds to a fully implicit Runge-Kutta method, and accordingly it possesses well known and strong stability properties. By parameterizing the variable profiles by polynomials, the dynamic optimization problem is translated into a non-linear programming (NLP) problem which may be solved by a numerical NLP solver. This NLP is, however, very large. In order to efficiently find a solution to the NLP, derivative information as well as the sparsity patterns of the constraint Jacobians need to be provided to the solver. The simultaneous optimization algorithm has been interfaced with the large-scale NLP solver IPOPT [10], which has been developed particularly to solve NLP problems arising in simultaneous dynamic optimization methods.

The choice of a simultaneous optimization algorithm fits well with the properties of the dynamic optimization problems treated in this paper. In particular, simultaneous methods handle unstable systems well, and also, state and input inequality constraints are easily incorporated.

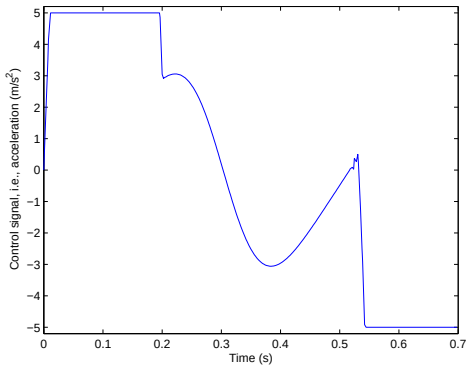

Figure 1: Control signal in the constrained double integrator example in Section 2.2.

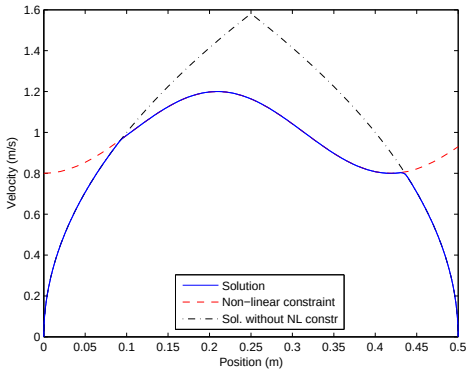

Figure 2: Phase plot of x and $\dot{x}$ in the constrained double integrator example in Section 2.2. Also the non-linear constraint and the solution to the unconstrained problem are plotted.

2.2 Optimica example

In this section, the Optimica syntax is explained by stating and solving a double integrator optimization problem. The example will also serve as an evaluation of the accuracy of the Optimica solution compared to the optimal solution. The following optimization problem is solved:

$$\begin{aligned} \min_{u} \quad & t_f \\ \text{subject to} \quad & \ddot{x} = u \\ & 0.2\cos 15x + \dot{x} \leq 1 \\ & |u| \leq 5 \\ & x(0) = 0 \qquad \dot{x}(0) = 0 \\ & x(t_f) = 0.5 \quad \dot{x}(t_f) = 0 \end{aligned} \tag{1}$$

where t_f is the final time, u is the control signal, x is the position and $\dot{x}$ is the velocity. The non-linear constraint is added to make the problem a bit more complex. A Modelica model for a double integrator is:

```
model DoubleIntegrator
  package SI = Modelica.SIunits;
  SI.Position x(start=0);
  SI.Velocity x_dot(start=0);
  input SI.Acceleration u;
equation
  der(x) = x_dot;
  der(x_dot) = u;
end DoubleIntegrator;
```

An Optimica specification of the problem is:

```
optimization DIopt (objective=finalTime,
                    startTime=0,
                    finalTime=(free=true,
                               initialGuess=1))
  DoubleIntegrator DI(u(free=true,
                        initialGuess=0.0));
constraint
  DI.x(finalTime)=0.5;
  DI.x_dot(finalTime)=0;
  0.2*cos(15*DI.x)+DI.x_dot <= 1;
  DI.u <= 5;
  DI.u >= -5;
end DIopt;
```

In the first line of the Optimica specification the optimization objective is specified. In this case the objective to be minimized is the final time. Then the Modelica model of the dynamical system that is used in the optimization is specified and u is chosen to be the decision variable. Then all constraints, inequality as well as equality constraints, are listed.

The solution to (1) is obviously to accelerate with maximum positive acceleration until, or if, the constraint is reached. Then continue with maximum allowed velocity until deceleration is needed to reach $x = 0.5$ and $\dot{x} = 0$. This behaviour is clearly seen in the Optimica solution of the problem, Figures 1 and 2.

