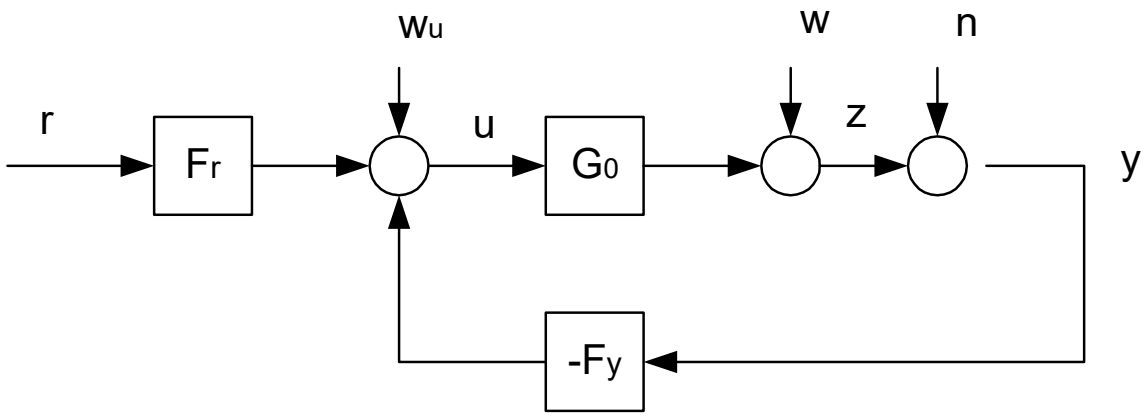
Chapter 8

- LQG control
- Robustness of LQG control
- Loop transfer recovery (LTR)
- Introduction to Model Predictive Control
- Final steps in the course
- Course end

Chapter 8: *LQG*-control



$$z = Gu + w$$

$$y = z + n$$

$$u = F_r r - F_y y$$

$$e = z - r$$

Note:

LQG theory (Linear-Quadratic-Gaussian) means optimal (LQ) control with Gaussian noise disturbances present. The stochastic theory of continuous time systems is difficult.

It is possible to present the theory in a ”simplified” form, but that is omitted here. The important thing is to know that noise intensities (variance does not exist for continuous time stochastic signals) are mostly used as tuning parameters only. The optimal state estimator, Kalman filter, is used to estimate the states, which are fed back according to the LQ theory.

LQG = Kalman filter + LQ

General state-space realization of the process is

$$\dot{x} = Ax + Bu + Nv_1$$
$$z = Mx$$
$$y = Cx + v_2$$

and the criterion

$$\min\left(\|e\|_{Q_1}^2 + \|u\|_{Q_2}^2\right) = \min\int\left[e^T(t)Q_1e(t) + u^T(t)Q_2u(t)\right]dt$$

$$e = z - r$$

Consider the *regulator problem* ($r = 0$)

$$\dot{x} = Ax + Bu + Nv_1$$

$$z = Mx$$

$$y = Cx + v_2$$

white noise $\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ with intensity $\begin{bmatrix} R_1 & R_{12} \\ R_{12}^T & R_2 \end{bmatrix}$

Determine

$u(t) = -F_y(p)y(t)$ such that

$V = \|z\|_{Q_1}^2 + \|u\|_{Q_2}^2$ is minimized

Solution (without proof)**:** Let (A,B) be stabilizable and (A,C) detectable.

The optimal control law is

$$u(t) = -L\hat{x}(t)$$
$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + K\left[y(t) - C\hat{x}(t)\right]$$

in which the Kalman gain K is obtained by the Riccati equation

$$AP + PA^T + NR_1N^T - \left(PC^T + NR_{12}\right)R_2^{-1}\left(PC^T + NR_{12}\right)^T = 0$$
$$K = \left(PC^T + NR_{12}\right)R_2^{-1}$$

and the state feedback coefficient L

$$L = Q_2^{-1} B^T S$$

where S is the solution to the stationary Riccati equation (LQ)

$$A^T S + SA + M^T Q_1 M - SBQ_2^{-1} B^T S = 0$$

(symmetric and positive semidefinite solution)

(Note that only infinite optimization horizons are considered here, so that the stationary Riccati equations can be used.)

