Model Predictive Control (MPC)

Bo Bernhardsson and Karl Johan Åström

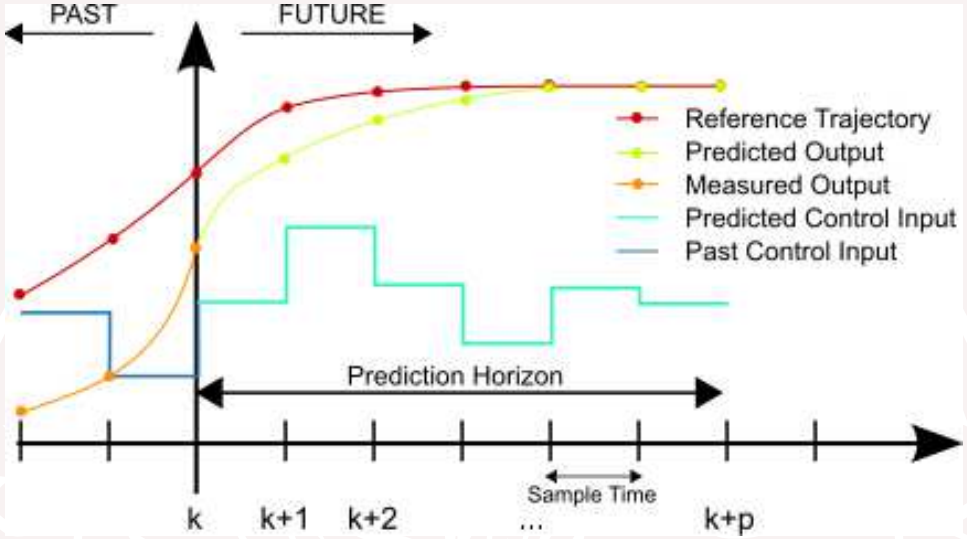
Department of Automatic Control LTH,
Lund University

Model Predictive Control (MPC)

- MPC Problem Setup and Parameters
- Optimization Problem
- MPC Tools
- How to get Integral Action
- Example - Quad Tank
- Explicit MPC and CVXGEN

Material:
Rawlings (2000), Tutorial overview of model predictive control
Åkesson (2006), Manual to MPC tools 1.0

Basic Idea of MPC: Receding Horizon Control



1. At time k solve an open loop optimal control problem over a predefined horizon and apply the first input
2. At time $k+1$ repeat the same procedure (the previous optimal solution can be used as initial guess)

Bellman's principle of optimality

Model Predictive Control

Model Predictive Control (MPC)

- Uses models explicitly to predict future plant behaviour
- Constraints on inputs, outputs, and states are respected
- Control sequence is determined by solving an (often convex) optimization problem each sample
- Combined with state estimation

MPC

- Refineries (Shell in 1970s, "Dynamic Matrix Control")
- (Bellman), Propoi (1963), Richalet, Prett, Morari, . . .
- Main reasons for success
  - Suitable for multivariable systems
  - Respects actuator limitations: no integrator wind-up, saturation problems handled explicitly
  - Process can be run close to constraints, which is often cost-effective

