FRTN15 Predictive Control — Model predictive control of the ball and beam process using CVXGEN

Fredrik Karlsson, Jonas Enerbäck and John Wahnström

December 9, 2012

Abstract

In this paper we discuss the control of a ball and beam system using model predictive control. The control law is given by solving an optimization problem at each sample using a custom made c-solver implemented in simulink.

Introduction

The ball and beam process

The system that is controlled in this project is the classical ball and beam system - a steel ball rolling on the top of a beam. The beam is mounted on the output shaft of an electrical motor which makes it possible to tilt the beam about its center axis. The open loop system is unstable and the control task is to stabilize the ball and be able to change the ball's position on the beam. The process model can be decomposed into two models one for the beam angle(G_ϕ), how the motor current influence the beam angle. And one for the ball position (G_x), how the ball position depend by the beam angle. The total transfer function from the beam current to the ball position is then $G_\phi \cdot G_x$.

Using classical mechanics and small angle approximation one can derive the following transfer function:

$$G_x = -\frac{7}{s^2}$$

To derive a transfer function for the angle process one can assume that the motor current is proportional to the angle velocity. Therefore the following

transfer function is achieved:

$$G_\phi = \frac{4.4}{s}$$

Where 4.4 is the proportionality constant. The total transfer function from the motor current to the position is then given by:

$$G = -\frac{7 \cdot 4.4}{s^3}$$

Convex optimization and CVXGEN

The goal of convex optimization is to minimize a convex function over a convex set. A simple example of a convex optimization problem is finding the minima of a second degree polynomial with only positive coefficients. The convex optimization is in some sense "easier" than the general case, mostly because a local minima is also a global minima. Thus when you find a minimum point it is the minima of the function, also the function will always have a negative slope towards the minima. This makes convex optimization problems ideal for computers.

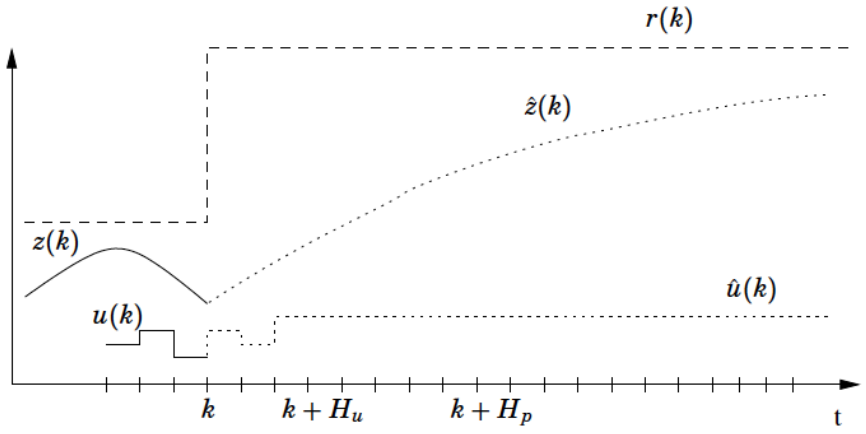
Convex optimization is used in model predictive control to solve for the optimal control signal with weights on the deviation from reference value, rate of change of the control signal or the control signal itself.

CVXGEN is one of the available tools for solving small convex optimization problems. It generates fast custom code for the given problem, thus making in ideal for embedded systems which requires short execution times. One of the applications is model predictive control.

Model predictive control

Model predictive control (MPC) MPC is a control method that predict future input values, $\hat{u}(k)$, by given a prediction of predicted future outputs, $\hat{z}(k)$, see figure 1.

By creating a model of the controlled system and then apply the MPC method on the model, the problem will be solved iterative. The MPC will solve, at each sample, an optimal control problem over a fixed selected interval, known as the prediction horizon, Hp. A too short selected horizon may cause instability but if the predicted horizon is chosen too long the complexity of the calculations will be too high. The deviations that accrue in the controlled variables, $z(k)$, are minimized by a chosen cost function. After these calculations have been done, the MPC only takes out and implement the first control variable as a control signal and waste the rest values in the sequence. Next sample then begins and the procedure is repeated and executed once again but with new measurements data as the initial state.



Figur 1: The idea of MPC (Figure taken from MPCtools 1.0 – Reference Manual, p. 8 figure 1)

Method

Process discretization

Given the transfer functions (G_x and G_ϕ) and the physical model of the process the following continuos time state space model can be formed:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -7 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 4.4 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x. \end{aligned}$$

Using this state space model the first state x_1 becomes the position of the ball, the second x_2 the velocity of the ball and the third x_3 the angel of the beam. The available measurements from the process is position y_1 and angel y_2

Using a sampling frequency of 100 Hz (a sampling time of 0.01 s), the model is discretizised to:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 1 & 0.01 & -0.0004 \\ 0 & 1 & -0.07 \\ 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -0.0015 \\ 0.044 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x. \end{aligned}$$

CVX model

In cvxgen model the prediction horizon, Hp, is decided together with the number of state, n. Then the model is build to minimize the sum of the square of the positions error but the model also allows to minimize the weighted square of the output value, $u(k)$ and square differens between $u(k)$ and $u(k-1)$. Different constraints like the ball position and the angle of the beam are furthermore add to the model.