The solution has the *separation property* : the optimal state observer and optimal state feedback can be designed independent of each other. The whole solution is then the ”combination” of these.

If the states are measurable , $y = x$, the Kalman-filter is not needed and the state feedback is formed directly from the state.

The theory guarantees that the resulting closed-loop system is stable.

Terms: LQ (linear quadratic)
LQG (linear quadratic gaussian)
ARE (algebraic Riccati equation)
(Separation principle)

Matlab:

kalman, estim
lqgreg
lqr, dlqr
lqe, dlqe
lqrd
sigma, dsigma

But how about the robustness of the LQ (LQG) – controller ?

$$e = (I - G_c)r - Sw + Tn$$
$$u = G_{ru}r + G_{wu}(w + n)$$

Not necessarily very good!

The weight matrices and noise intensities can be thought to be *tuning parameters* in control design.

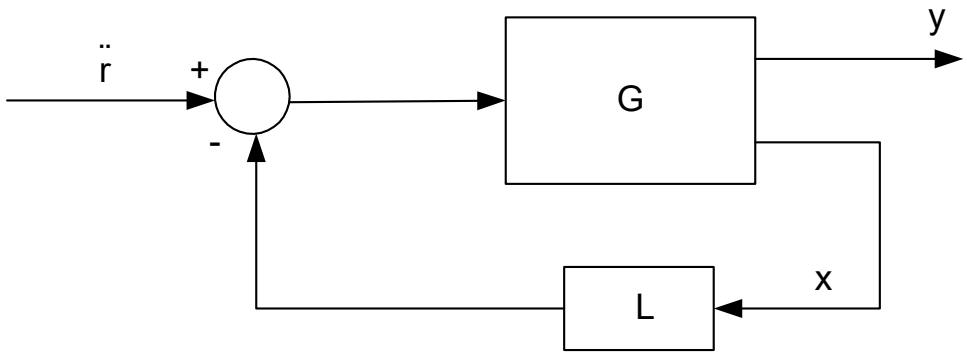
After design the frequency domain analysis and simulations must be carried out to verify the performance of the controller. Next, consider robustness a bit closer.

Robustness of LQ/LQG-controllers

LQ:

$$u(t) = -Lx(t)$$

$$\dot{x}(t) = Ax(t) + Bu(t)$$



The loop transfer function (gain in control signal)

$$G_k(s) = L\left(sI - A\right)^{-1} B$$

But in that transfer function

$$H(s) = L\left(sI - A\right)^{-1} B$$

L is determined from the equations

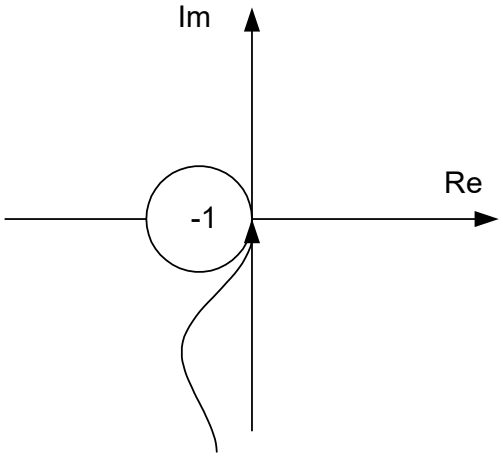
$$L = Q_2^{-1} B^T S$$
$$A^T S + SA + Q_1 - SBQ_2^{-1} B^T S = 0$$

and now it holds (apply the lemma 5.2 in the textbook)

$$\left[I + H(-i\omega)\right]^T Q_2 \left[I + H(i\omega)\right] \geq Q_2$$

which in *SISO*-case is

$$\left|1 + H(i\omega)\right| \geq 1$$



That means that the Nyquist curve will never go inside the circle shown in the figure:

-phase margin at least 60 degrees
-gain margin infinite
-the magnitude of the sensitivity function is less than one
-the magnitude of the complementary sensitivity function is smaller than two.

