EL2520
Control Theory and Practice

Lecture 13:
Dealing with Hard Constraints

Elling W. Jacobsen
School of Electrical Engineering and Computer Science
KTH, Stockholm, Sweden

Lecture 14 - Wed May 15

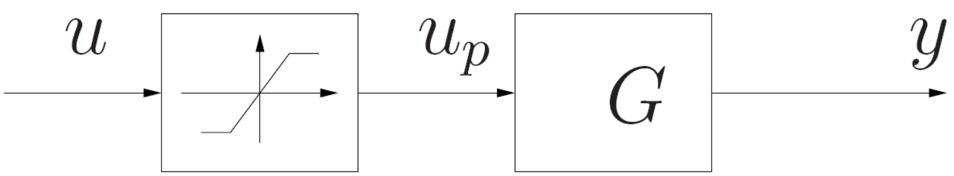
- Brief summary of course and information about exam
- Opportunity for getting things repeated or questions answered
- If you want specific things repeated, send me an email at latest on May 14 (tomorrow!)

Input Constraints

Dealing with input constraints:

- Linear control design: punish large control moves, e.g.,
  - LQG: choose large input weight Q_2
  - H_∞: include e.g, $\|G_{wu}\|_\infty$ in objective function
- But, inputs often have hard constraints

$$u_{min} \leq u_p \leq u_{max}$$



Outline Lecture 13

Dealing with hard constraints

- Constrained Receding Horizon Control / MPC
  - recap and additional issues
- Anti reset windup
  - the classical approach to deal with hard constraints
  - IMC approach

Model Predictive Control

- Finite-horizon discrete time LQR with hard constraints on u and y:

$$\begin{array}{ll} \text{minimize} & J(U) = \sum_{k=0}^{N-1} (x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N \\ \text{subject to} & u_{\min} \le u_k \quad \le u_{\max}, \quad k = 0, \ldots, N-1 \\ & y_{\min} \le C x_k \le y_{\max}, \quad k = 1, \ldots, N \\ & x_{k+1} = A x_k + B u_k \end{array}$$

- Can be simplified by eliminating $\{x_1, \ldots, x_N\}$ (as in last lecture)
  - results in a quadratic programming problem in $\{u_0, \ldots, u_{N-1}\}$

- Implement only u_0 , let system evolve one sample and redo optimization (with new state estimate)
  - results in receding horizon optimization

Quadratic Programming (QP)

- Minimizing a quadratic objective function subject to linear constraints

$$\begin{array}{ll} \text{minimize} & u^T P u + 2q^T u + r \\ \text{subject to} & Au \leq b \end{array}$$

- Any u satisfying $Au \leq b$ is said to be **feasible.**
  - clearly, not all quadratic programs are feasible
    (depends on A, b; more about this later…)

- “Easy” to solve when objective function is **convex** (P positive semidefinite)
  - optimal solution found in polynomial time
  - commercial solvers deal with 10,000’s of variables in a few seconds

Constrained control via QP

$$\begin{array}{ll} \text{minimize} & J(U) = \sum_{k=0}^{N-1}(x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N \\ \text{subject to} & u_{\min} \le u_k \quad \le u_{\max}, \;\; k = 0, \dots, N-1 \\ & y_{\min} \le C x_k \le y_{\max}, \;\; k = 1, \dots, N \\ & x_{k+1} = A x_k + B u_k \end{array}$$

Last lecture, introducing $X = (x_0, \dots, x_N),\; U = (u_0, \dots, u_{N-1})$, we can write

$$X = GU + Hx_0$$

and the objective function can be written as

$$J(U) = U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ}$$

Convex if $Q_2 \succ 0$ (implies that P_{LQ} is positive semi-definite)

What about the constraints?

Predictive control with constraints

Constraints on outputs can be written

$$Y \geq y_{\min}\mathbf{1} \Leftrightarrow \overline{C}(GU + Hx_0) \geq y_{\min}\mathbf{1} \Leftrightarrow \underbrace{\overline{C}G}_{A_{\underline{Y}}} U \geq \underbrace{y_{\min}\mathbf{1} - \overline{C}Hx_0}_{b_{\underline{Y}}}$$

$$Y \leq y_{\max}\mathbf{1} \Leftrightarrow \overline{C}(GU + Hx_0) \leq y_{\max}\mathbf{1} \Leftrightarrow \underbrace{\overline{C}G}_{A_{\overline{Y}}} U \leq \underbrace{y_{\max}\mathbf{1} - \overline{C}Hx_0}_{b_{\overline{Y}}}$$

and constrained LQR problems can then be formulated as QP problem:

$$\begin{array}{ll} \text{minimize} & U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ} \\ \text{subject to} & \begin{bmatrix} A_{\overline{Y}} \\ -A_{\underline{Y}} \\ I \\ -I \end{bmatrix} U \leq \begin{bmatrix} b_{\overline{Y}} \\ -b_{\underline{Y}} \\ u_{\max}\mathbf{1} \\ -u_{\min}\mathbf{1} \end{bmatrix} \end{array}$$