MPC Optimization Criterion

Control changes $\Delta u(k+i) = u(k+i) - u(k+i-1)$ are optimization variables

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

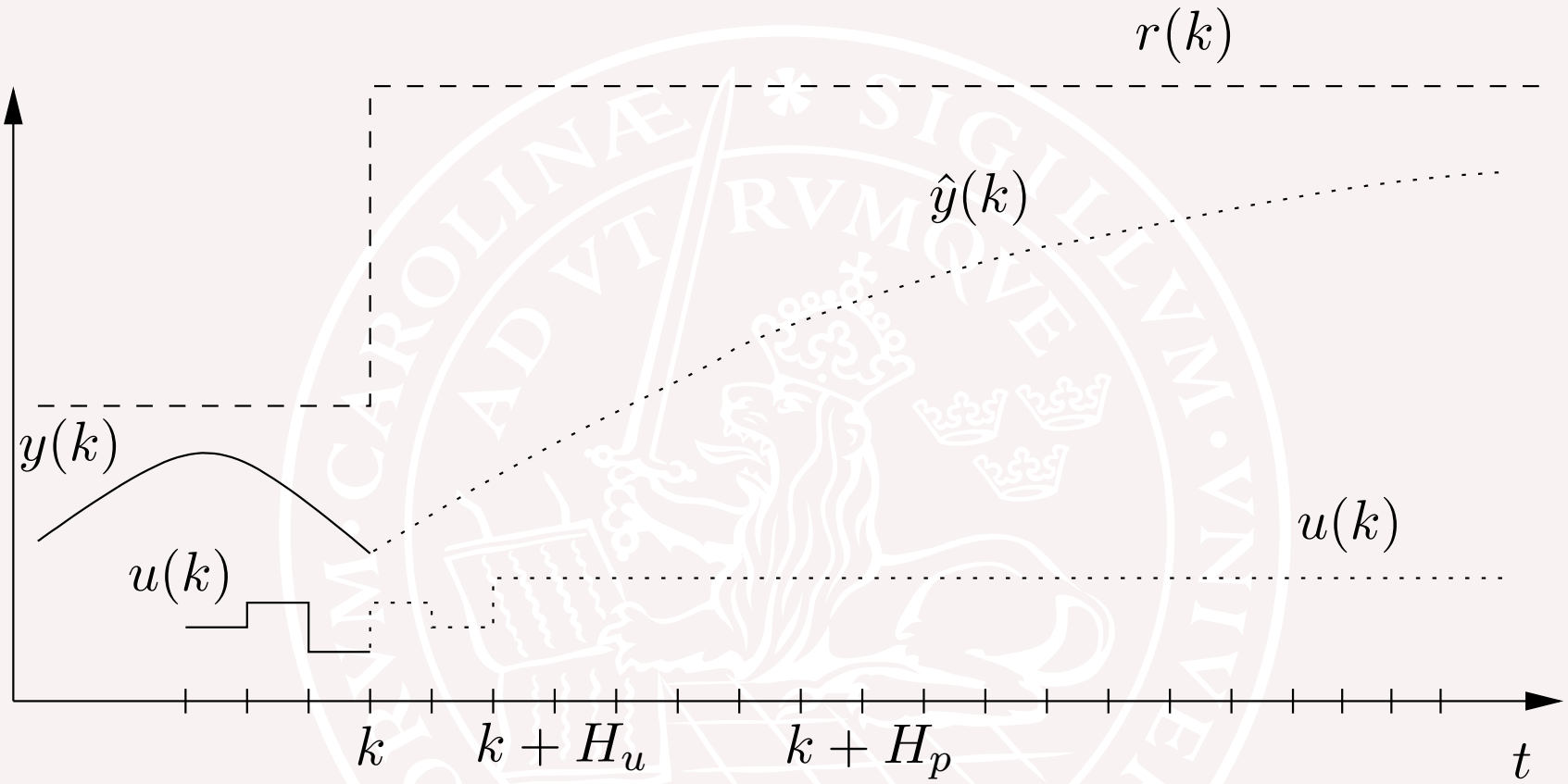
Evaluated signals $z = C_z x + D_z u$

H_w: system time-delay

H_u: cover typical closed loop time constant

H_p: somewhat larger than H_u

MPC Prediction Horizons (with $H_w = 0$)



Large H_u and H_p give better performance but more computations and numerical problems

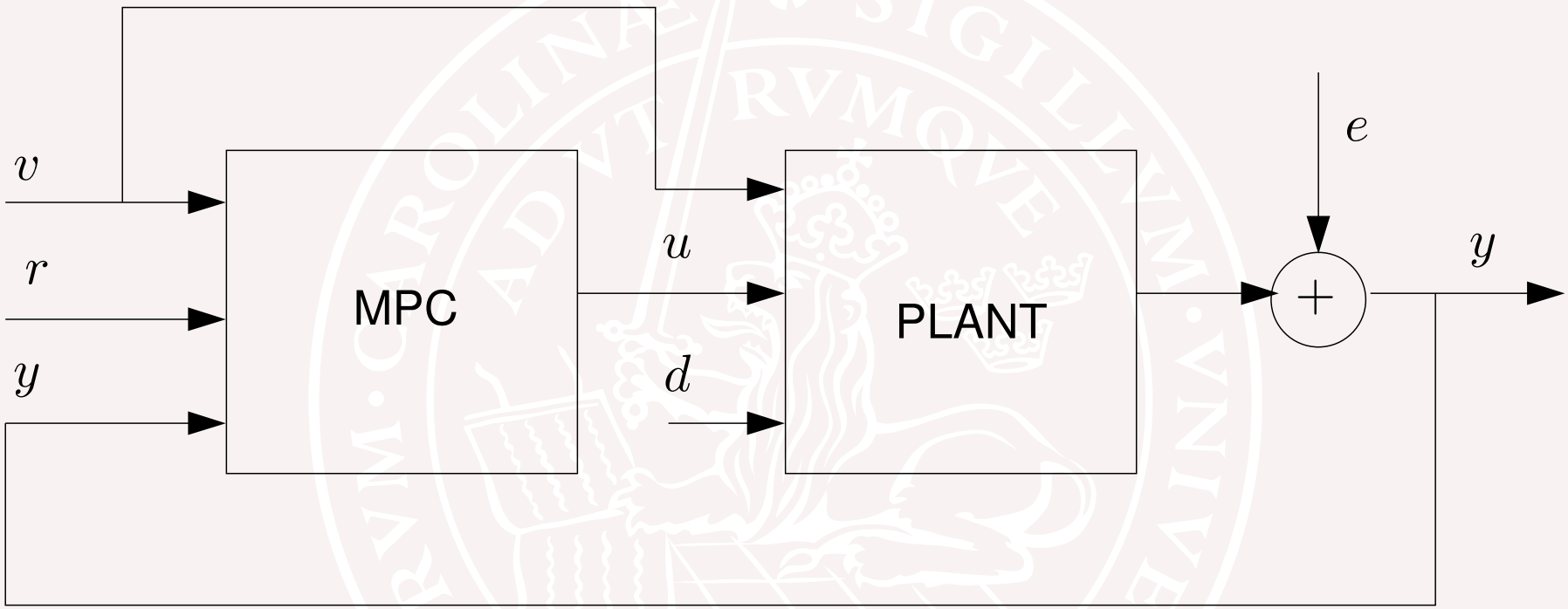
$\Delta u(k)$ vs $u(k)$

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

- Reference $r \neq 0$ requires $u \neq 0$ to avoid static error
  - No need to guess and specify u_r
- Still possible to penalize $u(k)$, just include u in z-vector

TAT: Do we get integral action by penalizing Δu instead of u?
Consider $y = u + d$, do we get $y = r$ in stationarity if $d \neq 0$?

MPC Block Diagram (typical)



v, d, e disturbances, v available for feedforward

r reference value

Assumed Dynamics

$$
\begin{aligned}
x_{k+1} &= Ax_k + B_u u_k + B_v v_k + B_w w_k \\
y_k &= Cx_k + D_v v_k + D_w w_k
\end{aligned}
$$

where v_k is a measured disturbance and $w_k = \begin{bmatrix} d_k \\ e_k \end{bmatrix}$.

