# Predictive Control
# Lecture 10
# More MPC

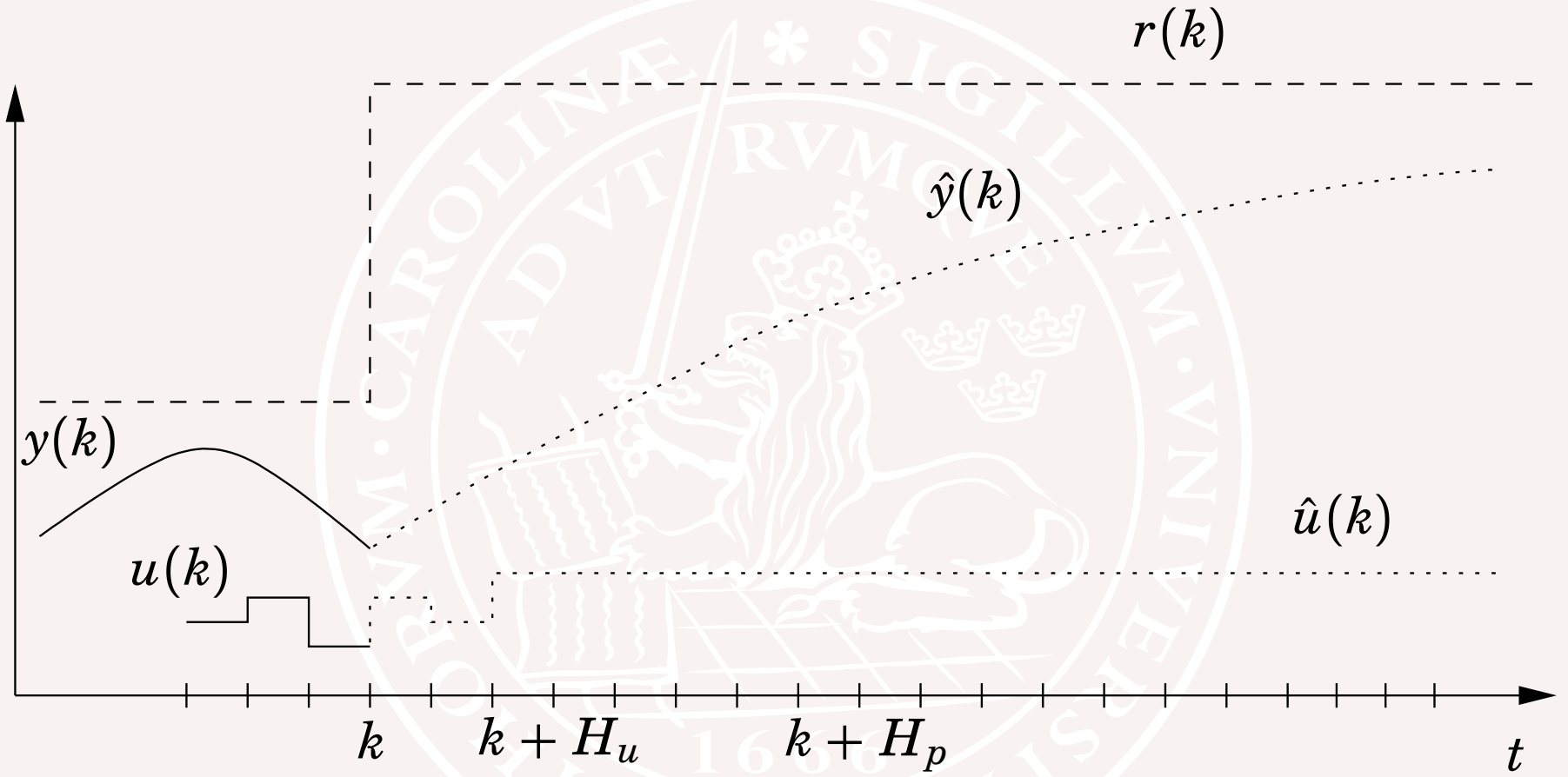
Department Automatic Control
Lund University

# Lecture 10

- MPC Tools - how to use it
- Getting Integral Action
- Example: Quadruple Tank
- Example: A Lab Helicopter
- Project suggestion: CVXGEN
- Example: MPC on a Robot using CVXGEN

Material: Lecture 10: Manual to MPC toolbox (scan)
Matlab code

# MPC Prediction Horizons



# MPC for Linear Systems

Model assumptions in MPC toolbox

$$
\begin{aligned}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= C_y x(k) \\
z(k) &= C_z x(k) + D_z u(k) \\
z_c(k) &= C_c x(k) + D_c u(k)
\end{aligned}
$$

- Measured outputs $y$
- Controlled outputs $z$
- Constrained outputs $z_c$

(no known disturbances)

# An Optimal Control Problem

Cost function

$$J(k) = \sum_{i=H_w}^{H_p+H_w-1} \|\hat{z}(k+i|k) - r(k+i|k)\|_Q^2 + \sum_{i=0}^{H_u-1} \|\Delta u(k+i|k)\|_R^2 \tag{1}$$

- Prediction Horizon, $H_p$
- Control Horizon, $H_u$
- First sample to be included, $H_w$

# Constraints

The MPC controller should respect the constraints

$$\Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max}$$
$$u_{min} \leq u(k) \leq u_{max}$$
$$z_{min} \leq z_c(k) \leq z_{max}$$

Some variables might be constrained, but have no reference values

If a constrained variable is not measured, the constraints will be put on an estimate instead

# Why penalize $\Delta u(k)$ instead of $u(k)$

- Simple to handle $r(k) \neq 0$
  - No need to specify $u_r$
- Still possible to penalize $u(k)$
  - Just include $u$ in $z$-vector: choose $C_z = 0$ and $D_z = I$

# Solving the Quadratic Program (QP)

One can rewrite the optimization criterion on the form

$$\min J(k) = \Delta\mathcal{U}^T \mathcal{H} \Delta\mathcal{U} - \Delta\mathcal{U}^T \mathcal{G} + \mathcal{E}^T Q \mathcal{E}$$
$$\text{subject to } \Omega\Delta\mathcal{U} \leq \omega$$

where $\Delta\mathcal{U}, \mathcal{H}, \mathcal{G}, \mathcal{E}, Q, \Omega, \omega$ are large vectors/matrices, used to stack up the equations above for all time indices.