Model predictive control algorithm

1. Given state at time t compute ("predict") future states

$$x_{t+k}, \qquad k = 0, 1, \ldots, N$$

   as function of future control inputs

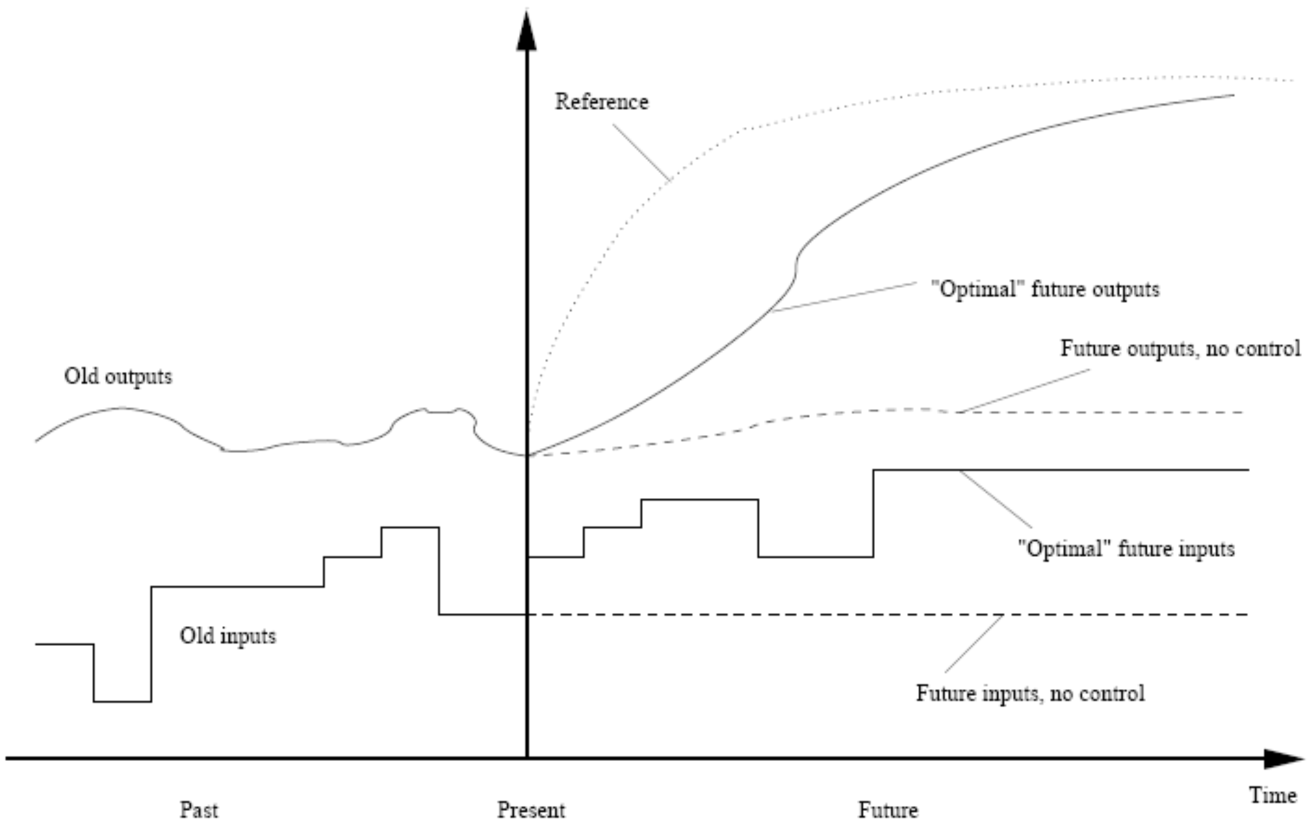
$$u_{t+k}, \qquad k = 0, 1, \ldots, N-1$$

2. Find "optimal" input by minimizing constrained cost function
   - a quadratic program, efficiently solved

3. Implement u(t)

4. A next sample (t+1) return to 1 (with new estimate of state from state estimator, e.g., Kalman filter)

MPC trajectories



Example: the DC servo

Discrete-time model (sampling time 0.05 sec)

$$A = \begin{bmatrix} 1 & 0.139 \\ 0 & 0.861 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0107 \\ 0.139 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Constrained input voltage

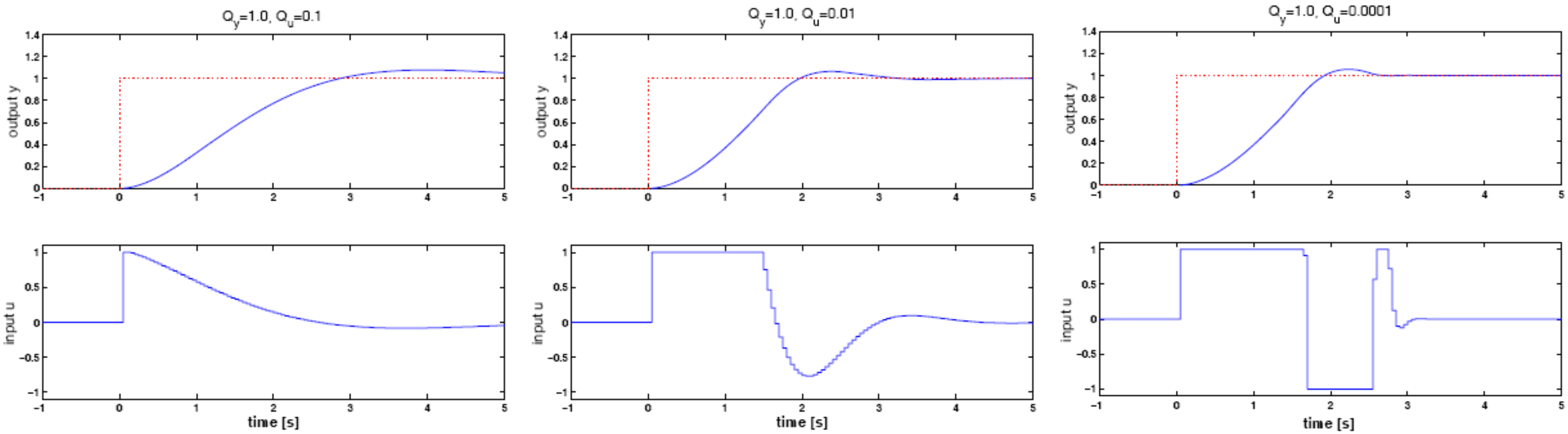
$$-1 \leq u \leq 1$$

Constrained position

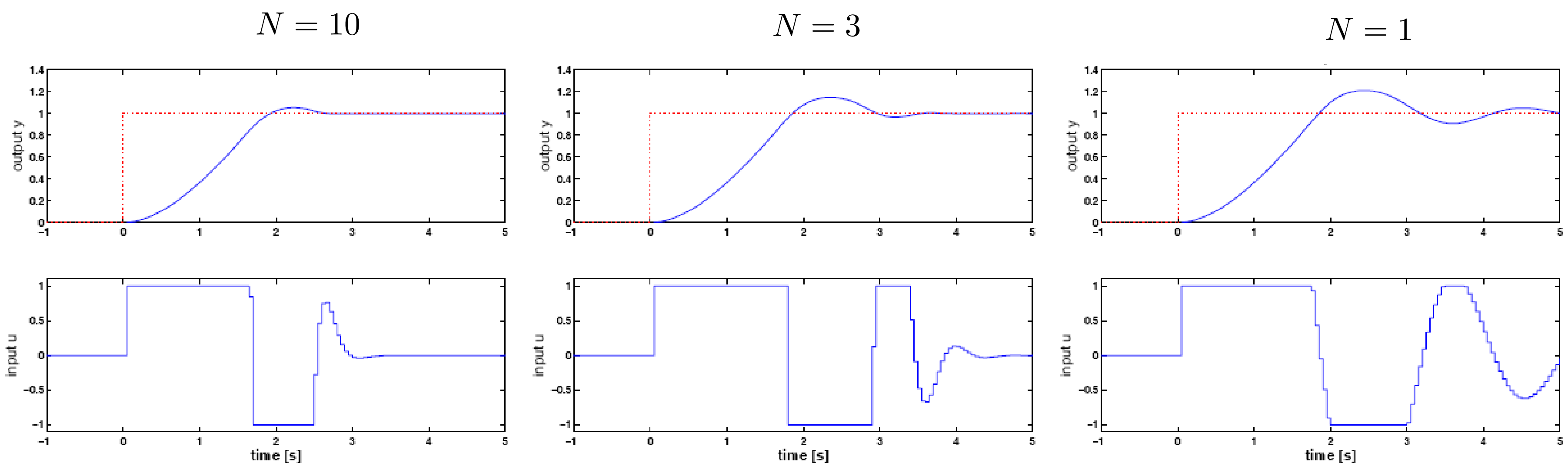
$$y_{\mathsf{min}} \leq y_k \leq y_{\mathsf{max}}$$

Impact of state and control weights

Prediction horizon N=10.