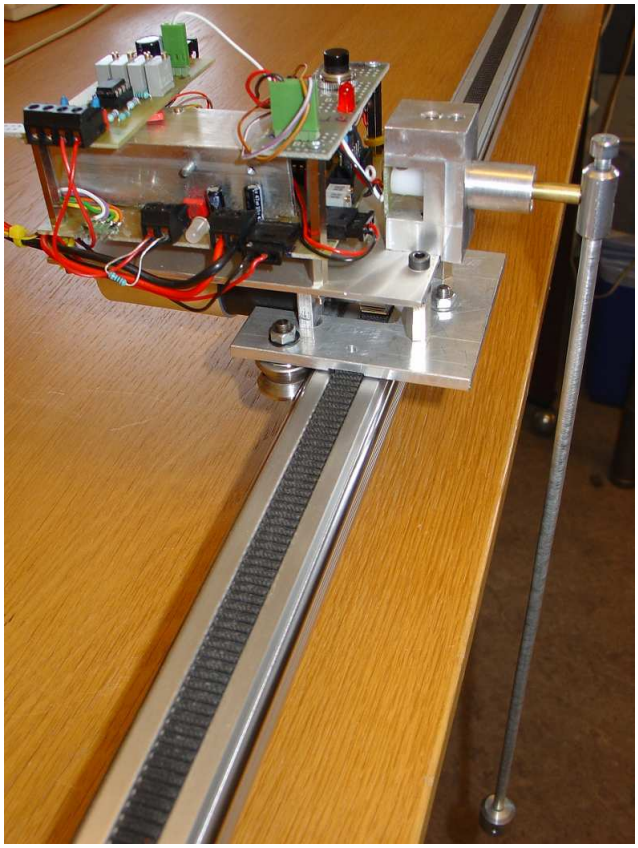

Figure 3: Photo of the cart and pendulum system described in Section 3.

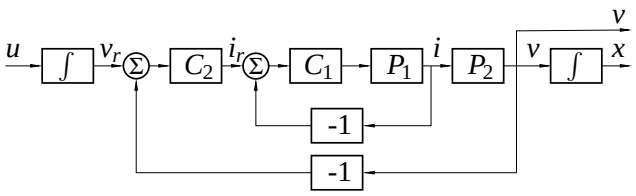

Figure 4: Cascaded control structure for the cart control.

3 The Process

The Department of Automatic Control in Lund has a history of designing and building laboratory processes. One of the latest processes that are built in-house is the pendulum and cart process depicted in Figure 3. This process is used in this paper to demonstrate the applicability of Optimica and optimal control.

3.1 Cart control

The cart is driven by a DC-motor which is controlled in a cascaded structure. See Figure 4 for a schematic view of the cascaded control structure. There is an inner loop that controls the current through the DC-motor. P_1 represents the current dynamics which behaves like a first order system with a time-constant of 0.17 ms. C_1 represents the PI-controller in the current loop that controls the current, i, to its reference, i_r. The current reference, i_r, is set by the outer loop that controls the cart velocity. The current dynamics are fast in comparison to the velocity dynamics, which makes $i_r \approx i$ a good approximation. The transfer function from i to v, i.e. P_2, is ideally an integrator with a gain. The velocity dynamics are controlled with another PI-controller, C_2. The reference to the velocity control loop, v_r, is integrated from an acceleration reference, u, since acceleration is our desired control signal. This cascaded control structure is suitable when fast closed loop dynamics from v_r to v is desired. Since $v_r \approx v$ is a good approximation, we have double integrator dynamics from control signal, u, to cart position, x.

3.2 Hardware setup

On the cart there are two Atmel ATmega16 micro processors. The current controller, C_1 in Figure 4, discussed in Section 3.1, is running on one of them at a sampling rate of 28.8 kHz. This micro processor gets the current reference, i_r, from the other micro processor, where the velocity controller, C_2, is running at 1 kHz. This second micro processor also communicates with a PC via the serial interface. This communication is performed at frequencies around 50 Hz. From Matlab/Simulink on the PC, the velocity reference, v_r, is sent to the velocity controller on the micro processor. The velocity reference is obtained by integrating the acceleration reference, u, on the PC-side. Since a smooth acceleration profile of the cart is desired, the velocity reference needs to be updated more frequently than at 50 Hz. Therefore the acceleration reference, u, is also sent to the velocity controller from the PC. The velocity reference is updated in the micro processor at a frequency of 1 kHz according to $v_r(t) = v_r(t_0) + u(t_0)(t - t_0)$, where t_0 is the time when the last references was received from the PC, $t \in [t_0, t_0 + h]$ and h is the PC communication sampling time. These updates are consistent with the velocity reference in the next sample from the PC which is $v_r(t_0 + h) = v_r(t_0) + u(t_0)h$.

One alternative would be to send only the acceleration reference to the micro processor and to calculate the velocity reference there. This would imply that in order to stop the cart, it must be controlled with a feedback loop on the PC. With our implementation structure the cart can easily be stopped by setting the velocity reference to zero.