The CVXGEN code used to generate the solver is shown below

```
dimensions
  n = 3 %Problem dimension
  Hp = 40 %Prediction horizon
end

parameters
  %System matrices
  A (n,n)
  B (n,1)
  C (1,n)
  D (1,n)
  %Weights
  Q nonnegative %Position deviation
  R nonnegative %Output
  S nonnegative %Difference between outputs

  r %Reference value

  x0 (n,1) %Initial values of states

  a %Constraint on position
  b %Constraint on control signal
end

variables
  x[t] (n), t=0..Hp, %Array of states
  u[t], t=0..Hp      %Array of control signal
end

minimize
  sum[t=0..Hp-1](Q*square(r-C*x[t])+R*square(u[t+1]-u[t]) +S
*square(u[t]))
subject to
  x[t+1] == A*x[t]+B*u[t], t=0..Hp-1 %Process model
```

```
  x[0] == x0                          %Initial conditions
  -a <= C*x[t] <= a, t=0..Hp
  -b <= u[t] <= b, t=0..Hp
end
```

Kalman filter

When using model predictive control all states must be possible to compute or measure. Since the process only measures the ball position and the beam angle a kalman filter has been designed to estimate the velocity state. A noise model is constructed such that the process noise and the measurement noise are independent.

$$\begin{aligned}\dot{x} &= \begin{bmatrix} 1 & 0.01 & -0.0004 \\ 0 & 1 & -0.07 \\ 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -0.0015 \\ 0.044 \end{bmatrix} u + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} w \\ y &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} v.\end{aligned}$$

The covariance matrices for the process noise w and measurement noise v are tuned by hand to give an acceptable trade of between rate of convergence and accuracy. The covariance matrices are

$$Q_w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } Q_v = \begin{bmatrix} 20 & 0 \\ 0 & 10 \end{bmatrix}.$$

Simulink model

Figure 2 shows the simulink model that is used both in simulations and on the real process.

Results

Simulation

In figure 3 the result of a simulated step response from 0 to 1 is shown. The settling time is around 75 samples (0.75 s). During this simulation the control signal reached the maximum/minimum values (± 10) for a large part of the step. Also a small overshoot in position is noticed.

Real process

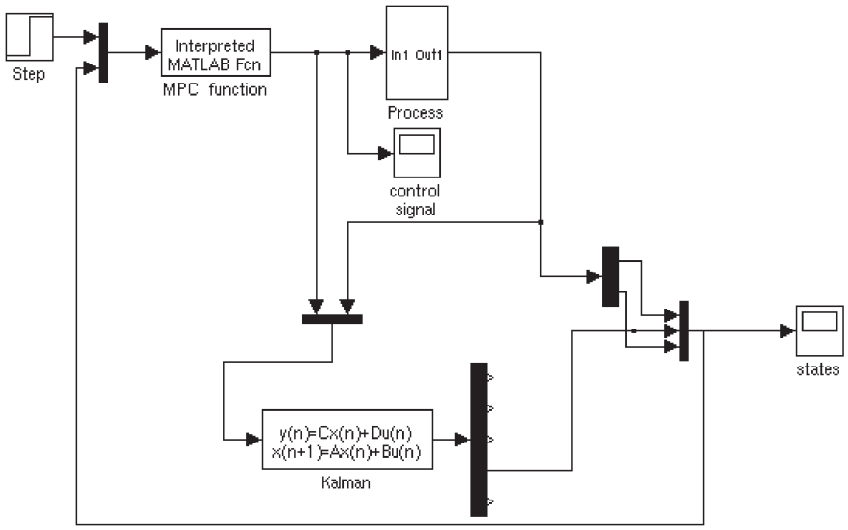
In figure 4 the result of a change in reference value is shown. The settling time is around 500 samples (5 s). The control signal is rather oscillatory, but does not reach the maximum/minimum (± 10). Also the reference signal is rather oscillatory and a stationary error arises.