Details are in the MPC toolbox manual, but will not be needed

Convex problem - efficient algorithms

# State Estimation

Use a traditional Kalman filter, having the form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K(y(k) - C_y\hat{x}(k)).$$

Gain matrix $K$ obtained by solving a Riccati equation

(information about state constraints are typically not utilized, since it is hard to do this)

# Error-free tracking - Integral Action 1

- One option is to use a disturbance observer
- A step disturbance is assumed to act on the input, the following extended model is then used (when $r = 0$):

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \\ d_{k+1} \end{pmatrix} = \begin{pmatrix} A & 0 & B \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_k \\ v_k \\ d_k \end{pmatrix} + \begin{pmatrix} B \\ 0 \\ 0 \end{pmatrix} u_k$$

$$z = y_z = \begin{pmatrix} C_z & 0 & 0 \end{pmatrix} \begin{pmatrix} x_k^T & v_k^T & d_k^T \end{pmatrix}^T$$

$$y_a = \begin{pmatrix} C_a & I & 0 \end{pmatrix} \begin{pmatrix} x_k^T & v_k^T & d_k^T \end{pmatrix}^T$$

- Using an observer with this model structure will introduce integral action giving $z = 0$ in stationarity.

If more outputs than inputs, one must introduce constant output disturbances $v_k$ on the outputs that shouldn't get integral action

If nr inputs = nr outputs one doesn't need $y_a$ and $v_k$

# Error-free tracking - Integral Action 2

Another option is to add integrator states in the model

$$\begin{bmatrix} x(k+1) \\ x_i(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C_z & I \end{bmatrix} x(k) + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(k)$$

$$y(k) = \begin{bmatrix} C_y & 0 \end{bmatrix}$$

$$z(k) = \begin{bmatrix} C_z & 0 \end{bmatrix}$$

A stabilizing feedback is calculated using the extended state. Note that the state $x_i$ need not be estimated, since it is known perfectly by the controller

Polynomial design interpretation to the two methods

$$A(q-1)R' + BS = A_m A_o B^+$$

where either $A_o$ (previous slide) or $A_m$ (this slide) has an increased order compared to the minimal order

# Linear Properties of the MPC Controller

The MPC controller is nonlinear, because of constraints on state and control

However, if the constraints are not active, the controller is linear

The minimizing solutions of the unconstrained QP is then

$$\begin{aligned}\Delta\mathcal{U}(k) &= (\Theta^T Q\Theta + \mathcal{R})^{-1}\Theta^T Q\mathcal{E}(k) \\ &= \dots \\ &= \bar{K}_s \begin{bmatrix} r(k) \\ u(k-1) \\ \hat{x}(k) \end{bmatrix}\end{aligned}$$

for some vector $\bar{K}_s$

# Linear Properties of the MPC Controller

This means the control law can be written

$$\Delta u(k) = \begin{bmatrix} K_{sr} & K_{su} & K_{sx} \end{bmatrix} \begin{bmatrix} r^T(k) & u^T(k-1) & \hat{x}^T(k) \end{bmatrix}^T$$

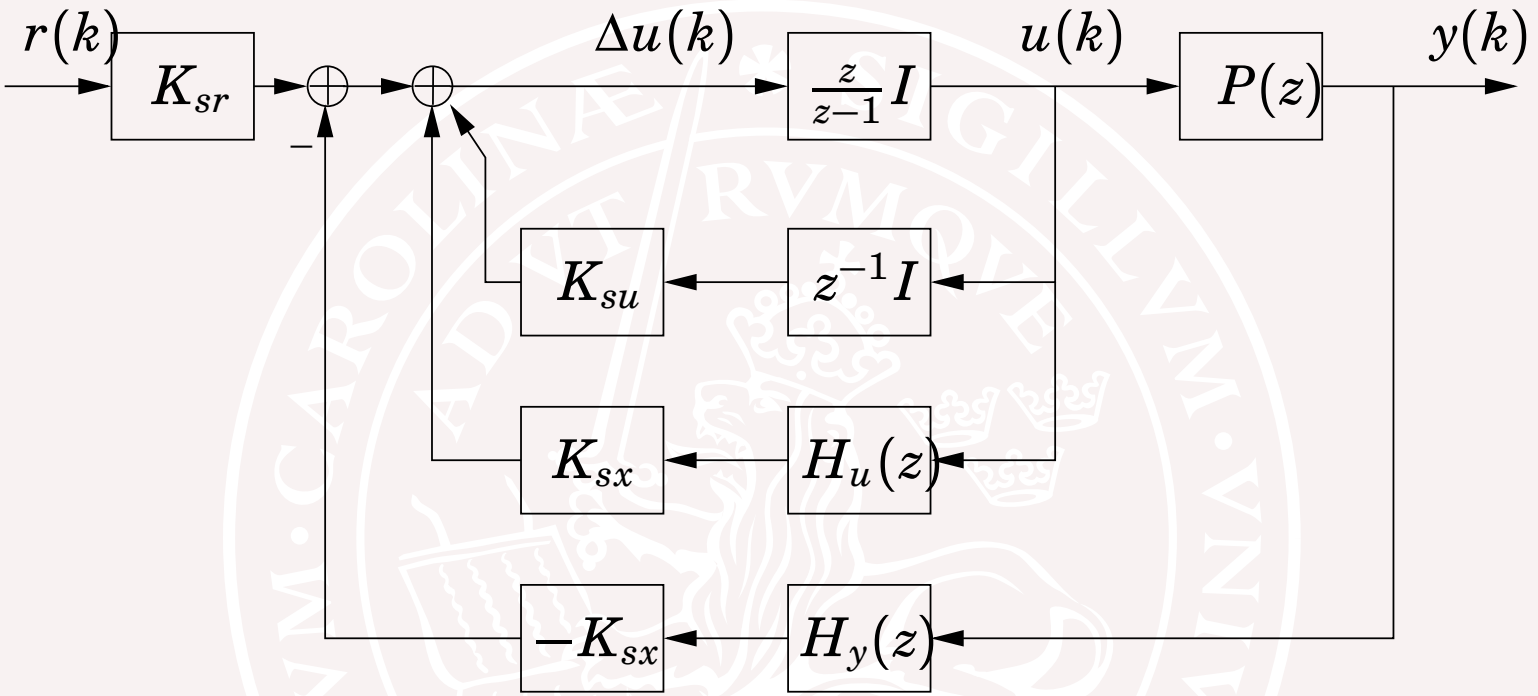
where $\begin{bmatrix} K_{sr} & K_{su} & K_{sx} \end{bmatrix}$ is given by the first $m$ rows of $\bar{K}_s$

This means that with

$$\begin{aligned} P(z) &= C_y(zI - A)^{-1}B \\ H(z) &= -K_{sx}H_y(z) \\ H_y(z) &= (zI - A + KC_y)^{-1}K \\ H_u(z) &= (zI - A + KC_y)^{-1}B \end{aligned}$$

we get the following figure

# Linear Properties of the MPC Controller




Equivalent controller

$$K(z) = \frac{z}{z-1}\left[I - \frac{1}{z-1}K_{su} - \frac{z}{z-1}K_{sx}H_u(z)\right]^{-1}$$

# MPC Tools

MPC controller calculate ,simulate and evaluate in Matlab/Simulink

Good QP solver implementations with active set and interior point methods

Main commands: MPCInit (output: data-structure "md"), MPCSim, MPCController, MPCfrsp

- `Mode 0:` State feedback.
- `Mode 1:` State feedback with explicit integrators.
- `Mode 2:` Observer-based output feedback.
- `Mode 3:` Observer-based output feedback with explicit integrators.
- `Mode 4:` Observer-based output feedback with a disturbance model that gives error free tracking.