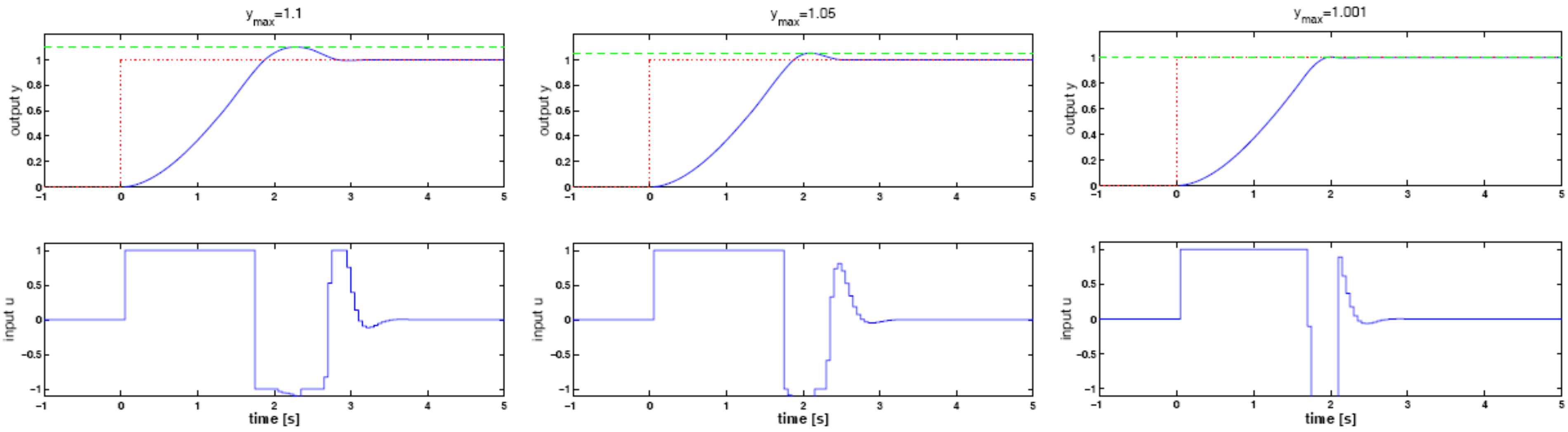


Impact of horizon



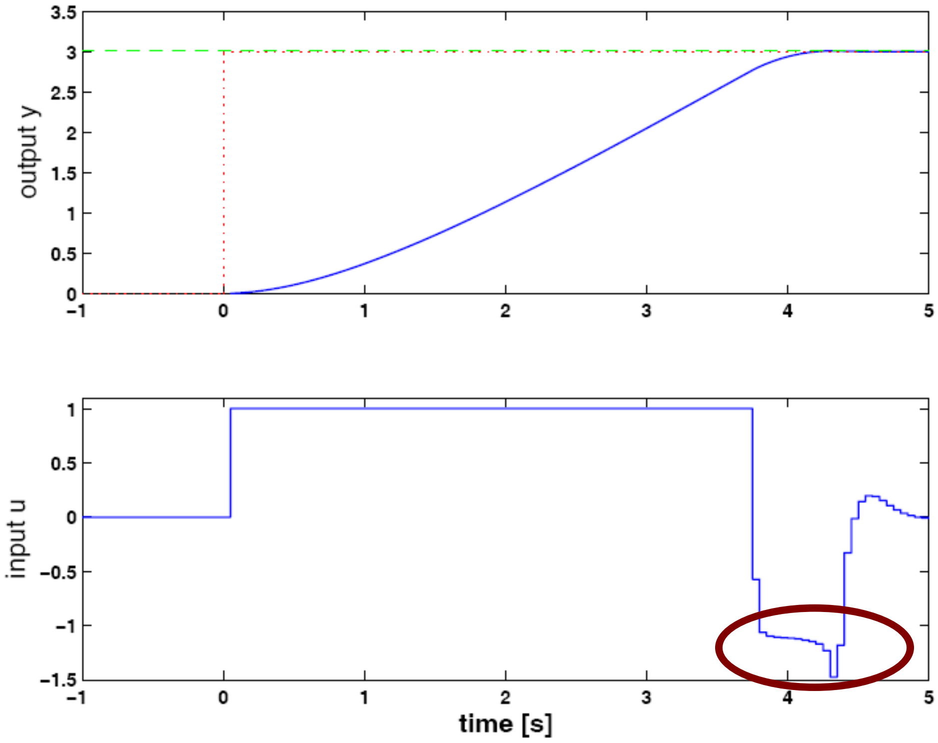
Too short horizon ➔ inaccurate predictions ➔ poor performance

Adding output constraints



Infeasibility

What happens when there is no solution to the QP?



Not clear what control to apply!

Ensuring feasibility

- One way to ensure feasibility:
  - introduce slack variables $s_{ck} \geq 0$
  - "soften" constraints

$$u_k \leq u_{\max} \Rightarrow u_k \leq u_{\max} + s_{ck}$$

  - add term in quadratic programming objective to minimize slacks

$$\underset{U}{\text{minimize}} \quad U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ}$$

$$\Downarrow$$

$$\underset{U,S}{\text{minimize}} \quad U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ} + \kappa S^T S$$

Notes:

- still QP, but more variables; added penalty κS
- Usually better to soften "physically soft" constraints (e.g. output constraints)

Reference tracking

- Would like z to track a reference sequence $\{r_1, \ldots, r_N\}$, i.e. to keep

$$\sum_{k=0}^{N-1} \left((z_k - r_k)^T Q_1 (z_k - r_k) + u_k^T Q_2 u_k \right) + (z_N - r_N)^T Q_f (z_N - r_N)$$