MPC Prediction Algorithms

Future trajectories starting at time k over the prediction horizon H_p

Assuming v is not known in advance

$$y_{k+H_p|k} = C\left(A^{H_p}x_k + \sum_{j=0}^{H_p-1} A^{H_p-1}B(u_{k-1} + \sum_{i=0}^{j}\Delta u_i)\right) + D_v v_{H_p}$$

MPC Prediction Algorithms

All H_p predicted time steps can be summarized as

$$\begin{pmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+H_p} \end{pmatrix} = S_x x_k + S_{u-1} u_{k-1} + S_u \begin{pmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_p-1} \end{pmatrix} + H_v \begin{pmatrix} v_k \\ v_{k+1} \\ \vdots \\ v_{k+H_p} \end{pmatrix}$$

MPC Prediction Algorithms

...where

$$S_x = \begin{pmatrix} CA \\ CA^2 \\ \vdots \\ CA^{H_p} \end{pmatrix} \in \mathbb{R}^{H_p n_y \times n_x}$$

$$S_{u-1} = \begin{pmatrix} CB_u \\ CAB_u + CB_u \\ \sum_{j=0}^{H_p-1} CA^j B_u \end{pmatrix} \in \mathbb{R}^{H_p n_x \times n_u}$$

MPC Prediction Algorithms

... and

$$S_u = \begin{pmatrix} CB_u & 0 & \dots & 0 \\ CB_u + CAB_u & CB_u & \dots & 0 \\ \vdots & & \ddots & \vdots \\ \sum_{j=0}^{H_p-1} CA^j B_u & \sum_{j=0}^{H_p-1} CA^j B_u & \dots & CB_u \end{pmatrix} \in \mathbb{R}^{H_p n_y \times H_p n_u}$$

$$H_v = \begin{pmatrix} CB_v & D & 0 & \dots & 0 \\ CAB_v & CB_v & D & & 0 \\ \vdots & & & \ddots & \vdots \\ CA^{H_p-1}B_v & CA^{H_p-2}B_v & CA^{H_p-3}B_v & \dots & D \end{pmatrix} \in \mathbb{R}^{H_p n_y \times (H_p+1) n_v}$$

MPC Blocking Factors

- At each step the following cost function is minimized

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

- Blocking factors can be used to ease the computational requirements. This means that constraints and cost is only evaluated at certain time steps, contained in sets I_p and I_u.

$$J(k) = \sum_{i \in I_p} ||\hat{y}(k+i|k) - r(k+i|k)||_Q^2 + \sum_{i \in I_u} ||\Delta u(k+i|k)||_R^2$$

- Move blocking restrictions can also be imposed on the control signal, so that $\Delta u = 0$ at certain time steps

MPC Optimization Variables

Introduce the optimization variables, if we have reference control signals u_k^r

$$e_u = \begin{pmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+H_p-1} \end{pmatrix} - \begin{pmatrix} u_k^r \\ u_{k+1}^r \\ \vdots \\ u_{k+H_p-1}^r \end{pmatrix},$$

and

$$e_{\Delta u} = \begin{pmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u-1} \end{pmatrix}, \quad e_y = \begin{pmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+H_p} \end{pmatrix} - \begin{pmatrix} r_{k+1} \\ r_{k+2} \\ \vdots \\ r_{k+H_p} \end{pmatrix}$$

for control moves and reference tracking

MPC Optimization Variables

For unconstrained optimization, the cost function is

$$J(z) = e_u^T W_u^2 e_u + e_{\Delta u}^T W_{\Delta u}^2 e_{\Delta u} + e_y^T W_y^2 e_y$$

where the first term penalizes control signals, the second penalizes control moves and the last term penalizes tracking error.

Matrices W_u, $W_{\Delta u}$, and W_y are design parameters.

Alternative notation (here without e_u)

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

Matrices Q and R are (possibly time-varying) design parameters.

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

Soft or Hard Constraints?

Always need to find a u. Problem if constraints unfeasible.

Replace hard constraints with soft constraints.

Minimize slack variable $\epsilon \geq 0$ so that

$$\begin{pmatrix} y^{min}_{k+1} \\ \vdots \\ y^{min}_{k+H_p} \\ u^{min}_{k+1} \\ \vdots \\ u^{min}_{k+H_p} \\ \Delta u^{min}_{k} \\ \vdots \\ \Delta u^{min}_{k+H_u-1} \end{pmatrix} -\epsilon V^{min} \leq \begin{pmatrix} y_{k+1} \\ \vdots \\ y_{k+H_p} \\ u_k \\ \vdots \\ u_{k+H_p-1} \\ \Delta u_k \\ \vdots \\ \Delta u_{k+H_u-1} \end{pmatrix} \leq \begin{pmatrix} y^{max}_{k+1} \\ \vdots \\ y^{max}_{k+H_p} \\ u^{max}_{k+1} \\ \vdots \\ u^{max}_{k+H_p} \\ \Delta u^{max}_{k} \\ \vdots \\ \Delta u^{max}_{k+H_u-1} \end{pmatrix} +\epsilon V^{max}$$

Relaxation vectors $V^{min} > 0, V^{max} > 0$ are user defined parameters. Larger V, more relaxed constraint

MPC Optimization Constraints

For the constrained case, a suitable MPC cost criterion to be minimized is

$$J(z,\epsilon) = e_u^T W_u^2 e_u + e_{\Delta u}^T W_{\Delta u}^2 e_{\Delta u} + e_y^T W_y^2 e_y + \rho_\epsilon \epsilon^2$$

subject to dynamics and constraints

A large value of ρ_ϵ penalizes constraint violation harder.

In practice, softening of output constraints is often a good idea.

MPC Design Parameters

- Internal Dynamic Model
- (State Estimation Parameters)
- Prediction- and Control Horizons
- Blocking factors
- Weighting Matrices
- Constraint Parameters
- Sample rate
- Integral action? alt. disturbance model

Quadratic Optimization

- The optimization problem is often posed as quadratic programming

$$\min_x \quad \frac{1}{2}x^T H x + f^T x$$
$$\text{subject to } Ax \leq b$$

- In MATLAB: 'quadprog'

Advantages with QP: Fast, reliable software

Disturbance Models

- When solving the optimization problem, predicted future values of the disturbances are needed
- With some knowledge about the nature of the disturbances the prediction can be improved
- Often Gaussian noise and Kalman filter is used
- Can also formulate the prediction problem as a QP problem, e.g. maximizing log-likelihood of measurements
- Optimization approach to estimation can e.g. give robustness to outliers