# MPC Tools

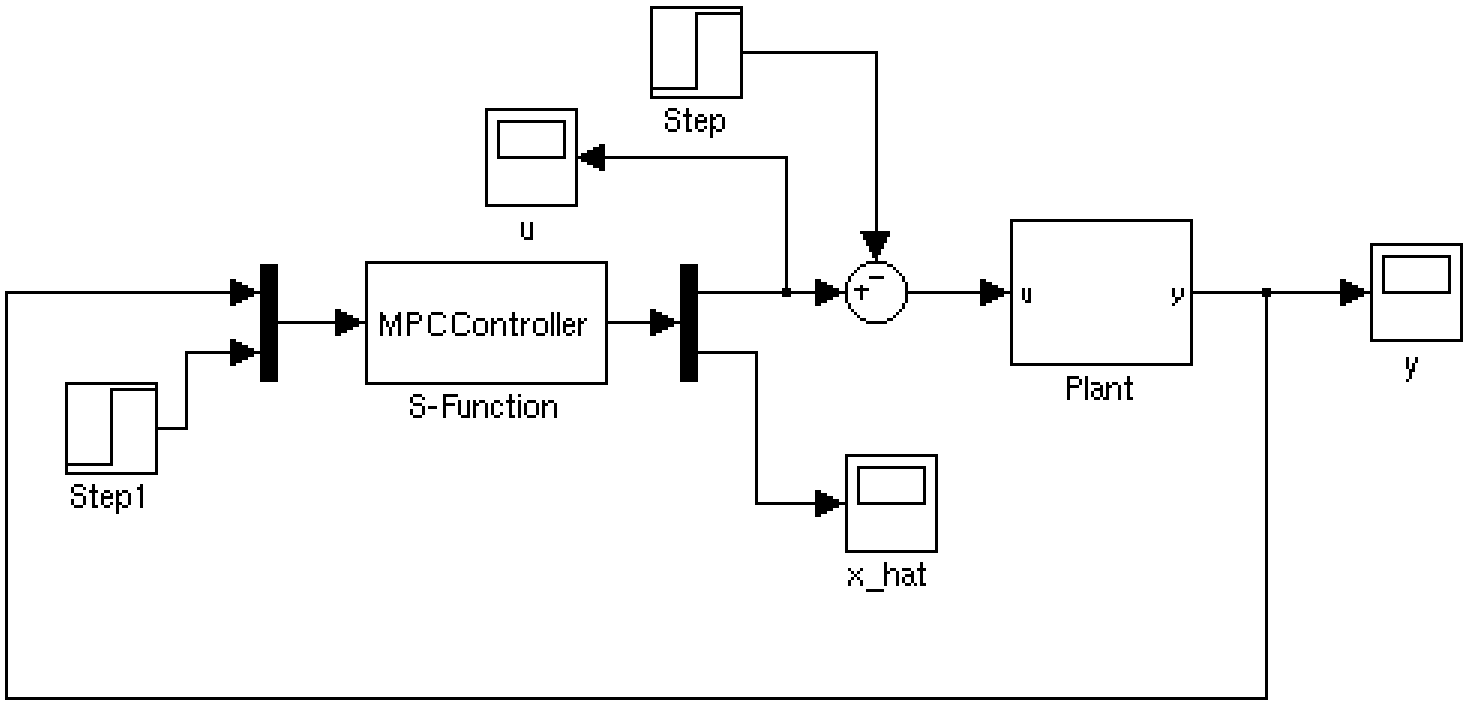
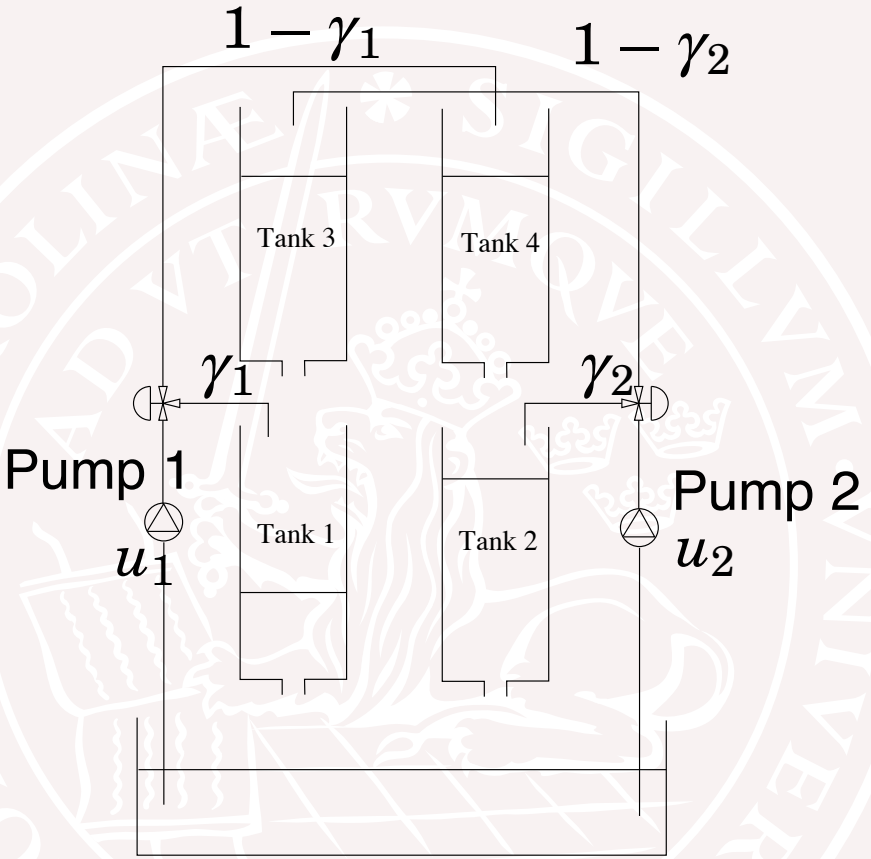



Figure: A Simulink model where the MPC controller is used to control a nonlinear plant.