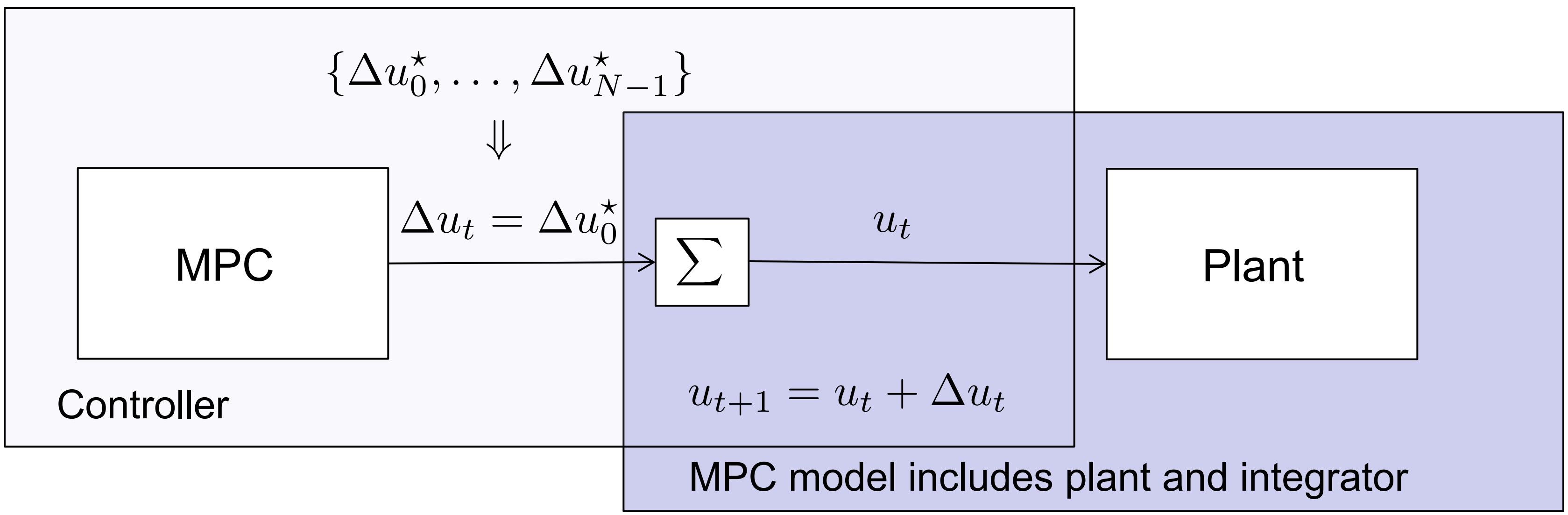
  small.

- Problem: making $z_k = r_k \neq 0$ typically requires $u_k \neq 0$
  - a trade-off between zero tracking errors and using zero control
  - often results in steady-state tracking error

Including integral action

Integral action often included by a change in free variables

- use $\Delta u_i = u_i - u_{i-1}$ as variables in the optimization
- actual input obtained by summing up MPC outputs



Including integral action cont'd

Form augmented model with state $\bar{x}_k = (x_k,\ u_k)$ and input Δu_k :

$$\begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \Delta u_k$$

$$y_k = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Consider finite-horizon cost

$$\sum_{k=0}^{N-1} \left((z_k - r_k)^T Q_1 (z_k - r_k) + \Delta u_k^T Q_2 \Delta u_k \right) + (z_N - r_N)^T Q_f (z_N - r_N)$$

Now, all terms can go to zero (at least when unconstrained, infinite horizon)

Apply control $\quad u_t = u_{t-1} + \Delta u_{t-1}$

MPC controller tuning

MPC has a large number of “tuning” parameters:

- The prediction model
  - we need to decide sampling interval
    (rule of thumb: sample frequency 10 times desired closed-loop bandwidth)
  - obtain discrete-time state-space model

- Finite-horizon optimal control
  - set prediction horizon
    (rule of thumb: equal to closed-loop rise time; could be smaller)
  - decide weight matrices (as for continuous-time LQG)
  - decide final state penalty

MPC controller tuning

- Finite-horizon optimal control, advanced:
  - control horizon
    (try to set small, rule-of-thumb: use 1-10)
  - inner-loop control
    (guideline: stationary LQR controller for given weight matrices)

- Constraints and feasibility
  - specify control and state/output constraints (problem dependent)
  - introduce slacks to “soften” constraints
  - choose constraint penalty (large value on kappa)

- Integral action (almost always a good idea to include).

Advanced issues: stability

- Receding horizon control might yield unstable closed-loop

- Stability can be guaranteed:
  - for infinite-horizon unconstrained case (this is LQR)
  - for finite-horizon unconstrained case
    - if final state is penalized correctly
    - if final state is enforced to lie in a given set
  - for constrained finite-horizon
    - if final state enforced to lie in a sufficiently small set **and**
    - initial QP (solved at time zero) is feasible

- Hard to verify for sure in advance…

Advanced issues: robustness

Consider the unconstrained quadratic program

$$\text{minimize} \quad u^T Q u + 2q^T u$$

which has optimal solution $u = -Q^{-1} q$

In the MPC setting, Q and q depend on the system model (matrices A, B, C), weights Q_1, Q_2, and also horizons.

Solution is sensitive to uncertainties if Q is ill-conditioned

- try scaling inputs and outputs in the model
- modify weight matrices Q_1 and Q_2
- almost always a good idea to include integral action

Advanced issues: observers

- MPC, as presented here, assumes full state feedback.

- We essentially always need to use an observer
  - to reconstruct states, and
  - to impose feedback to deal with unmeasured disturbances and model uncertainty!

- Limited theory, but separation principle holds in some cases.

- Suggests guideline
  - design observer as for (unconstrained, infinite-horizon) LQG
  - use estimated state in MPC calculations as if it was true state

MPC Course

EL2700 Model Predictive Control, 7.5cr, given in period 1.