State Estimation - Error Update

$$\begin{pmatrix} \hat{x}^0_{k|k} \\ \hat{x}^d_{k|k} \\ \hat{x}^m_{k|k} \end{pmatrix} = \begin{pmatrix} \hat{x}^0_{k|k-1} \\ \hat{x}^d_{k|k-1} \\ \hat{x}^m_{k|k-1} \end{pmatrix} + K\left(y^m_k - \hat{y}^m_k\right)$$

d : known disturbance

m : reference model

MPC Internal Dynamic Model

$$\begin{pmatrix} x^0_{k+1} \\ x^d_{k+1} \\ x^m_{k+1} \end{pmatrix} = \begin{pmatrix} A_0 & B_0C_d & 0 \\ 0 & A_d & 0 \\ 0 & 0 & A_m \end{pmatrix} \begin{pmatrix} x^0_k \\ x^d_k \\ x^m_k \end{pmatrix} + \begin{pmatrix} B_u \\ 0 \\ 0 \end{pmatrix} u_k$$

$$+ \begin{pmatrix} B_v \\ 0 \\ 0 \end{pmatrix} v_k + \begin{pmatrix} B_0D_d & 0 & B_u & B_v \\ B_d & 0 & 0 & 0 \\ 0 & B_m & 0 & 0 \end{pmatrix} \begin{pmatrix} w^d_k \\ w^m_k \\ w^u_k \\ w^v_k \end{pmatrix}$$

$$y^m_k = \begin{pmatrix} C_0 & C_d & C_m \end{pmatrix} \begin{pmatrix} x^0_k \\ x^d_k \\ x^m_k \end{pmatrix} + \begin{pmatrix} D_m & D & 0 & 0 \end{pmatrix} \begin{pmatrix} w^d_k \\ w^m_k \\ w^u_k \\ w^v_k \end{pmatrix}$$

State Estimation - Temporal Update

$$\begin{pmatrix} \hat{x}^0_{k+1|k} \\ \hat{x}^d_{k+1|k} \\ \hat{x}^m_{k+1|k} \end{pmatrix} = \begin{pmatrix} A_0 & B_0 C_d & 0 \\ 0 & A_d & 0 \\ 0 & 0 & A_m \end{pmatrix} \begin{pmatrix} \hat{x}^0_{k|k-1} \\ \hat{x}^d_{k|k-1} \\ \hat{x}^v_{k|k-1} \end{pmatrix} + \begin{pmatrix} B_u \\ 0 \\ 0 \end{pmatrix} u_k$$

$$+ \begin{pmatrix} B_v \\ 0 \\ 0 \end{pmatrix} v_k + \begin{pmatrix} B_0 D_d & 0 & B_u & B_v \\ B_d & 0 & 0 & 0 \\ 0 & B_m & 0 & 0 \end{pmatrix} \begin{pmatrix} w^d_k \\ w^m_k \\ w^u_k \\ w^v_k \end{pmatrix}$$

$$\hat{y}^m_k = \begin{pmatrix} C_0 & C_d & C_m \end{pmatrix} \begin{pmatrix} \hat{x}^0_{k|k-1} \\ \hat{x}^d_{k|k-1} \\ \hat{x}^m_{k|k-1} \end{pmatrix}$$

Software

- MPCtools - Developed by Johan Åkesson, Automatic Control, Lund
  - Separates measured/constrained/controlled outputs
  - Integral action by means of disturbance estimation
  - Different QP-solvers for the optimization problem
  - Used in Lab 3 i Predictive control
- Mathworks - Model Predictive Control Toolbox
  - GUI or text-based control design
  - Integrated observer design and basic disturbance modeling
- MPT - Multi-Parametric Toolbox for Matlab from ETH
  - Supports linear, nonlinear, and hybrid system descriptions
  - Explicit MPC
- CVX+CVXGen: MPC code generator, runs typically on micro or millisec for small size problems. Nice project!

MPC Toolbox

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)

MPCTool - 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

Solving the Quadratic Program (QP)

One can rewrite the optimization criterion on the form

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

where $\Delta\mathcal{U}, \mathcal{H}, \mathcal{G}, \mathcal{E}, \mathcal{Q}, \Omega, \omega$ 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 not utilized in MPCTool)

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 y_a 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

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

for some vector $\bar{K}_s$

Linear Properties of the MPC Controller

This means the control law can be written

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

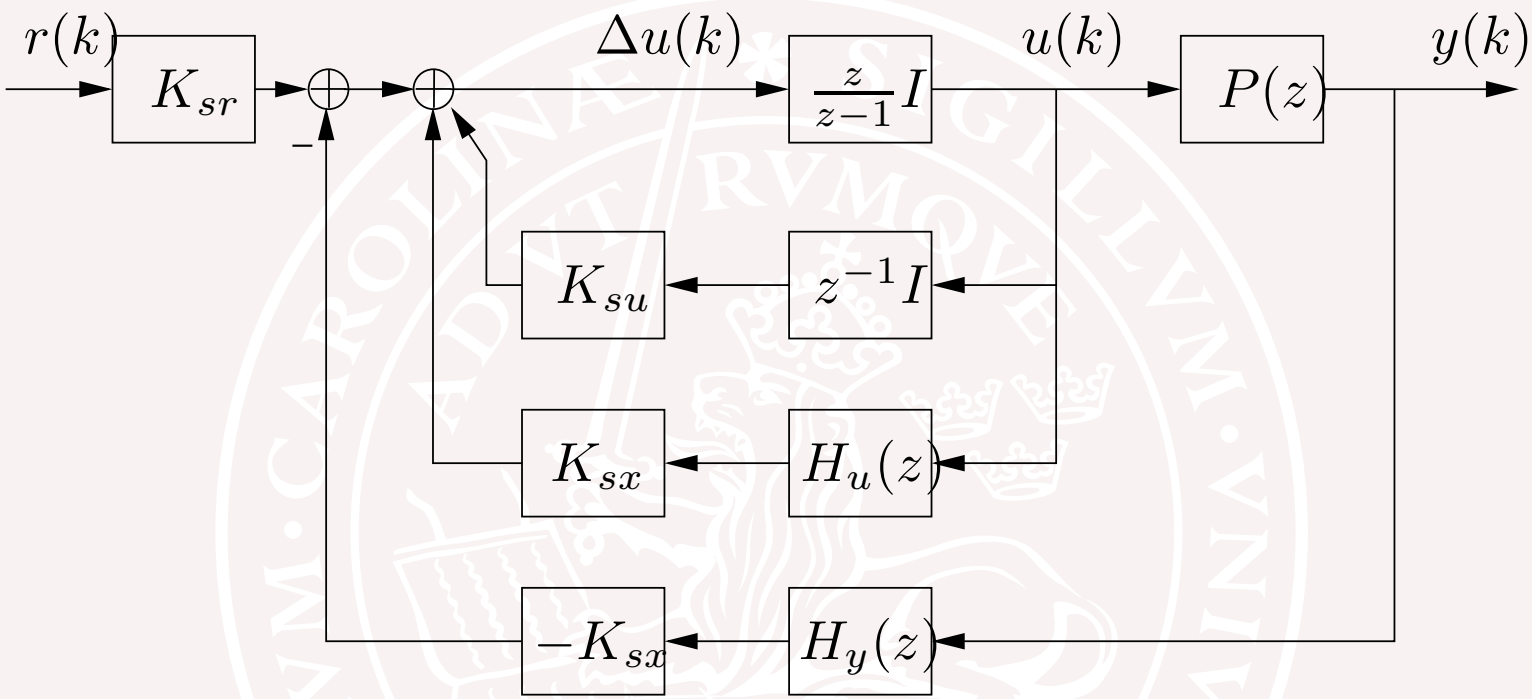
where $\begin{bmatrix} K_{sr} & K_{su} & K_{sx} \end{bmatrix}$ is given by the first m rows of $\bar{K}_s$