# Example 1: Quad Tank (=Lab3)




Challenging MIMO process. Parameters $\gamma_1, \gamma_2$ control the flow structure to the upper and lower tanks respectively

Non-minimum phase dynamics if e.g. $\gamma_1 = \gamma_2 = 0.3$

# Nonlinear Dynamics

$$\dot{x}_1 = -\frac{a_1}{A_2}\sqrt{2gx_1} + \frac{a_3}{A_1}\sqrt{2gx_3} + \frac{\gamma_1 k_1}{A_1}u_1$$
$$\dot{x}_2 = -\frac{a_2}{A_2}\sqrt{2gx_2} + \frac{a_4}{A_2}\sqrt{2gx_4} + \frac{\gamma_2 k_2}{A_2}u_2$$
$$\dot{x}_3 = -\frac{a_3}{A_3}\sqrt{2gx_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2$$
$$\dot{x}_4 = -\frac{a_4}{A_4}\sqrt{2gx_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1$$

Linearise around wanted stationary levels.

# Linearized Dynamics

With $\Delta x = x - x^0$, $\Delta u = u - u^0$ and $\Delta y = y - y^0$, we get

$$\Delta \dot{x} = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_4}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} \Delta x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2) k_2}{A_3} \\ \frac{(1-\gamma_1) k_1}{A_4} & 0 \end{bmatrix} \Delta u$$

$$\Delta y = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} \Delta x$$

where

$$T_i = \frac{A_i}{a_i} \sqrt{\frac{2 x_i^0}{g}}.$$

# MPC Controller Parameters

| Parameter | Value |
|---|---|
| $h$ | sampling rate = 3 sec |
| $H_p$ | 30 |
| $H_w$ | 1 |
| $H_u$ | 10 |
| $I_p$ | blocking factor 2 |
| $I_u$ | blocking factor 2 |
| $Q$ | diag(4, 1) |
| $R$ | diag(0.01, 0.01) |
| $W$ | diag(1, 1, 1, 1)/diag(1, 1, 1, 1, 1, 1) |
| $V$ | diag(0.01, 0.01) |

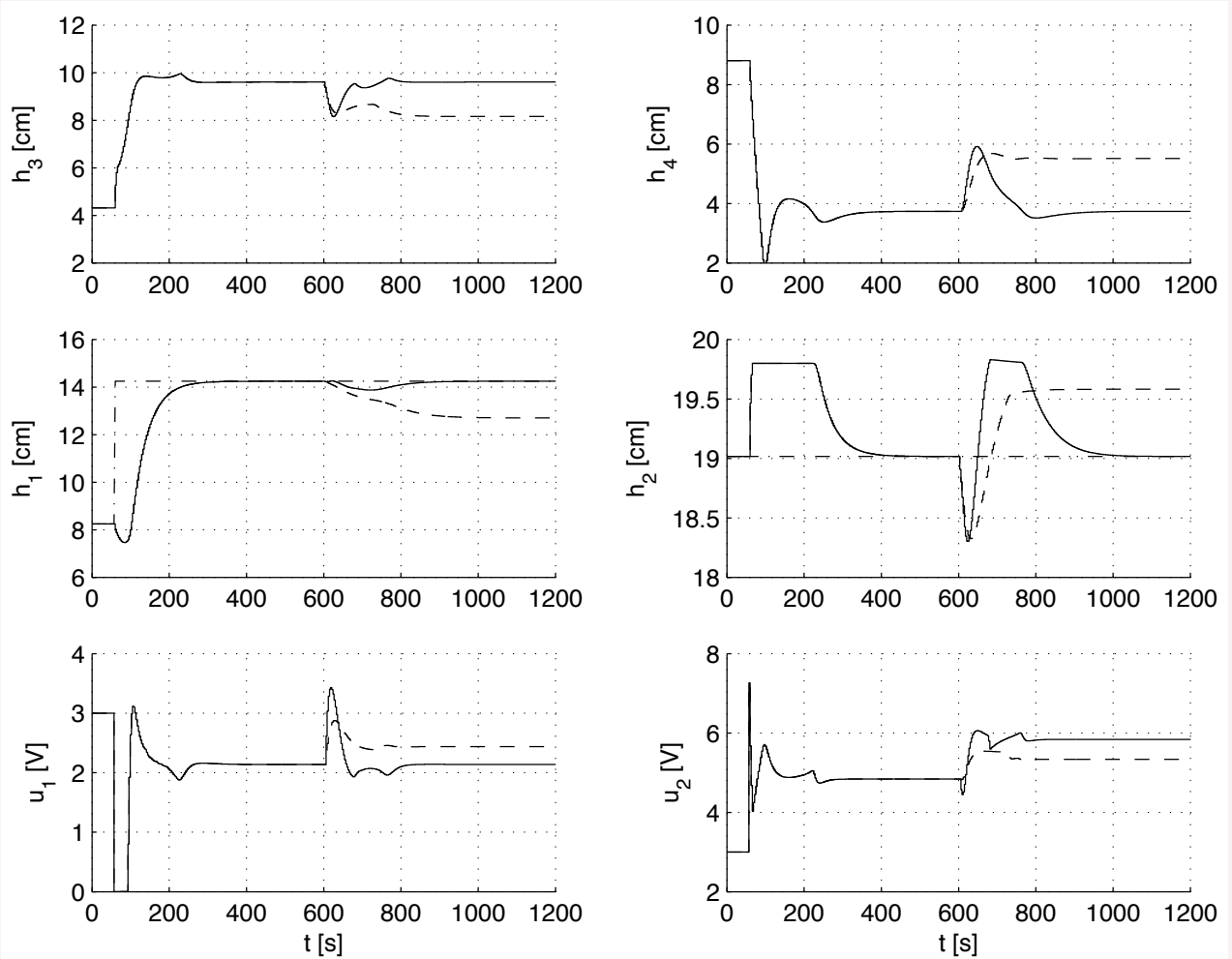
Constraints: $0 \leq x \leq 19.8$ cm on all tanks

$0 \leq u \leq 10$ V on both pumps

# Typical Code

```
% Some initialisation of matrices left out here
Hp = 30; % Prediction horizon
Hu = 10; % Horizon for varying input signal
Hw = 1; % First penalty sample
zblk=2;
ublk=2;
Q = diag([4 1]);
R = 0.01*diag([1 1]);
W = diag([1 1 1 1]);
V = diag(0.01*ones(1,2));
md = MPCInit(Ad,Bd,Cyd,Czd,Dzd,Ccd,Dcd,Hp,Hw,zblk,Hu,ublk,
     du_max,du_min,u_max,u_min,z_max, ...
     z_min,Q,R,W,V,h,2,'qp_as');
MPCfrsp(md,10);
[x,u,y,z,zp,up] = MPCSim(md,s,d);
% Plotting left out here
```