The PC also receives cart position and pendulum angle measurements as well as velocity estimates from the micro processor. This enables for us to, on the PC, create another level of feedback loops in the cascaded control structure.

3.3 System modeling

Due to the low level control of the cart, described in Section 3.1, the cart behaves as a double integrator. When x is the cart position and u the control signal, we have the following cart dynamics

$$\ddot{x} = u$$

The pendulum dynamics are well known; let θ be the pendulum angle and we get

$$\ddot{\theta} = -\frac{g}{l}\sin\theta + \frac{a}{l}\cos\theta$$

```
model Pendulum
  package SI = Modelica.SIunits;
  parameter SI.Length l = 0.4;
  constant SI.Acceleration g = 9.81;
  SI.Position x(start=0);
  SI.Velocity x_dot(start=0);
  SI.Angle theta(start=0);
  SI.AngularVelocity theta_dot(start=0);
  SI.Position x_p;
  SI.Position y_p;
  Real u_dot(unit="m/s3");
  input SI.Acceleration u;
equation
  der(x) = x_dot;
  der(x_dot) = u;
  der(theta) = theta_dot;
  der(theta_dot)=-g/l*sin(theta)+1/l*cos(theta)*u;
  der(u) = u_dot;

  x_p = x-l*sin(theta);
  y_p = -l*cos(theta);
end Pendulum;
```

Listing 1: A Modelica model for the pendulum and cart system.

where $\theta = 0$ is defined to be the pendulum downward position, g is the gravitational acceleration, l is the pendulum length and a is the horizontal acceleration of the pendulum pivot point. This horizontal acceleration, a, is equal to the cart acceleration, $\ddot{x}$, and thus equal to the control signal, u. This gives us the following model for the complete system

$$\ddot{\theta} = -\frac{g}{l}\sin\theta + \frac{u}{l}\cos\theta \quad (2)$$

$$\ddot{x} = u \quad (3)$$

A schematic view of the full system is found in Figure 5, where S_{pend} represents the non-linear pendulum dynamics (2) and G_{cart} represents the double integrator dynamics (3), z is a vector containing the states, $z = (x\ \dot{x}\ \theta\ \dot{\theta})^T$ and S represents the full system. The position of the cart and the pendulum angle are defined such that the pendulum end point in the horizontal direction, x_p, and in the vertical direction, y_p, are given

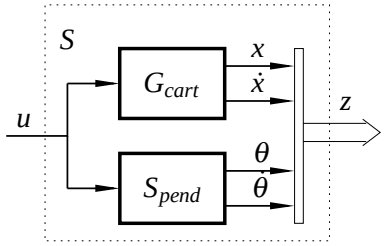

Figure 5: Schematic view of the system.

by

$$x_p = x - l\sin\theta$$
$$y_p = -l\cos\theta$$

The Modelica model that describes this pendulum and cart system is found in Listing 1.

4 Optimization

In this section we use Modelica and Optimica to solve two different optimization problems based on the pendulum and cart model. The first problem is to swing up the pendulum in as short time as possible. The second problem is to move the pendulum and cart from rest at one cart position on the track to another, while the end point of the pendulum must avoid an elliptical obstacle. Also in this second problem the objective to be minimized is the final time.

4.1 Time-optimal Swing-up

The optimization objective is to swing up the pendulum from the downward pendulum position to the inverted pendulum position in as short time as possible. The cart should stop at the same position as it started and the cart and angular velocities should be zero at the final time. The control signal, i.e., the cart acceleration, u, is limited to the interval $\pm$ 5 m/s^2. Its derivative, $\dot{u}$, is limited to the interval $\pm$ 100 m/s^3. The cart track is limited, which lead to constraints in the cart position. The cart position must satisfy -0.5 m $\leq x \leq 0.5$ m. The optimization problem is stated mathematically in (4)

$$\begin{aligned}
\min_{u} \quad & t_f \\
\text{subject to} \quad & \ddot{\theta} = -\tfrac{g}{l}\sin\theta + \tfrac{u}{l}\cos\theta \\
& \ddot{x} = u \\
& -0.5 \leq x \leq 0.5 \\
& |u| \leq 5 \qquad |\dot{u}| \leq 100 \\
& \theta(0) = 0 \qquad \dot{\theta}(0) = 0 \\
& x(0) = 0 \qquad \dot{x}(0) = 0 \\
& \theta(t_f) = \pi \qquad \dot{\theta}(t_f) = 0 \\
& x(t_f) = 0 \qquad \dot{x}(t_f) = 0
\end{aligned} \tag{4}$$

where t_f is the final time. The Modelica and Optimica codes that describe the optimization problem are found in Listings 1 and 2 respectively. The resulting time optimal state and control trajectories are found in Figures 6 and 7. The pendulum angle changes sign two times during the swing-up. It starts with a positive angle, switches to negative and finally it reaches its inverted position with a positive angle. This means that