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 linear timeinvariant 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

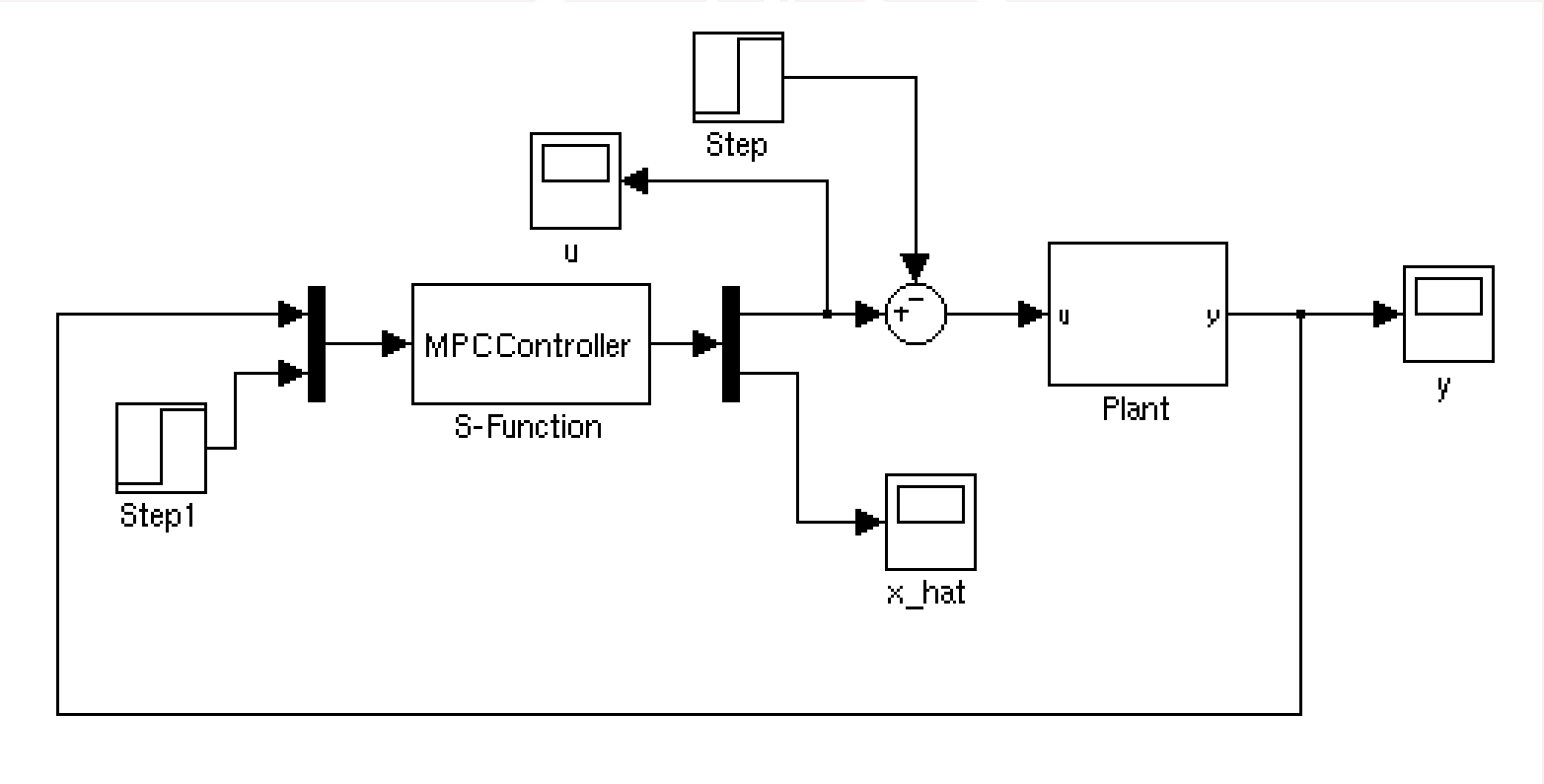
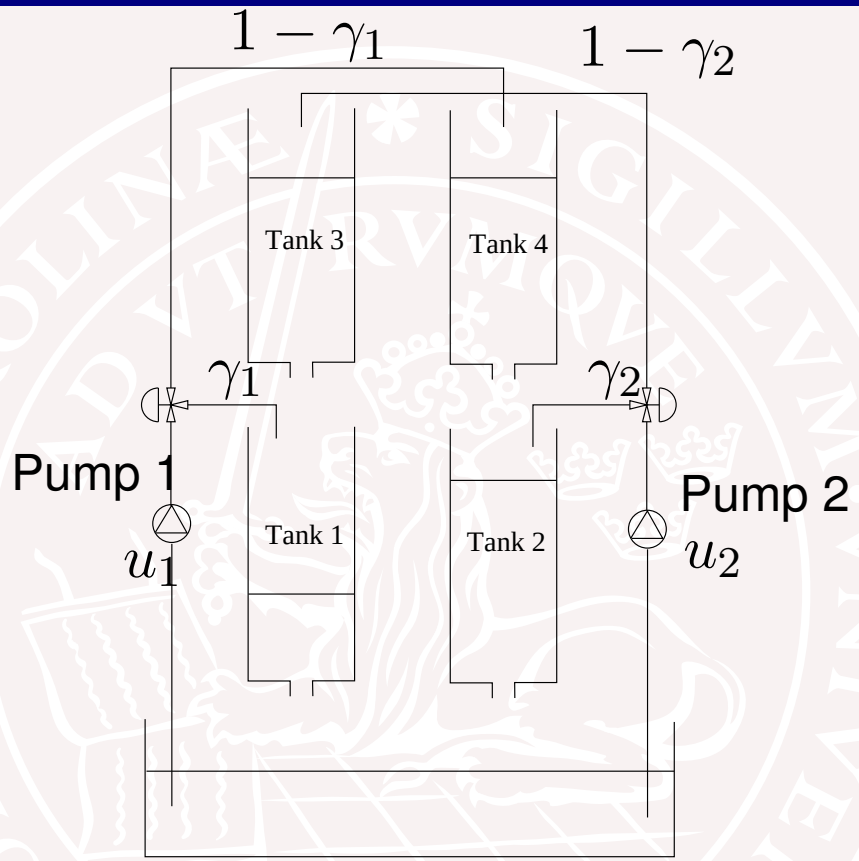