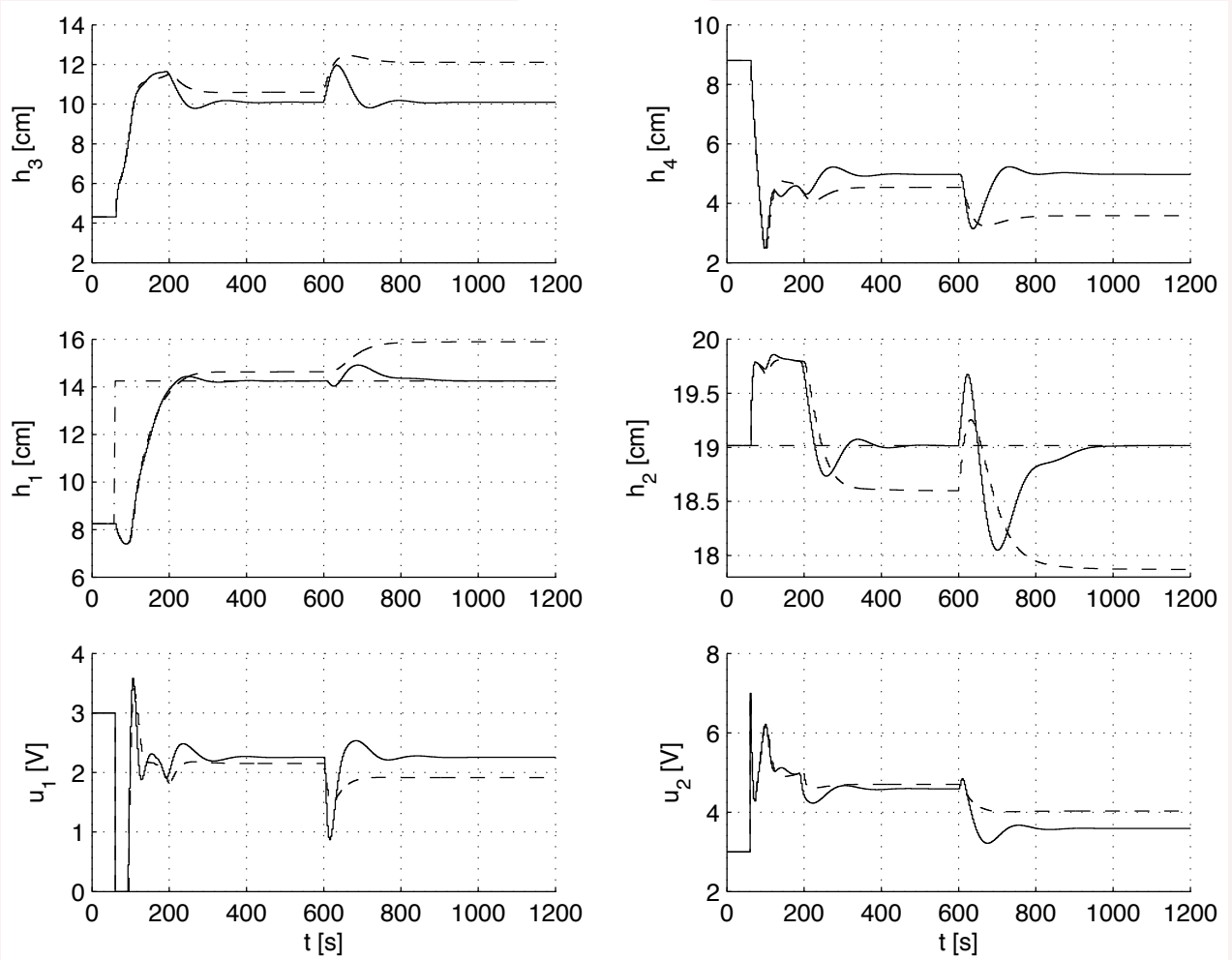
# Results, simulation on linearized plant




Dashed: Kalman filter, no integral action

Solid: Kalman filter with disturbance observer

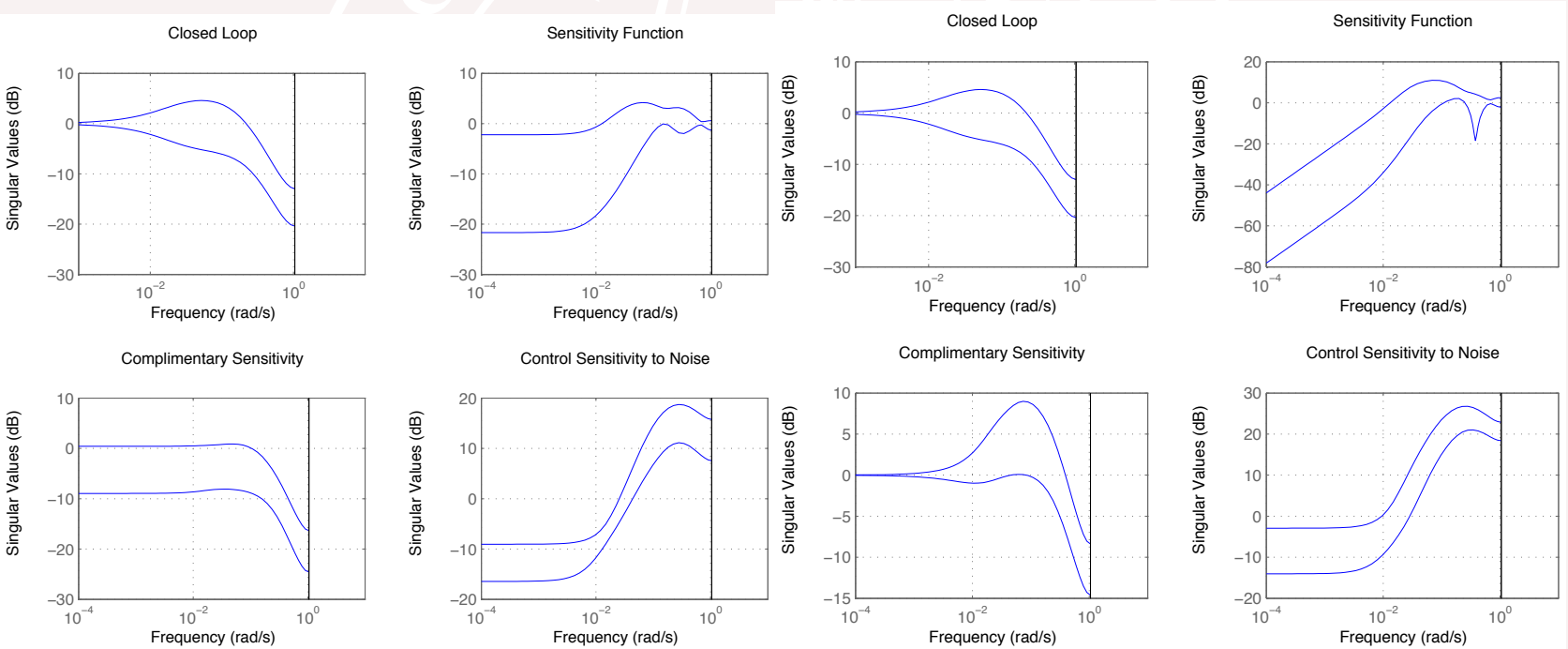
# Results, simulation on nonlinear plant