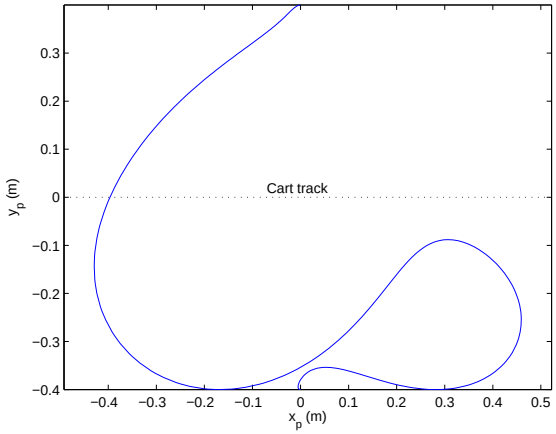

Figure 6: Optimal trajectory of the pendulum end point for swing-up problem in Section 4.1.

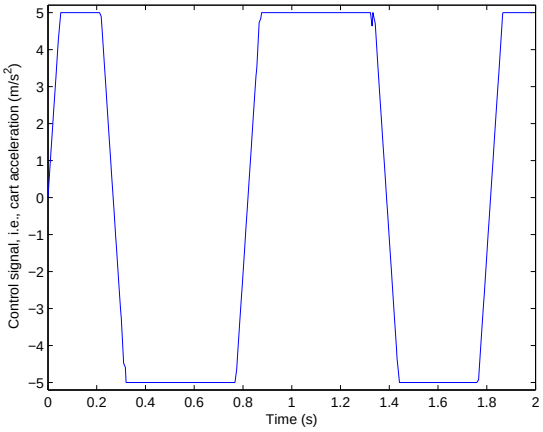

Figure 7: Optimal control signal, i.e., cart acceleration, for swing-up problem in Section 4.1.

the optimal swing-up is performed with three swings before the inverted position is reached. In [4] the minimum number of pendulum swings needed for swing-up, given a maximum acceleration of the pendulum pivot point, a_{max}, is analyzed. Three swings are needed if $0.388g \leq a_{max} \leq 0.577g$ which is equivalent to 3.81 m/s^2 $\leq a_{max} \leq$ 5.66 m/s^2. This analysis is not directly applicable to our setup since cart constraints and acceleration rate limitations are not considered in the analysis in [4]. When cart terminal position and acceleration rate constraints are chosen as in our setup, three swings are needed for swing-up if 4.45 m/s^2 $\leq a_{max} \leq$ 7.70 m/s^2. This interval is obtained simply by solving the swing-up problem with different acceleration constraints in Optimica. The fact that our problem with additional constraints requires more acceleration to swing-up the pendulum with a fixed number of swings, is not surprising. The max-

```
optimization Swingup (objective=finalTime,
                      startTime=0,
                      finalTime=(free=true,
                                 initialGuess=1))
  Pendulum pend(u(free=true,initialGuess=0.0));
constraint
  pend.x(finalTime)=0;
  pend.x_dot(finalTime)=0;
  pend.theta(finalTime)=3.1415;
  pend.theta_dot(finalTime)=0;

  pend.x <= 0.5;
  pend.x >= -0.5;
  pend.u <= 5;
  pend.u >= -5;
  pend.u_dot <= 100;
  pend.u_dot >= -100;
end Swingup;
```

Listing 2: An Optimica model for time optimal swing-up of the pendulum.

imum acceleration in our example is 5 m/s^2 which is within the interval where a minimum of three swings are needed.

In [4] they also discuss an energy based swing-up strategy that was originally proposed in [11]. The idea of the method is to control the system to the energy-level that corresponds to the inverted pendulum position using maximum acceleration in either way. When this energy based approach is applied to this system, with $a_{max} = 5$ m/s^2, the pendulum reaches its inverted position when the cart position is approximately 3m from its starting point. This position is far outside the track, which is why this energy based method is not directly applicable when track limitations are present.

4.2 Optimization with path-constraints

In this optimization problem we want the cart to start at rest at position $x = 0$ with the pendulum in the downward pendulum position, $\theta = 0$. At the final time, the cart and pendulum should be at rest at position $x = 0.8$ m and pendulum angle $\theta = 0$. We also introduce an additional constraint stating that the end point of the pendulum must never enter an elliptical area described by

$$\left(\frac{x_p - 0.5}{0.05}\right)^2 + \left(\frac{y_p + 0.4}{0.3}\right)^2 = 1$$