Figure: A Simulink model where the MPC controller is used to control a nonlinear plant.

Example - Quad Tank



Challenging MIMO process. Parameters γ_1, γ_2 control the flow structure to the upper and lower tanks respectively

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

See e.g. MPCTools manual for a model

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)

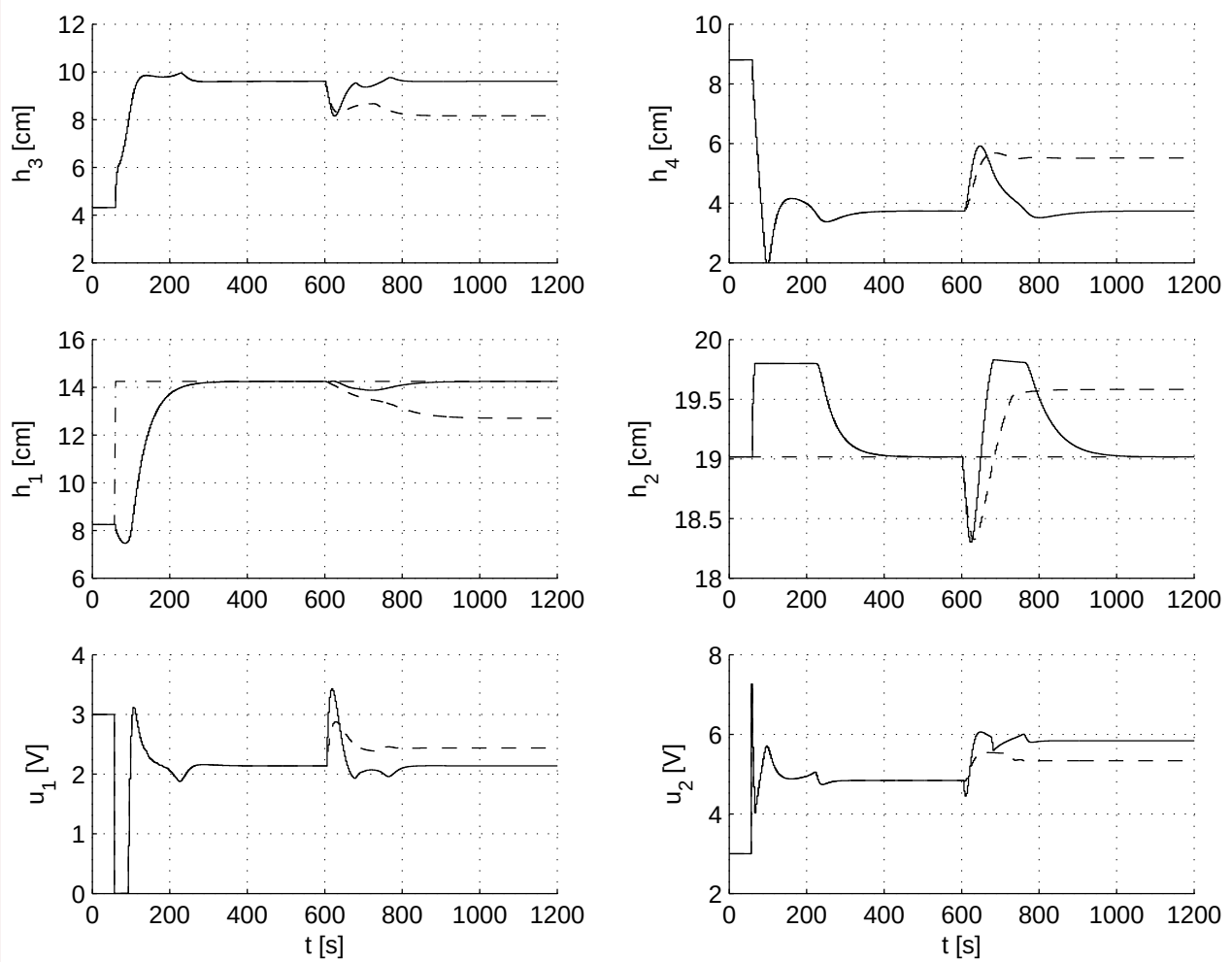
Constraints: $0 \leq x \leq 19.8$ cm on all tanks