See course homepage for more information

Summary MPC

Model predictive control (MPC)

- can handle input and output constraints
- predictive control computed via quadratic programming

Many parameters and their influence on the control

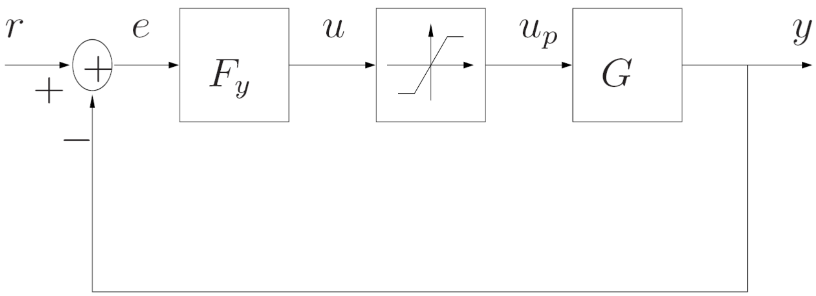
- system model, weights, horizons, constraints, …

Advanced issues:

- feasibility and slacks to “soften” constraints
- integral action
- different prediction and control horizons
- stability and the terminal weight
- the need for a state observer

Anti-Windup

- The problem with saturating input



  - feedback broken, i.e., system open-loop, when u in saturation
  - problem in particular if F or G unstable
  - F usually has integrator (unstable)
- The classical approach to deal with hard constraints on the input is called anti-reset windup

Anti-reset Windup

Many controllers based on feedback from observed states

- Observer:

$$\frac{d}{dt}\widehat{x}(t) = A\widehat{x}(t) + Bu(t) + K(y(t) - C\widehat{x}(t))$$

- Feedback from observed states

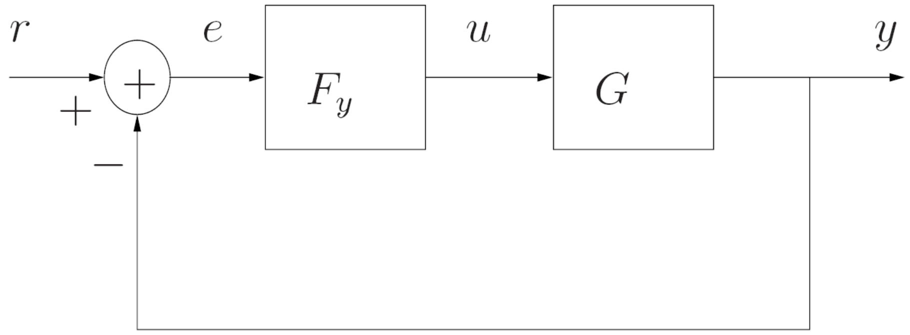
$$u = -L\widehat{x}$$

- Controller transfer-function

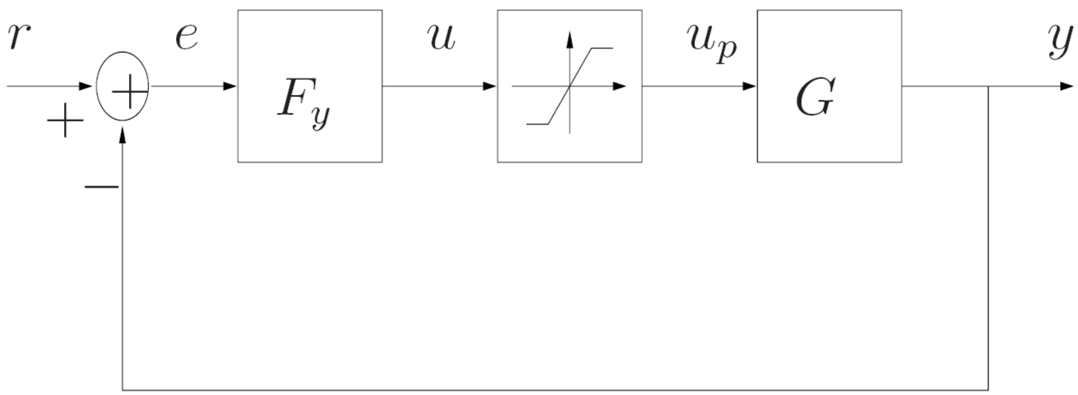
$$U(s) = -F_y(s)Y(s) = -L(sI - A + BL + KC)^{-1}KY(s)$$

Magnitude limitations on control

Linear model



Actual implementation



Example: DC servo

Servo:

$$G(s) = \frac{1}{s(s+1)}$$

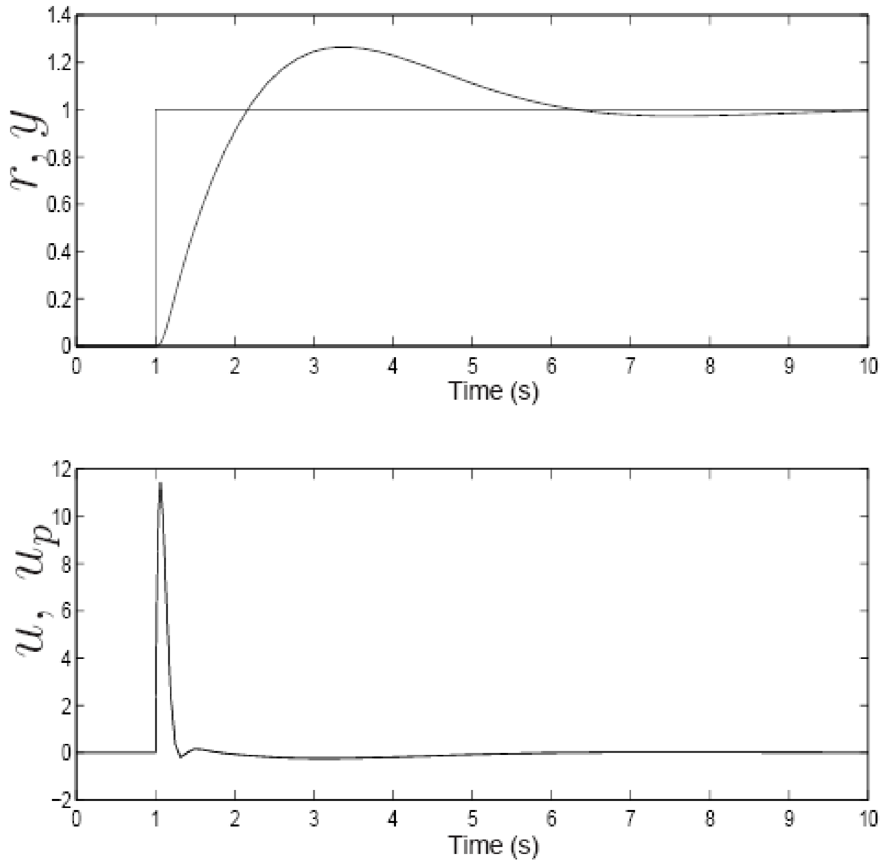
A controller designed using LQG is

$$F_y(s) = \frac{439s^2 + 710.5s + 316.2}{s^3 + 26.47s^2 + 349.8s - 7.13}$$

which has poles in -13.2444 +- 13.2255i, and 0.0204

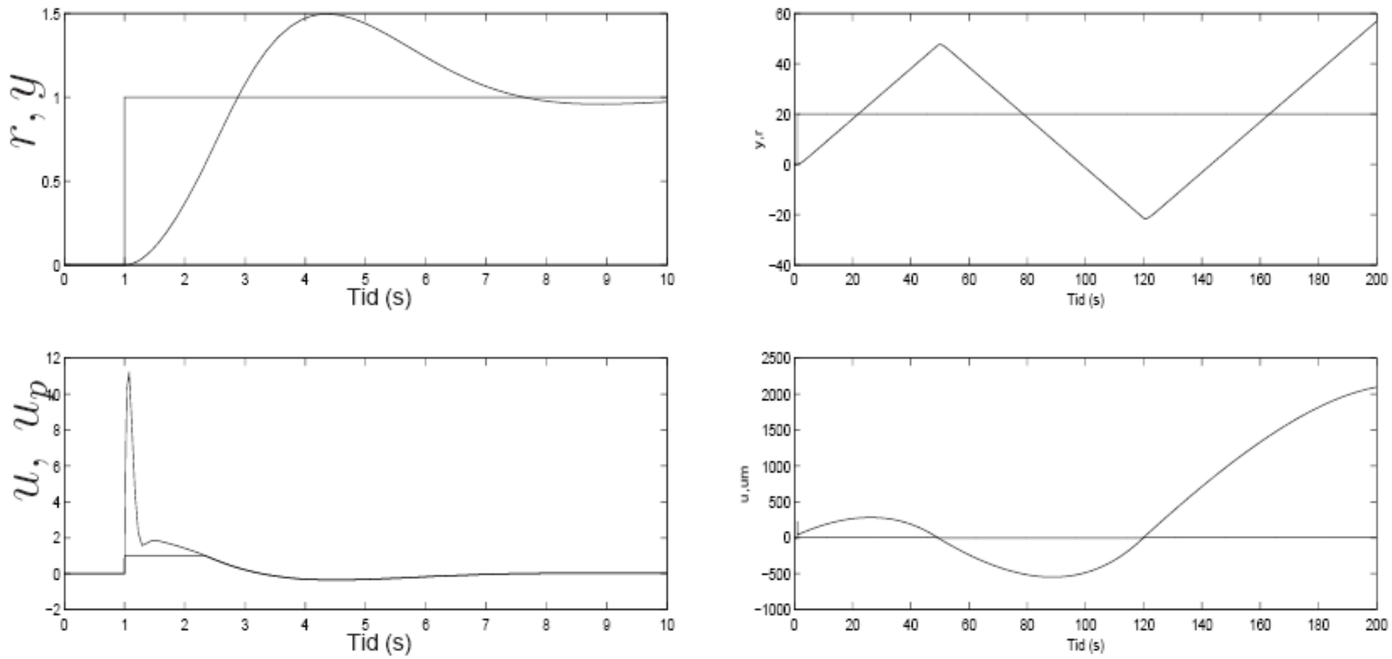
Note: controller is unstable, but closed loop is internally stable!

Step response (no constraints)



Step response with saturated input

$$-1 \leq u_p \leq 1$$



Slower, larger overshoot

unstable

A solution: modified observer

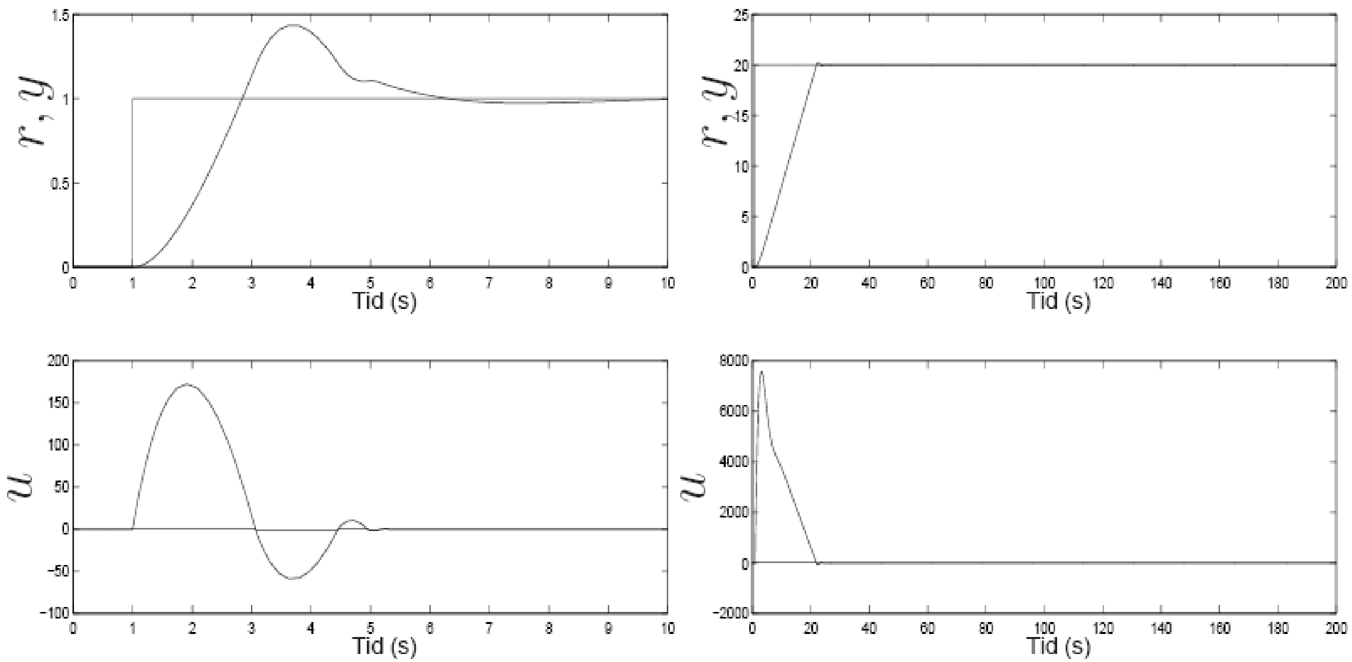
Observer should reflect true dynamics

$$\frac{d}{dt}\widehat{x}(t) = A\widehat{x}(t) + Bu_p(t) + K(y(t) - C\widehat{x}(t))$$

The constrained (actually applied) input is used in observer

- a nonlinear observer!
- based on measuring the actual input or having a model of the constraint

Step responses with modified observer



Analysis: stability also in saturation

$$\frac{d}{dt}\hat{x}(t) = A\hat{x}(t) + Bu_p(t) + K(y(t) - C\hat{x}(t)) =$$
$$= (A - KC)\hat{x}(t) + Bu_p(t) + Ky(t)$$