Due to the track limitations we need the cart position to satisfy -0.1 m $\leq x \leq 0.9$ m. The control signal limitations are the same as in the previous optimization problem, i.e. -5 m/s^2 $\leq u \leq 5$ m/s^2 and -100 m/s^3 $\leq \dot{u} \leq 100$ m/s^3. The objective of the optimization is to reach the final states as fast as possible. The optimization problem is described mathematically in (5)

$$\begin{aligned}
\min_{u} \quad & t_f \\
\text{subject to} \quad & \ddot{\theta} = -\tfrac{g}{l}\sin\theta + \tfrac{u}{l}\cos\theta \\
& \ddot{x} = u \\
& x_p = x - l\sin\theta \\
& y_p = -l\cos\theta \\
& \left(\tfrac{x_p - 0.5}{0.05}\right)^2 + \left(\tfrac{y_p + 0.4}{0.3}\right)^2 \geq 1 \\
& -0.1 \leq x \leq 0.9 \\
& |u| \leq 5 \qquad |\dot{u}| \leq 100 \\
& \theta(0) = 0 \qquad \dot{\theta}(0) = 0 \\
& x(0) = 0 \qquad \dot{x}(0) = 0 \\
& \theta(t_f) = 0 \qquad \dot{\theta}(t_f) = 0 \\
& x(t_f) = 0.8 \qquad \dot{x}(t_f) = 0
\end{aligned} \tag{5}$$

where t_f again is the final time. The codes in the corresponding Modelica and Optimica files are found in Listings 1 and 3 respectively. This problem turns out

```
optimization Path (objective=finalTime,
                   startTime=0,
                   finalTime=(free=true,
                              initialGuess=1))
  Pendulum pend(u(free=true,initialGuess=0.0));
constraint
  pend.x(finalTime)=0.8;
  pend.x_dot(finalTime)=0;
  pend.theta(finalTime)=0;
  pend.theta_dot(finalTime)=0;

  pend.x <= 0.9;
  pend.x >= -0.1;
  pend.u <= 5;
  pend.u >= -5;
  pend.u_dot <= 100;
  pend.u_dot >= -100;

  ((pend.x_p-0.5)/0.05)^2+((pend.y_p+0.4)/0.3)^2>=1;
end Path;
```

Listing 3: An Optimica model for the path following problem.

to be more difficult to solve than the swing-up problem. Actually it is not easy to find a solution that is feasible, i.e., that satisfies all constraints. In order to solve this problem we need to give the solver an initial guess that is feasible and not too far away from the optimum. One crucial decision to make is if the pendulum should follow behind the cart over the obstacle, or if it should go in front of the cart. It turns out that if the pendulum follows behind the cart we get

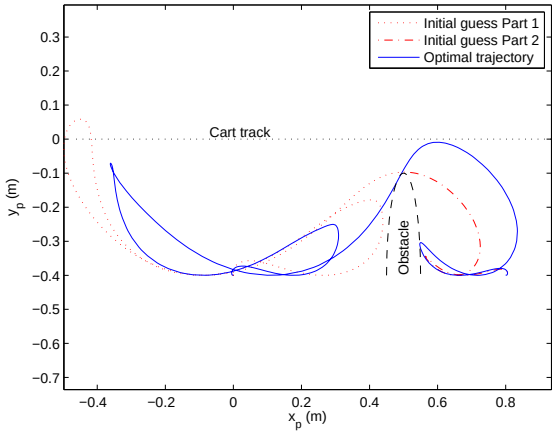

Figure 8: Optimal trajectory of the pendulum end point for path constrained problem in Section 4.2. Also the pendulum end point in the two parts of the initial guess is plotted.

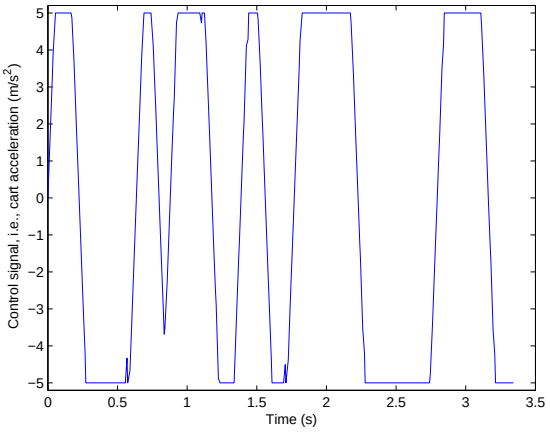

Figure 9: Optimal control signal, i.e., cart acceleration, for path constrained problem in Section 4.2.

very large oscillations after passing the obstacle. It is time-inefficient to damp these resulting pendulum oscillations because of the track and control limitations. Thus the time-optimal solution must have the pendulum in front of the cart when passing the obstacle. To help the optimizer finding this solution the problem is divided into two smaller and easier subproblems.