LQG: $$u(t) = -L\hat{x}(t) + \tilde{r}(t)$$

$$p\hat{x}(t) = A\hat{x}(t) + Bu(t) + K\left[y(t) - C\hat{x}(t)\right]$$

It follows

$$u(t) = -F_y(p)y(t) + F_r(p)\tilde{r}(t)$$

$$F_y(p) = L\left(pI - A + BL + KC\right)^{-1} K$$

$$F_r(p) = I - L\left(pI - A + BL + KC\right)^{-1} B$$

and the loop gains are

$$GF_y = C\left(sI - A\right)^{-1} BL\left(sI - A + BL + KC\right)^{-1} K$$

$$F_yG = L\left(sI - A + BL + KC\right)^{-1} KC\left(sI - A\right)^{-1} B$$

looked from the output or input side of the process . (For *SISO*-systems the functions are the same.)

But now the good robustness properties do not necessarily hold, even though K and L have been chosen according to the *LQG*-formulas (the phase margin can even be arbitrarily small).

An idea to fix that problem: let L be chosen as above. Can K be chosen such that

$$F_y G = L\left(sI - A + BL + KC\right)^{-1} KC\left(sI - A\right)^{-1} B \approx L(sI - A)^{-1} B$$

which would make it possible to enjoy the ”ideal” loop transfer function again.

Result: Yes, it is possible by choosing

$$K = \rho B$$

where ρ is large enough. That holds generally and also in MIMO case; the number of inputs and outputs must be the same.

This technique is called the **loop transfer recovery** (*LTR*).

The idea is to calculate L as the solution to the optimal control problem and then change K as described above. (To increase ρ until the desired sensitivity functions are obtained.) But there is no guarantee that the filter remains stable.

Use another procedure to aim at $K = \rho B$

$$\dot{x} = Ax + Bu + Nv_1$$

$$z = Mx$$

$$y = Cx + v_2$$

$$K = PC^T R_2^{-1}$$

$$PA^T + AP - PC^T R_2^{-1} CP + NR_1 N^T = 0 \qquad \text{(Kalman)}$$

Choose

$R_1 = \alpha R_2, \quad N = B$ α being ”large”. Then

$$PA^T + AP - KR_2 K^T + \alpha B R_2 B^T = 0$$

where the last two terms dominate, as α grows. Hence

$$K \approx \sqrt{\alpha} B$$

holds, and also the Kalman-filter remains Ok.

Now the tuning parameters were N, R_1 and R_2.

Note. The presented method was *input-LTR*, because the loop transfer function was $F_y G$ (gain of the input signal)

There exists also an *output-LTR* method based on GF_y

In *SISO*-case the two methods are the same.

Model Predictive Control (MPC)

Literature:

- **Wang, L, Model Predictive Control System Design and Implementation Using MATLAB, Springer, 2009.**
- Maciejowski, J. M., Predictive Control, with Constraints, Pearson Education, 2002.
- Rawlings, J. B., and Mayne, D. Q., Model Predictive Control, Theory and Design, Nob Hill, 2009.


LET'S START AT THE BEGINNING. A MANIFOLD IS A TOPOLOGICAL SPACE, WITH A STRUCTURE OF CHARTS ...
I DON'T WANT TO LEARN THIS! IT'S COMPLETELY IRRELEVANT TO MY LIFE!
THIS ISN'T IRRELEVANT. ALL CONTROL ENGINEERS NEED TO KNOW THIS.
I DON'T! I CAN GET ALONG FINE WITHOUT NONLINEAR CONTROL.
OH YEAH? HOW ARE YOU GOING TO MANAGE IN THE REAL WORLD? EVERYTHING IS NONLINEAR!
THAT'S NOT TRUE! I'LL USE...
PREDICTIVE CONTROL! YEAH!
THAT'S NOT REAL CONTROL.

Remark 1.1. Note that a number of control methods which have gained popularity in industry, such as fuzzy control and neural control, do not explicitly address any of the fundamental control issues in a quantitative way at all. This is the main reason why control people do not usually take these methods seriously. Naturally, these methods will still often work at least in not too demanding applications. They also appeal to many people with a non-control background and their functioning can more easily be explained to process operators and the media.