Works well

# Results, MPC linear behavior

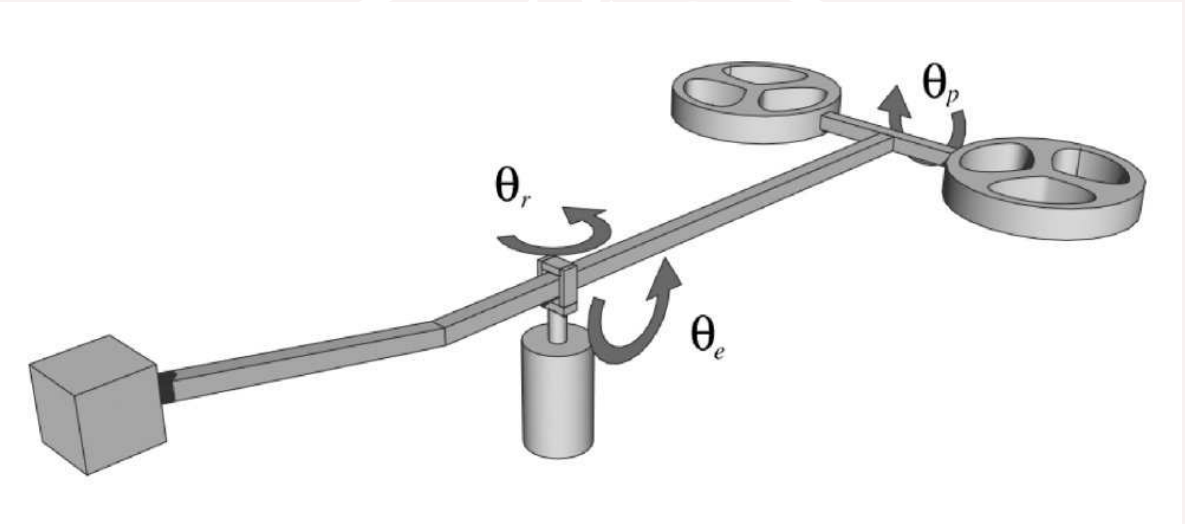
Without the contraints, the linear controller has the following amplitude curves. The plots show the singular values $\sigma_1$ and $\sigma_2$ (the maximal and minimal gains) for the $2 \times 2$ system.




Left: Without integral action

Right: With integral action

# Example 2: Helicopter Process




$$\ddot{\theta}_e = (K_f l_a / J_e)(V_f + V_b) - T_g / J_e$$
$$\ddot{\theta}_r = -(F_g l_a / J_t) \sin \theta_p$$
$$\ddot{\theta}_p = (K_f l_h / J_p)(V_f - V_b)$$

Inputs: Voltages $V_f$, $V_b$ to the propellers

Constraints: $-0.5 \leq \theta_e \leq 0.6, \quad -1 \leq \theta_p \leq 1$

# MPC Parameters

The process was linearized around the stationary point

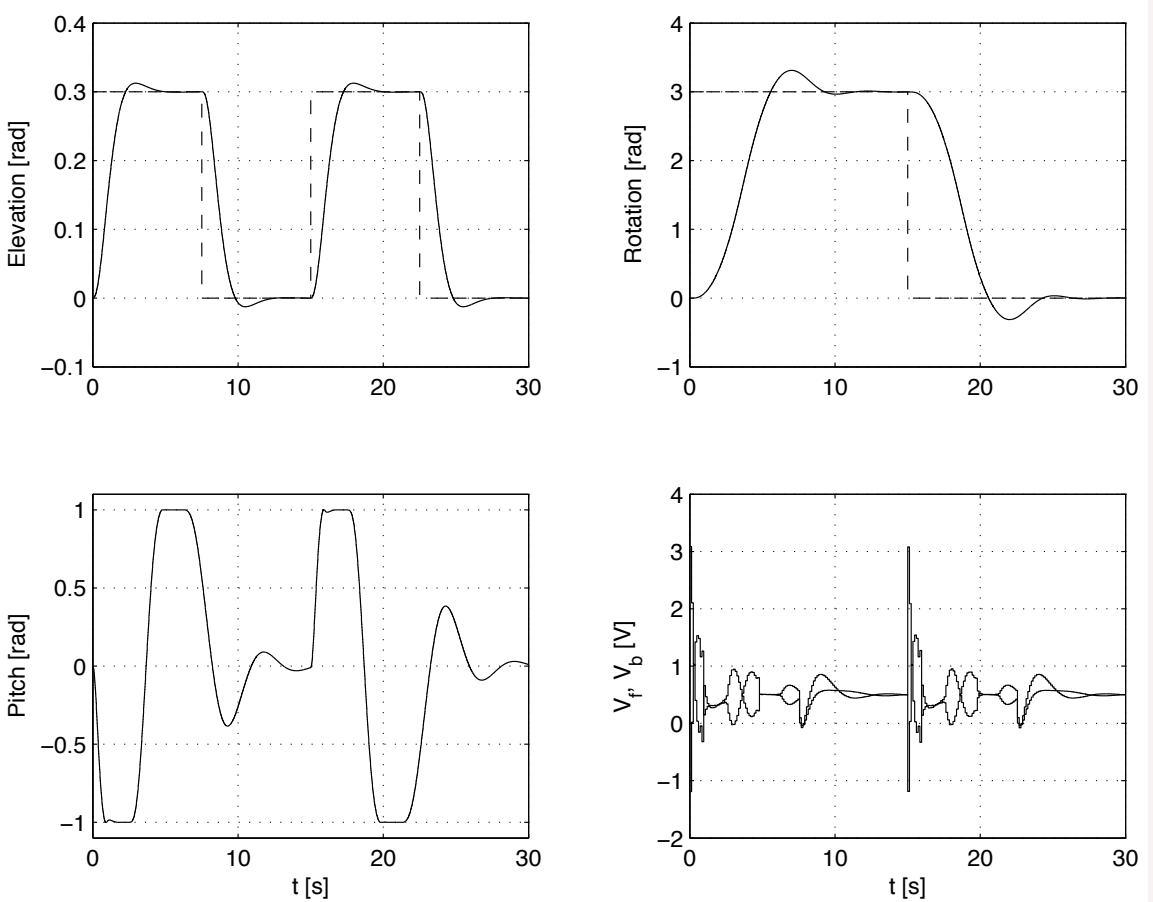
$$\left(\theta_e^0,\ \theta_r^0,\ \theta_p^0,\ V_f^0,\ V_b^0,\right) = \left(0,\ 0,\ 0,\ T_g/(2K_f l_a),\ T_g/(2K_f l_a)\right)$$

and then discretized using the sampling interval $h = 0.2$ s.