The first subproblem is an altered version of the original problem (5). The elliptical constraint is removed and the final constraints are set to

$$\begin{aligned} \theta(T) &= -\tfrac{75.52\pi}{180} & \dot{\theta}(T) &= 0 \\ x(T) &= 0.1127 & \dot{x}(T) &= 1.4 \end{aligned} \tag{6}$$

This terminal point of the optimization corresponds to when the pendulum is precisely above the obstacle with the pendulum leaning in the forward direction.

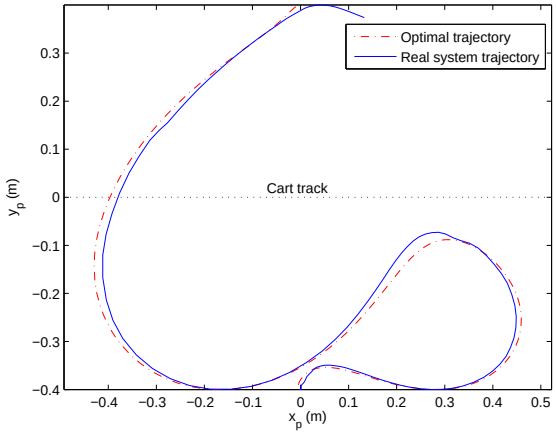

Figure 10: Experimental results for swing-up problem when control trajectory applied in open loop as described in Section 5.1. The optimal pendulum end point trajectory is also plotted for comparison reasons.

The terminal cart velocity, $\dot{x}$, is set to a positive value since we want the cart to have a forward motion over the obstacle. The angular velocity of the pendulum, $\dot{\theta}$, is set to zero which makes it possible for the pendulum angle to decrease directly after passing the obstacle. The terminal cart and pendulum angular velocities are chosen intuitively to enable a fast transition from above the obstacle to the terminal point of the original problem (5).

The second subproblem continues from where the first subproblem terminated. The initial conditions in the second subproblem are the same as the terminal constraints of the first subproblem, (6). The terminal constraints of this second subproblem are the same as in the original problem, (5). This means that the pendulum continues on the other side of the obstacle until it reaches the terminal point.

The resulting optimal trajectories of the two subproblems are then merged and given as an initial guess when solving the original problem. Given this initial guess, the solver converges to the optimal solution. The resulting pendulum end point movements for the two parts of the initial guess and for the optimal solution are found in Figure 8. The control signal for the optimal solution is found in Figure 9. The final time for the merged initial guess is 3.39 s while the optimal solution has a final time of 3.34 s. The first part of the initial guess takes 2.18 s while the second part is performed in 1.21 s. The corresponding first and second parts of the optimal solution take 1.94 s and 1.40 s respectively. This means that the intuition behind the choice of terminal constraints for the first subproblem

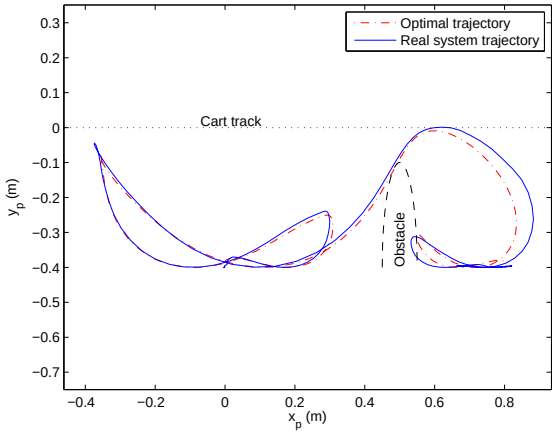

Figure 11: Experimental results for path constrained problem when control trajectory applied in open loop as described in Section 5.1. The optimal pendulum end point trajectory is also plotted for comparison reasons.

(6), i.e., to enable for a fast second part, is good. The second part of the initial guess is fast and the merged initial guess is not very far from the optimal one in terms of the optimization objective, namely the final time, t_f.

5 Experiments on the real Pendulum

In this section the optimal control trajectories obtained in the previous section are applied to the real system. These experiments will serve as an evaluation of how well the model describes the actual system and it will show the practical applicability of optimal control trajectories in a real system.

5.1 Open loop results

Figures 10 and 11 show how the real system responds to the optimal control trajectories. The figures also show the optimal trajectories from the previous section for comparison reasons. The trajectories are very similar, which means that the model of the system is accurate.

In the optimizations it is assumed that the initial conditions of the pendulum and cart are such that the cart is at rest at position, $x = 0$, and the pendulum is at rest at angle $\theta = 0$. If the experiments are performed with initial conditions of the pendulum that do not satisfy the assumed ones, i.e., if the pendulum is swinging when the experiment is started, we get results as shown in Figures 12 and 13. The magnitude of the initial swings are approximately 45° in these experiments. The figures show that we are far from reaching our objectives when this kind of disturbances are present. To make the optimization results usable in reality, we need feedback to take care of deviations from the optimal trajectories.