Controller transfer function

$$U(s) = -L(sI - A + KC)^{-1}KY(s) - L(sI - A + KC)^{-1}BU_p(s)$$

In saturation ($u<u_{min}$ or $u>u_{max}$), u_p is constant

Thus, in saturation, the controller dynamics is given by A-KC whose eigenvalues are -0.5446±0.7276i, -1.2106 (i.e. stable)

This modification is known as *anti-reset windup.*

Interpretation: feedback from u-u_p

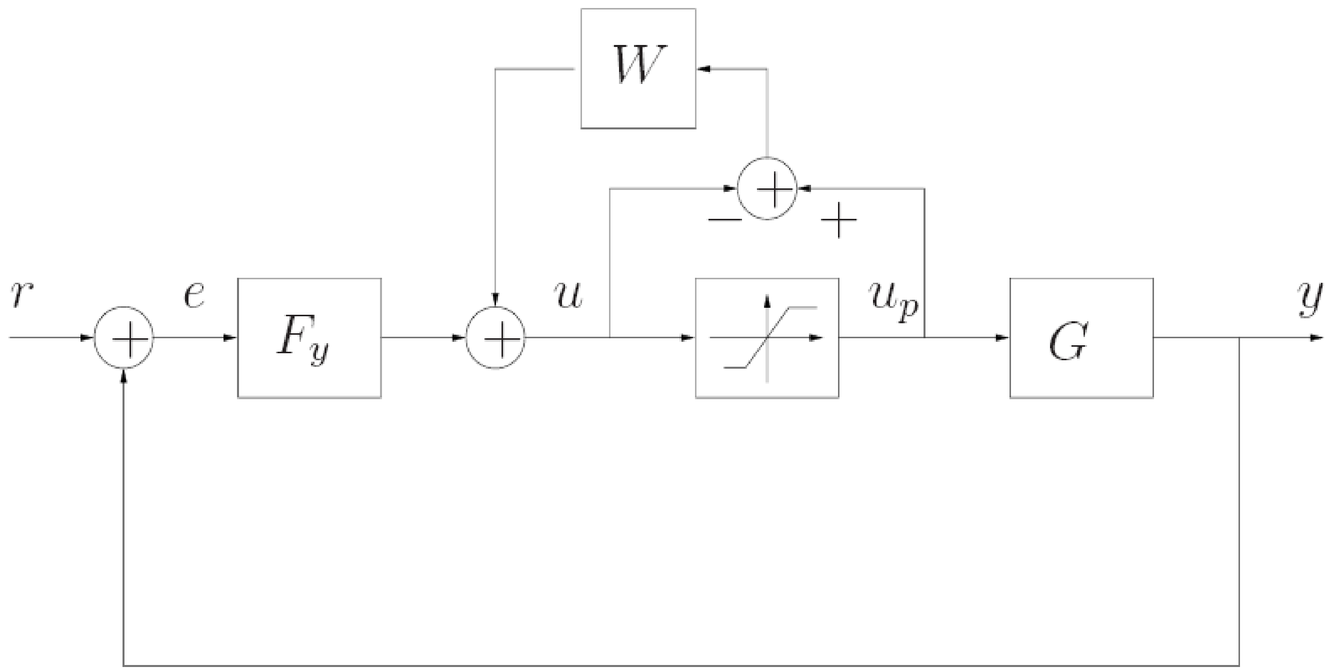
Write controller as

$$\begin{aligned}\frac{d}{dt}\widehat{x}(t) &= A\widehat{x}(t) + Bu_p(t) + K(y(t) - C\widehat{x}(t)) = \\ &= (A - KC)\widehat{x}(t) + B(u_p(t) + u(t) - u(t)) + Ky(t) = \\ &= (A - BL - KC)\widehat{x}(t) + Ky(t) + B(u_p(t) - u(t))\end{aligned}$$

Taking Laplace transforms

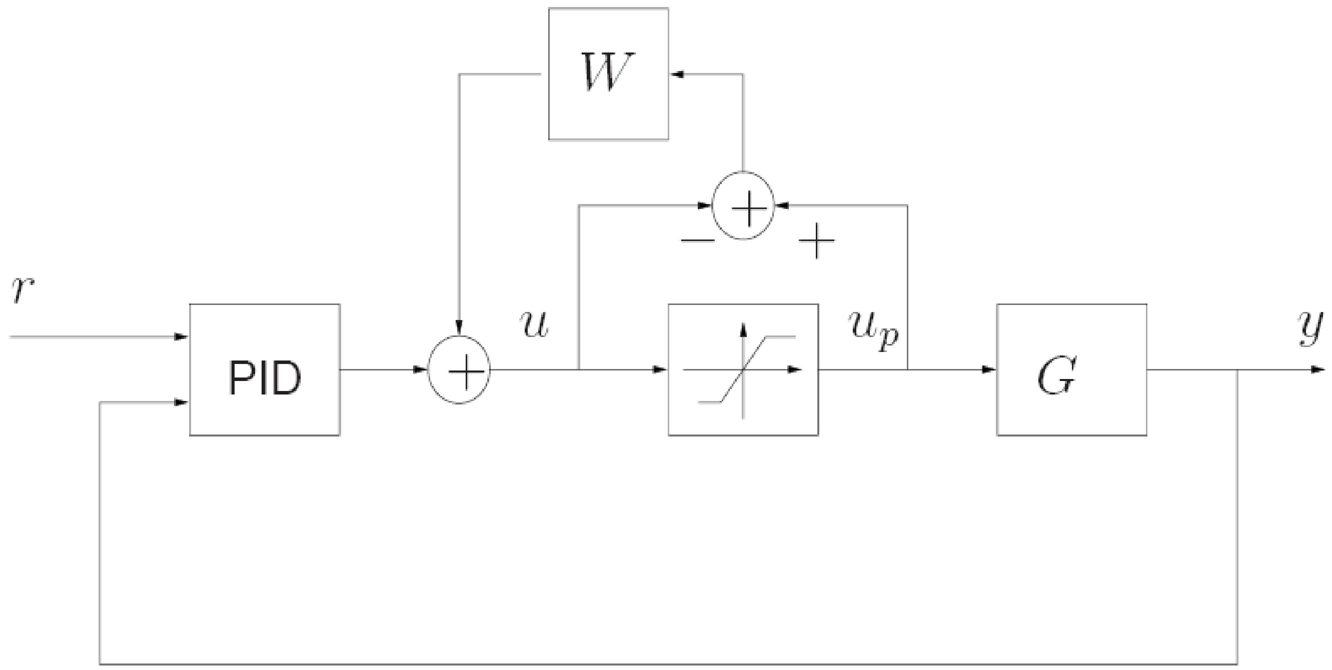
$$\begin{aligned}U(s) =& - L(sI - A + BL + KC)^{-1}KY(s) \\ & - L(sI - A + BL + KC)^{-1}B(U_p(s) - U(s)) = \\ =& - F_y(s)Y(s) + W(s)(U_p(s) - U(s))\end{aligned}$$