Discussion and conclusions

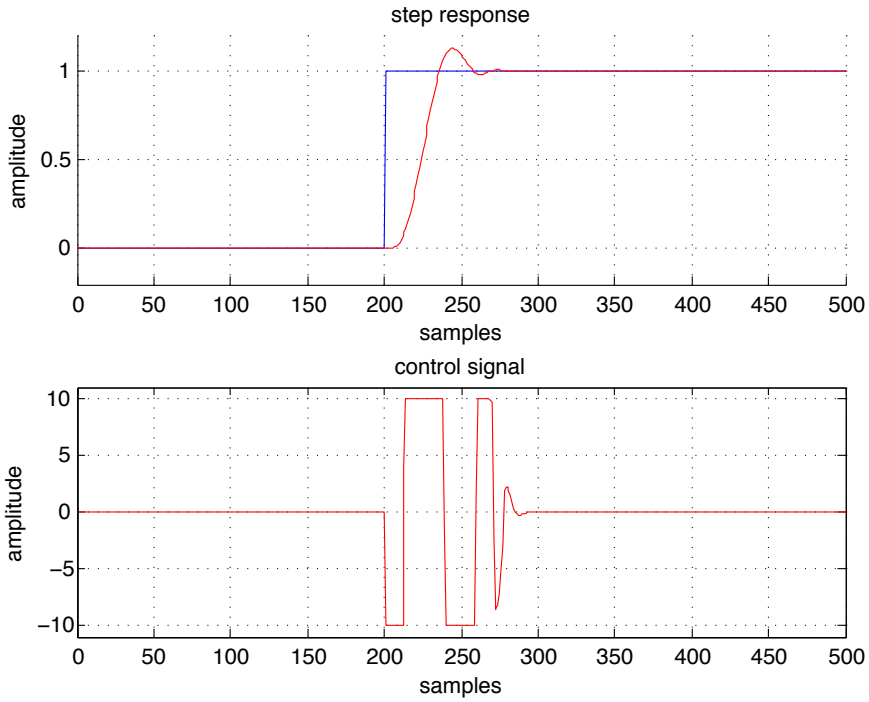
The most important difference between the simulation and the real process is that during the simulation there is no noise in the signal but we have a lot of noise in the real experiments. The velocity state is sensitive to noise since we use a kalman filter to estimate this. The stationary error that is present in the step response could be explained by the noise signals. To get rid of this an integrator state could be introduced. Furthermore the control signal is oscillatory which also could be explained by the noisy signals.

Another topic off error is that the model that we have used is linear, the angles has been approximated to be small. We think that better performance could be achieved if an nonlinear model and nonlinear MPC would have been used.

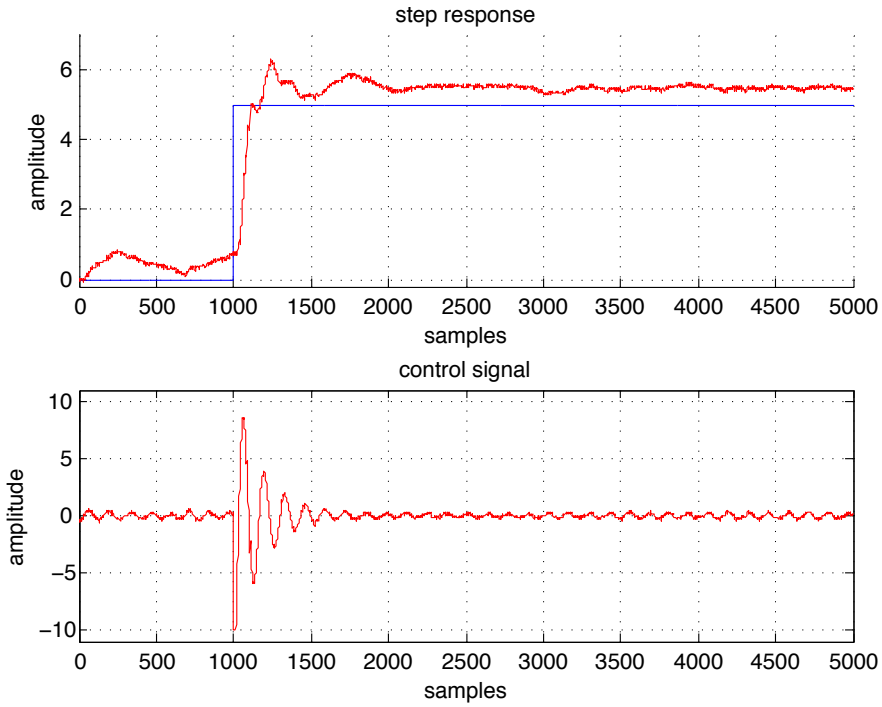
The conclusion drawn from this project is that MPC requires a good model of the process, and model errors limits the control performance. However this model results in a decent controller. Also the MPC is good if the goal is a fast controller but another approach might be preferable if the goal is a robust controller.



Figur 2: The simulink model of controller and process being used.



Figur 3: Results of a simulated change in reference value from 0 to 1, one sample is 0.01 s. The control signal is also shown.



Figur 4: Results of a change in reference value on the real process from 0 to 5, one sample is 0.01 s. The control signal is also shown.