| Parameter | Value |
|---|---|
| $H_p$ | 30 |
| $H_w$ | 1 |
| $H_u$ | 10 |
| $I_p$ | Every 2nd sample included in the optimization problem. |
| $I_u$ | Every 2nd control increment assumed to be zero |
| $Q$ | diag(1,1) |
| $R$ | diag(0.1,0.1) |

All states assumed measurable. No integral action

# Simulation - Results




Simulation of the MPC controller applied to the nonlinear helicopter plant.

Reference changes are tracked, pitch and voltages within limits

# Project Suggestion: CVXGEN

Developed at Stanford

CVXGEN generates fast custom code for small, QP-representable convex optimization problems, using an online interface with no software installation. With minimal effort, turn a mathematical problem description into a high speed solver.

See http://cvxgen.com/

# CVXGEN

- Describe your small, quadratic program (QP) representable problem with a simple, powerful language.
- CVXGEN automatically creates library-free C code for a custom, high-speed solver. This can be downloaded and used immediately, and requires nothing but a C compiler. CVXGEN also supplies a Matlab function that, with one command, downloads and builds a custom Matlab mex solver.
- CVXGEN performs most transformations and optimizations offline, to make online solution as fast as possible. Code generation takes a few seconds or minutes, producing solvers that work in microseconds or milliseconds. Compared with generic code (CVX), solution times are typically at least 20 times faster, with the smallest problems showing speedup as large as 10,000x

# CVXGEN: MPC Example

## Optimization problem

We will model the optimization problem

$$
\begin{array}{ll}
\text{minimize} & \sum_{t=0}^{T} \left(x_t^T Q x_t + u_t^T R u_t\right) + x_{T+1}^T Q_{\text{final}} x_{T+1} \\
\text{subject to} & x_{t+1} = A x_t + B u_t, \quad t = 0, \ldots, T \\
& |u_t| \leq u_{\max}, \quad t = 0, \ldots, T \\
& \|u_{t+1} - u_t\|_\infty \leq S, \quad t = 0, \ldots, T-1
\end{array}
$$

with optimization variables

- $x_1, \ldots, x_{T+1} \in \mathbf{R}^n$ (state variables)
- $u_0, \ldots, u_T \in \mathbf{R}^m$ (input variables)

# CVXGEN: MPC Example - Input Code

```
dimensions
  m = 2  # inputs.
  n = 5  # states.
  T = 10  # horizon.
end

parameters
  A (n,n)  # dynamics matrix.
  B (n,m)  # transfer matrix.
  Q (n,n) psd  # state cost.
  Q_final (n,n) psd  # final state cost.
  R (m,m) psd  # input cost.
  x[0] (n)  # initial state.
  u_max nonnegative  # amplitude limit.
  S nonnegative  # slew rate limit.
end

variables
  x[t] (n), t=1..T+1  # state.
  u[t] (m), t=0..T  # input.
end

minimize
  sum[t=0..T]
(quad(x[t], Q) + quad(u[t], R)) + quad(x[T+1], Q_final)
subject to
  x[t+1] == A*x[t] + B*u[t], t=0..T  # dynamics constraints.
  abs(u[t]) <= u_max, t=0..T  # maximum input box constraint.
  norminf(u[t+1] - u[t]) <= S, t=0..T-1  # slew rate constraint.
end
```

# Robot control using CVXGEN

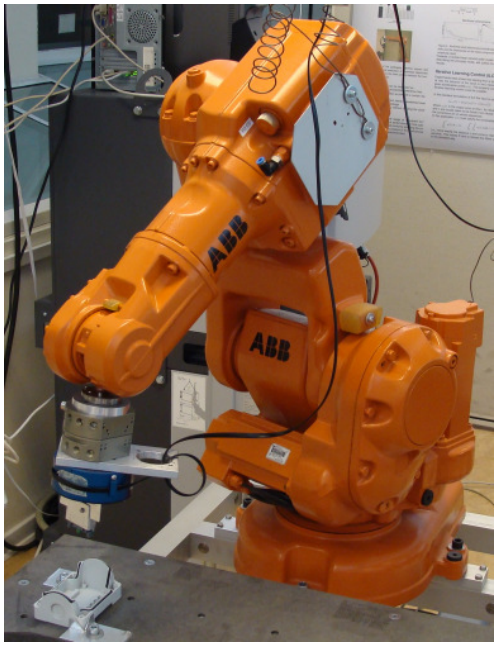
**Andreas Stolt**

Dept. of Automatic Control
LTH, Lund University

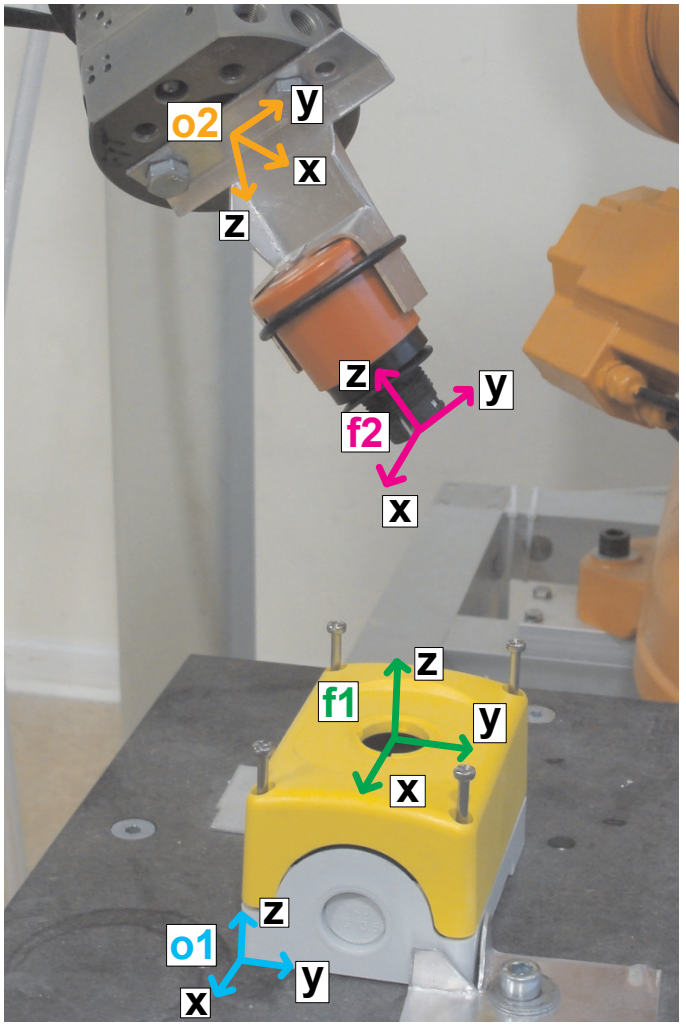
May 28th, 2011

# Introduction

- Industrial robots is basically a chain of links and actuated joints.
- The control problem is to make the end-effector follow a desired trajectory.
- Inconvenient to specify the trajectory in joint space.
  - Redundancy if the robot has extra degrees of freedom or the task is not fully constrained.