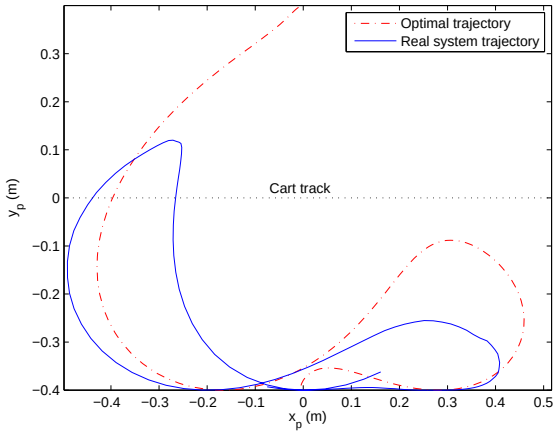

Figure 12: Pendulum end point trajectory for the real system when pendulum is swinging initially and no feedback is used as described in 5.1.

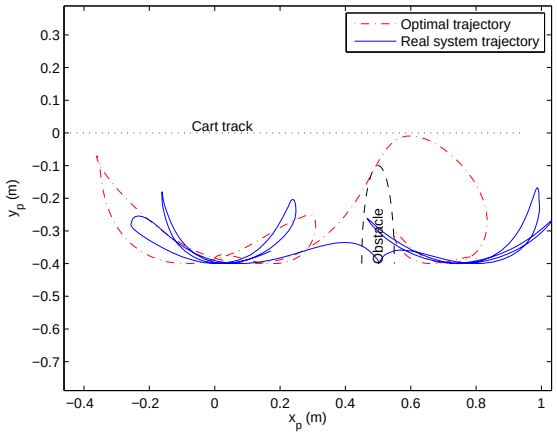

Figure 13: Pendulum end point trajectory for the real system when pendulum is swinging initially and no feedback is used as described in 5.1.

5.2 MPC-Feedback

Model Predictive Control feedback (MPC) is introduced to take care of disturbances to the system. A schematic view of how the feedback is introduced is found in Figure 14 where z and S are defined as in Figure 5. In MPC, a finite time-horizon optimization problem with state and control constraints is solved in every sample. In our setup, deviations from the opti-

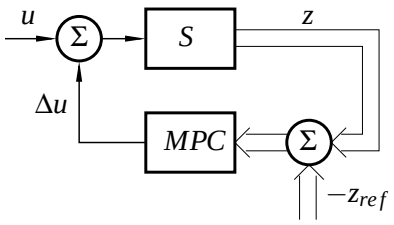

Figure 14: A schematic view of how the MPC-feedback is introduced.

mal state and control trajectories are minimized, such that control magnitude and cart position constraints are not violated.

The mathematical and implementational aspects of the MPC-feedback is beyond the scope of this paper and will be described in a future paper. The results when applying the MPC-feedback to the real system are, however, relevant to show that the optimal feed forward trajectories must be accompanied with feedback to be useful in reality. Experimental results of optimal trajectory feed-forward in combination with MPC-feedback are visualized in Figures 15 and 16. The experiments are performed with initial pendulum swings. Also here the initial swings have a magnitude of around 45° to be comparable to the results in the previous section. Due to the initial swinging, the trajectories are far from the optimal ones in the beginning but the feedback brings the system closer with time. If the feedback control authority is large enough, the original objectives of the optimizations can be achieved despite errors in the initial conditions. The figures show that we have enough control authority in these experiments since we manage to swing-up the pendulum in the first experiment and avoid the obstacle in the second experiment, as desired.

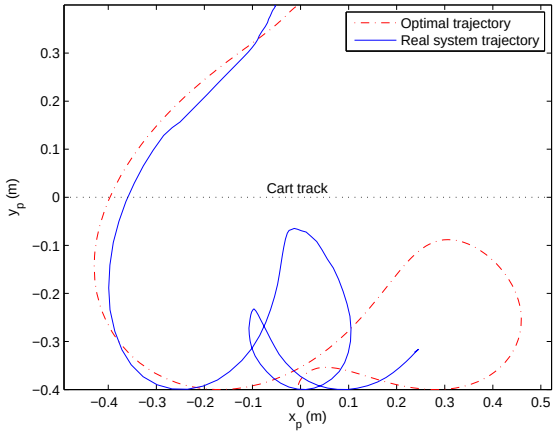

Figure 15: Pendulum end point trajectory for the real system when pendulum is swinging initially and feedback is used as described in 5.2.