$0 \leq u \leq 10$ 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
```

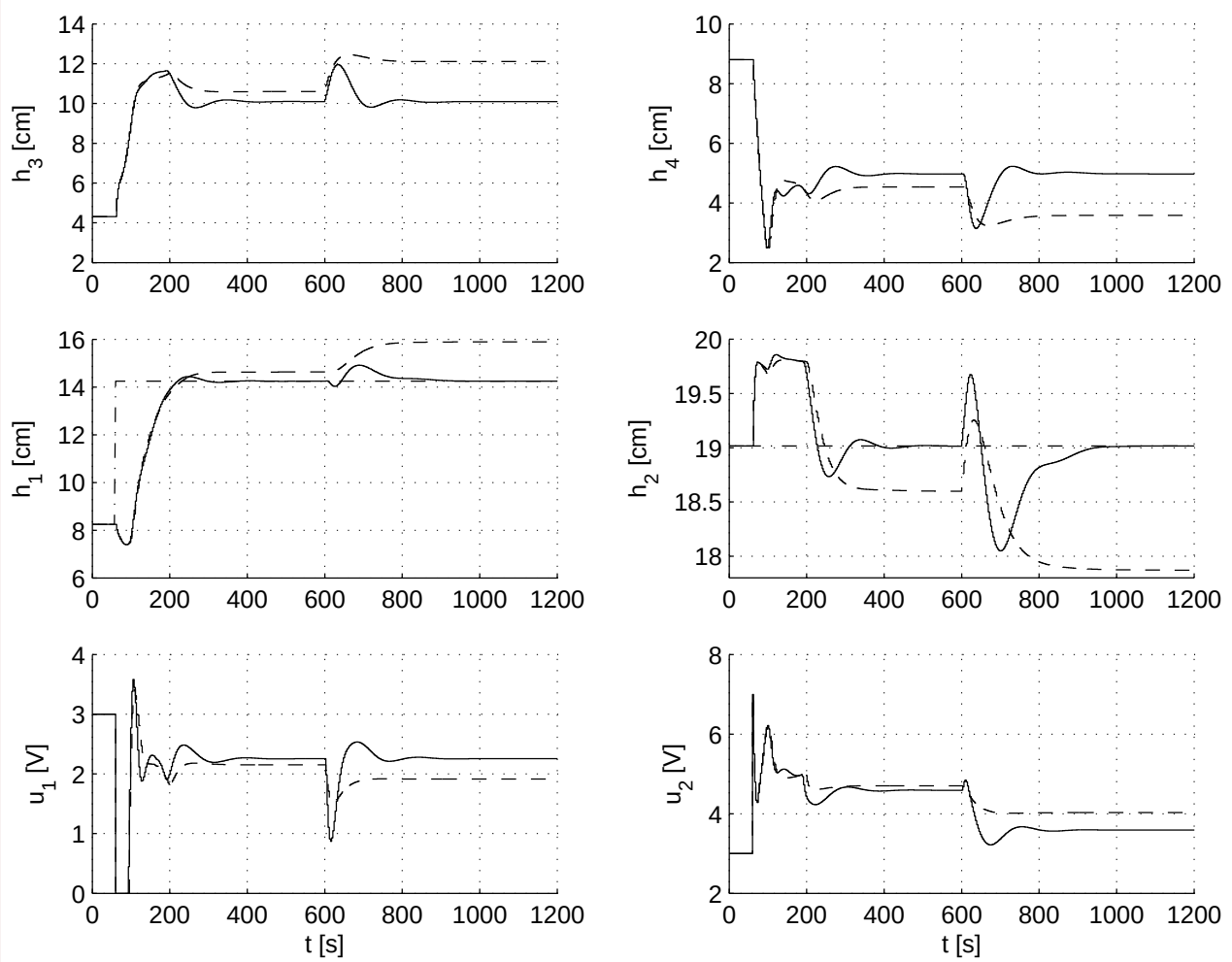
Results, simulation on linearized plant



Dashed: Kalman filter, no integral action

Solid: Kalman filter with disturbance observer

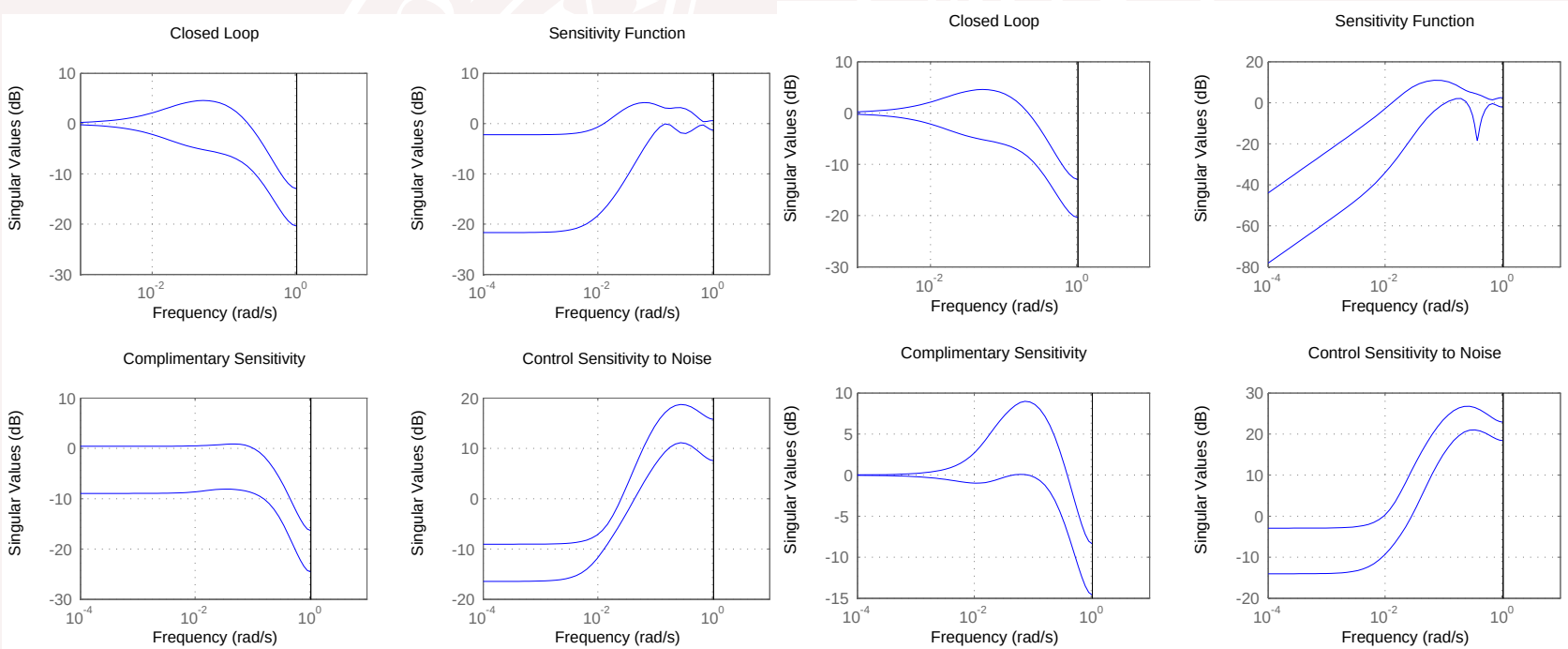
Results, simulation on nonlinear plant



Works well

Results, MPC linear behavior

Without the contraints, the MPC controller gives the following gain curves. The plots show singular values for the 2×2 system.



Left: Without integral action Right: With integral action

Explicit MPC

- For applications where very short sample times are required, there may not be time to solve the optimization problem at each time step.