# Video




Task specification.



# Experimental conditions

- Position and velocity references are sent to the robot at 250 Hz.
  - A solution is needed every 4 ms.
- A previous attempt to solve this problem was to use IPOPT together with CasADi
  - To long computation time introduced time delays (very inconvenient when in contact!)
- In this project CVXGEN is considered.

# The problem considered

$$
\begin{array}{ll}
\text{minimize (over } \dot{q}) & f_0(\dot{q}) \\
\text{subject to} & \dot{y}_d^0 = A\dot{q} \\
& \dot{q}_{min} \leq \dot{q} \leq \dot{q}_{max}
\end{array}
$$

where one of the following objectives has been used

$$f_0(\dot{q}) = \dot{q}^T M \dot{q}\,, \quad f_0(\dot{q}) = \|M\dot{q}\|_1\,, \quad f_0(\dot{q}) = \|M\dot{q}\|_\infty$$

# Experiments

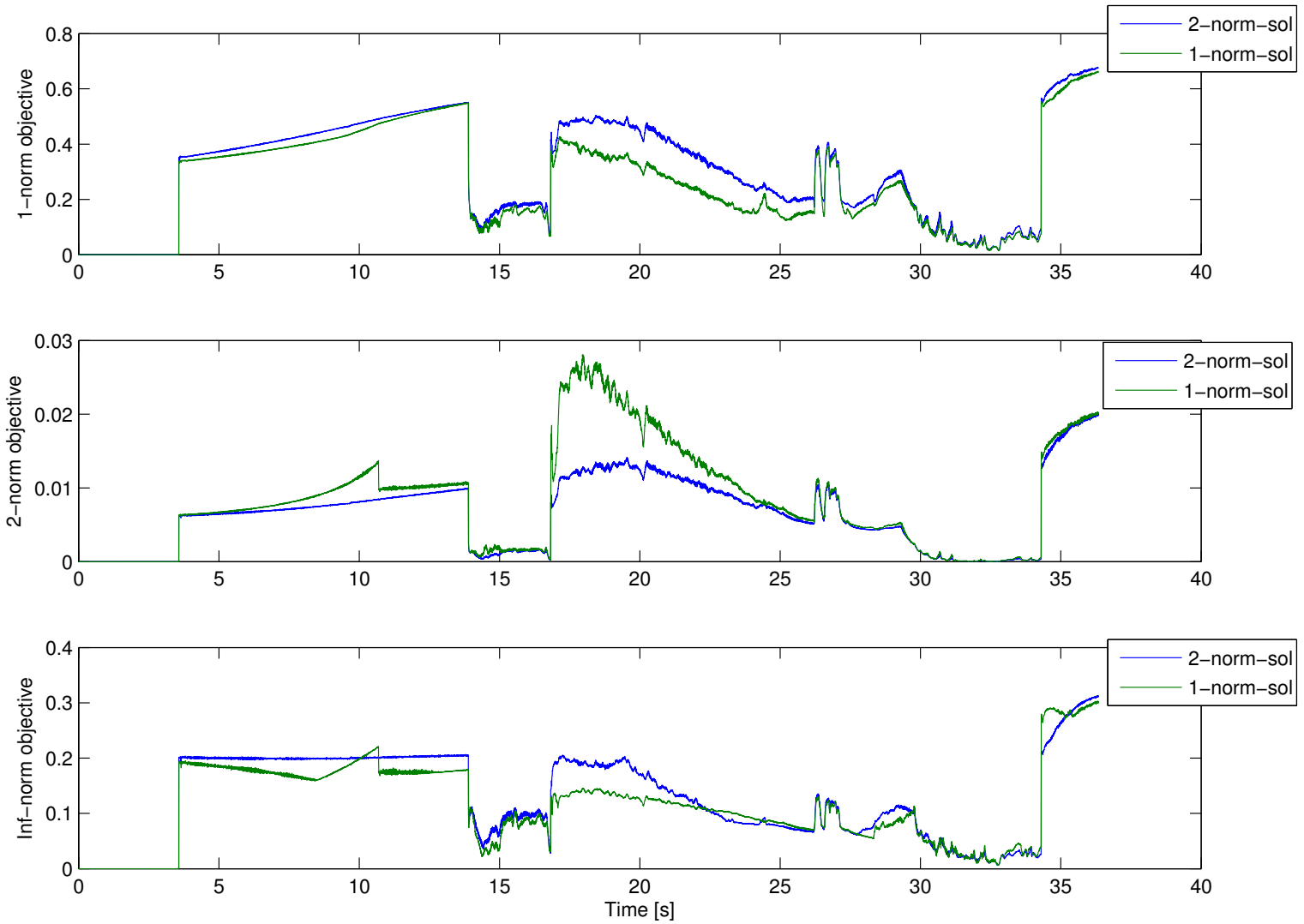
- The generated solver communicates with the robot control program via an ethernet connection.
- Unavoidable to introduce at least one sample delay.
- A way to avoid the delay is to calculate a Jacobian matrix for the neighborhood of the solution (by perturbing $\dot{y}_d^0$ and solving the problem multiple times)

$$\dot{q} = \dot{q}_{opt} + \boldsymbol{J} \cdot \left( \dot{y}_{d,actual}^0 - \dot{y}_d^0 \right)$$

# Results

- The mean computation time is around 0.16 ms (1.09 ms worst case), and one sample delay is introduced. However, no performace degradation has been observed.
- Using the approach with a Jacobian increases the computation time to around 0.50 ms (2.03 ms worst case), but this removes the delay.
  - Some ”spikes” in the calculated Jacobian
  - Due to non-smooth objective function?

# Results


1−norm objective
2−norm objective
Inf−norm objective
Time [s]
2−norm−sol
1−norm−sol