In block diagram



- Anti-reset windup is based on tracking input

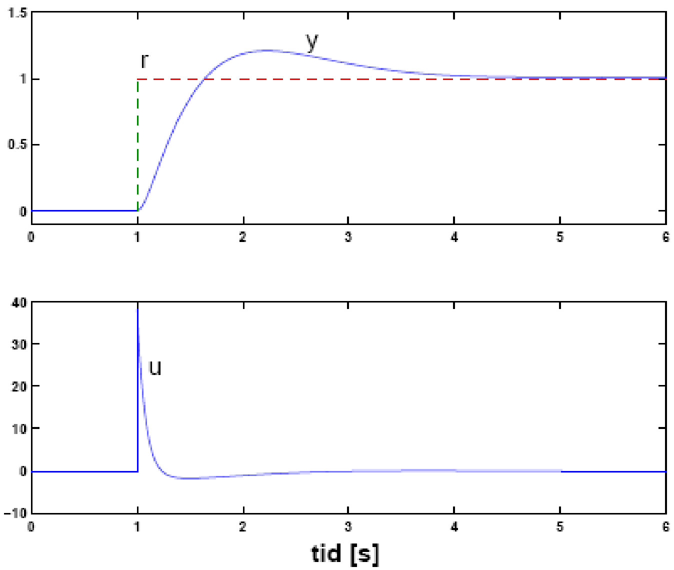
Application to PID controllers



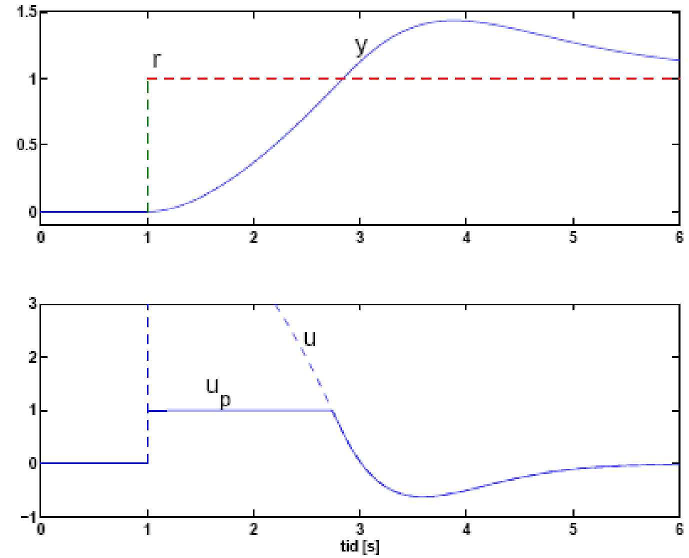
- Common choice: $W(s) = \dfrac{1}{sT_t}$

DC Servo under PID control

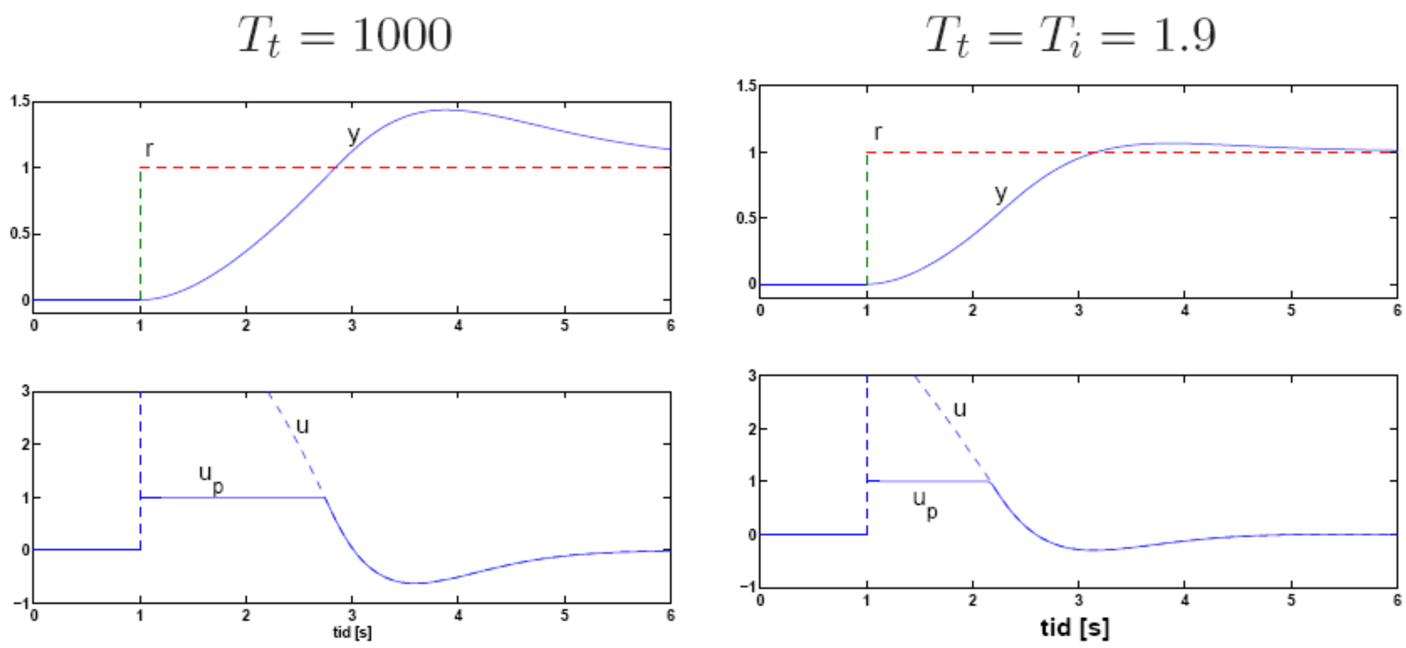
No constraint



$u_p \in [-1, 1]$, no compensation



Servo: PID+Anti-reset Windup



Summary

- Hard constraints: a nonlinearity essentially always present in real control systems
- Main problem: system drifts off when input in saturation
- Approaches to deal with hard constraints
  - constrained receding horizon LQG control (MPC)
  - anti-reset windup
  - IMC (next)