- Explicit MPC calculates a state feedback law that is equivalent to MPC in a specific region of the state-space
- As constraints are activated, the feedback law changes in different parts of the state-space
- Certainty equivalance used $u = -K(\hat{x})\hat{x}$

Explicit MPC Example

For the plant

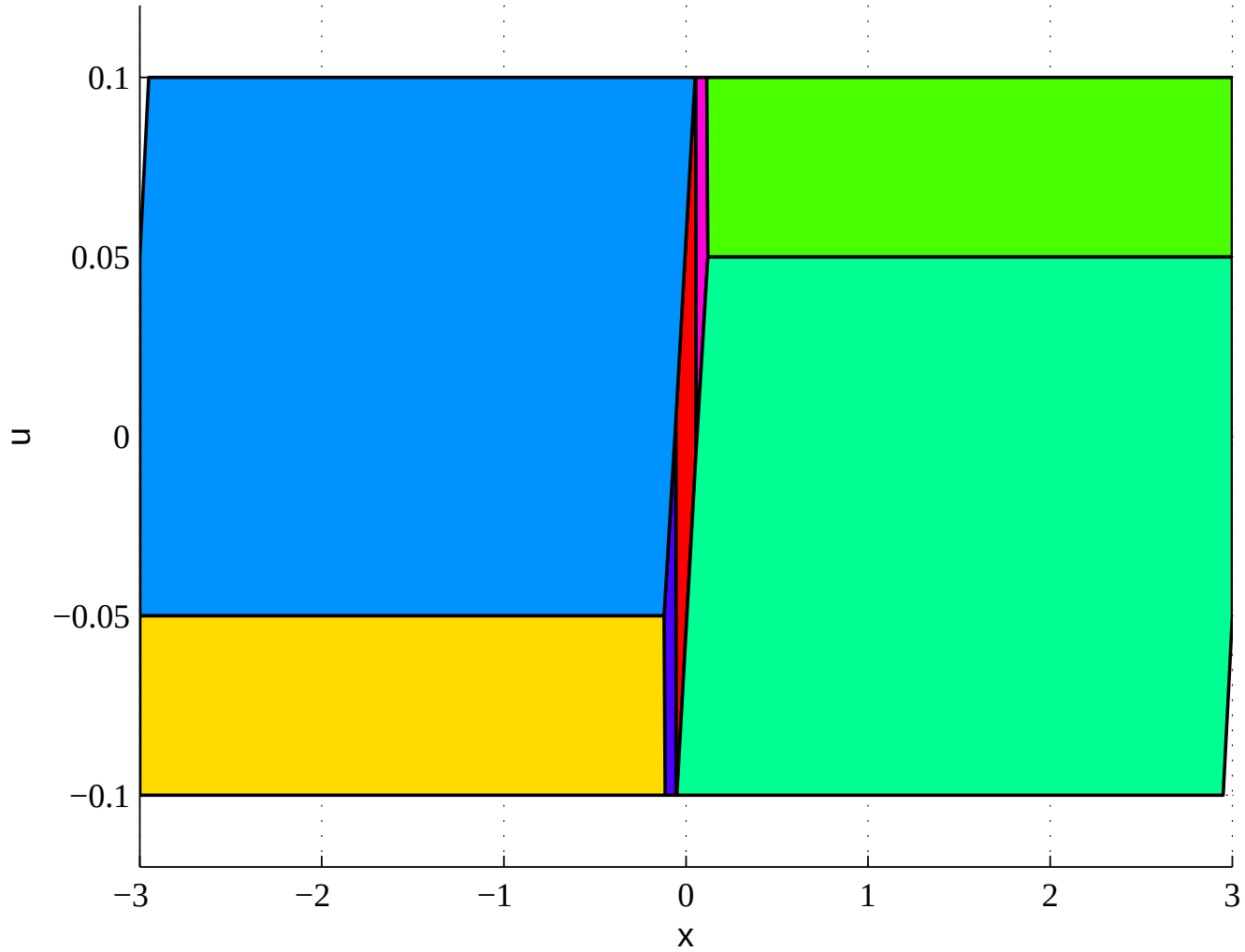
$$Y(s) = \frac{1}{s+1}U(s)$$

with constraints

$$\begin{aligned} -3 &\leq y(t) \leq 3 \\ -0.1 &\leq u(t) \leq 0.1 \\ -0.05 &\leq \Delta u(t) \leq 0.05 \end{aligned}$$

the regions might look like...

Explicit MPC Example



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
```