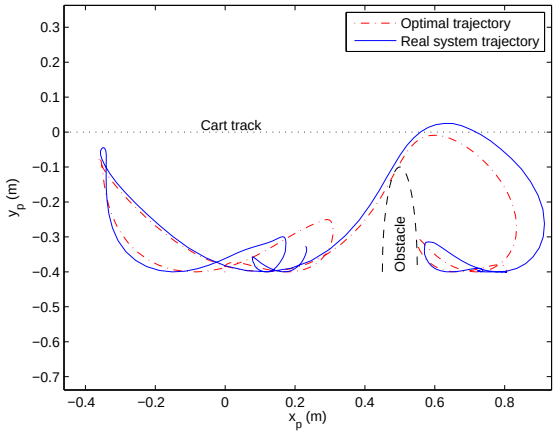

Figure 16: Pendulum end point trajectory for the real system when pendulum is swinging initially and feedback is used as described in 5.2.

6 Teaching

Optimal control of the cart-pendulum system was introduced as a new laboratory exercise in the course on Nonlinear Control and Servo systems (FRTN05) at the Dept. of Automatic Control, Lund, in 2009, see [9].

The cart system has previously been developed as a general module for different control experiments and has been used as a test bed in both student and research projects as well as in other courses [7].

The preliminary evaluation of the new computer and laboratory exercises has been very positive from both the students as well as from the lecturer and the teaching assistants. Optimal control has already before played an important role in the course curriculum, but was mainly focused on the theoretical aspects and lacked from the gap between constrained-low-order-pen-and-paper-problems and more realistic examples and applications. Here Optimica has played an important role to bridge that gap and to complement the previous course contents.

The new software gives the the students the possibility to concentrate on the formulation of the optimal control problem separately from the system modeling and to experimentally evaluate how solutions change with respect to the cost function and to the constraints.

Obtaining a numerical solution naturally raises the question of accuracy, but also to related questions on sensitivity to initial conditions and to discrepancies of the model and the real plant. In the lab exercises this is evaluated where pure feedforward solutions are com-

pared to the combination of a feedforward reference from the optimal control problem together with feedback around this trajectory, similar to what has been outlined in Sections 4 and 5.

7 Conclusions/Future Work

The second optimization problem with path constrained pendulum end point movements shows that optimal control problems can be difficult to solve. Although the Optimica tool is very powerful, one needs to understand the problem and sometimes supply an initial guess to help the solver converging to the correct solution.

The results of the Optimica optimizations are open loop control trajectories. When applying these to a real system, everything must be accurately modeled and only very small disturbances may be present to get good results. This is however rarely the case, which is why we need feedback that controls the actual state trajectories towards the optimal ones. This combination of optimal feedforward and feedback has shown to be very efficient when optimal transitions through nonlinear dynamics are desired.

In this paper, optimal trajectories are pre-calculated using Optimica and MPC-feedback is used to stay close to the optimal trajectories. An extension to this work would be to instead of pre-calculating the optimal trajectories, rather let an MPC-controller run with Optimica in real time. Then the optimization problems stated in (4) and (5) would be solved in each sample with different initial conditions. The initial conditions would be the measured state variables at the current sample. The main difficulty would be to ensure fast enough computations for this to be implementable in a real time application.

References

[1] Johan Åkesson. Optimica—an extension of modelica supporting dynamic optimization. In *In 6th International Modelica Conference 2008*. Modelica Association, March 2008.

[2] Johan Åkesson, Tove Bergdahl, Magnus Gäfvert, and Hubertus Tummescheit. Modeling and optimization with modelica and optimica using the jmodelica.org open source platform. In *Proceedings of the 7th International Modelica Conference 2009*. Modelica Association, September 2009.

[3] Johan Åkesson, Magnus Gäfvert, and Hubertus Tummescheit. Jmodelica—an open source platform for optimization of modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, February 2009. TU Wien.

[4] Karl Johan Åström and Katsuhisa Furuta. Swinging up a pendulum by energy control. *Automatica*, 36:278–285, February 2000.

[5] L.T. Biegler, A.M. Cervantes, and A Wächter. Advances in simultaneous strategies for dynamic optimization. *Chemical Engineering Science*, 57:575–593, 2002.

[6] Python Software Foundation. Python Programming Language – Official Website, 2009. `http://www.python.org/`.

[7] Per-Ola Larsson and Rolf Braun. Construction and control of an educational lab process - the gantry crane. In *Reglermöte 2008, Luleå*, June 2008.

[8] Modelon AB. JModelica Home Page, 2009. `http://www.jmodelica.org`.

[9] Pontus Giselsson. Laboratory Exercise in Nonlinear Control and Servo Systems, 2009. `http://www.control.lth.se/course/FRTN05/labs/lab3/lab3.html`.

[10] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–58, 2006.

[11] Magnus Wiklund, Anders Kristenson, and Karl Johan Åström. A new strategy for swinging up an inverted pendulum. In *Preprints IFAC 12th World Congress*, volume 9, pages 151–154, Sydney, Australia, July 1993.