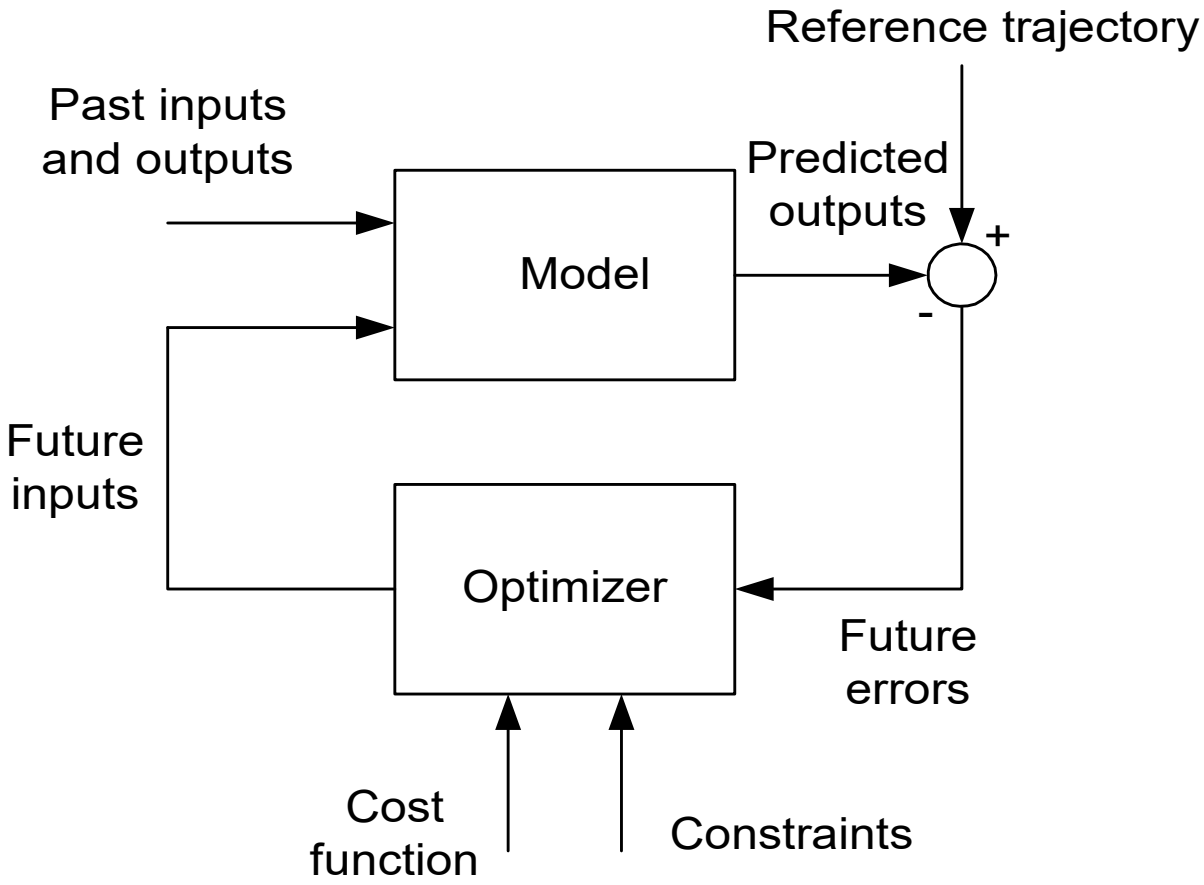
Model Predictive Control

- Can deal with constraints in a natural way
- The basic idea is easy to understand
- It extends to multivariable plants naturally
- Generally more powerful than traditional PID control
- Integrates optimal control, stochastic control, control of processes with dead-time, multivariable control, control that can handle constraints.
- A practical methodology, which has numerous technical applications, especially in the process industry.
- It was earlier neglected and critisized by the control engineering community (lack of stability proofs, robustness etc.); this situation has changed due to progress in theory.

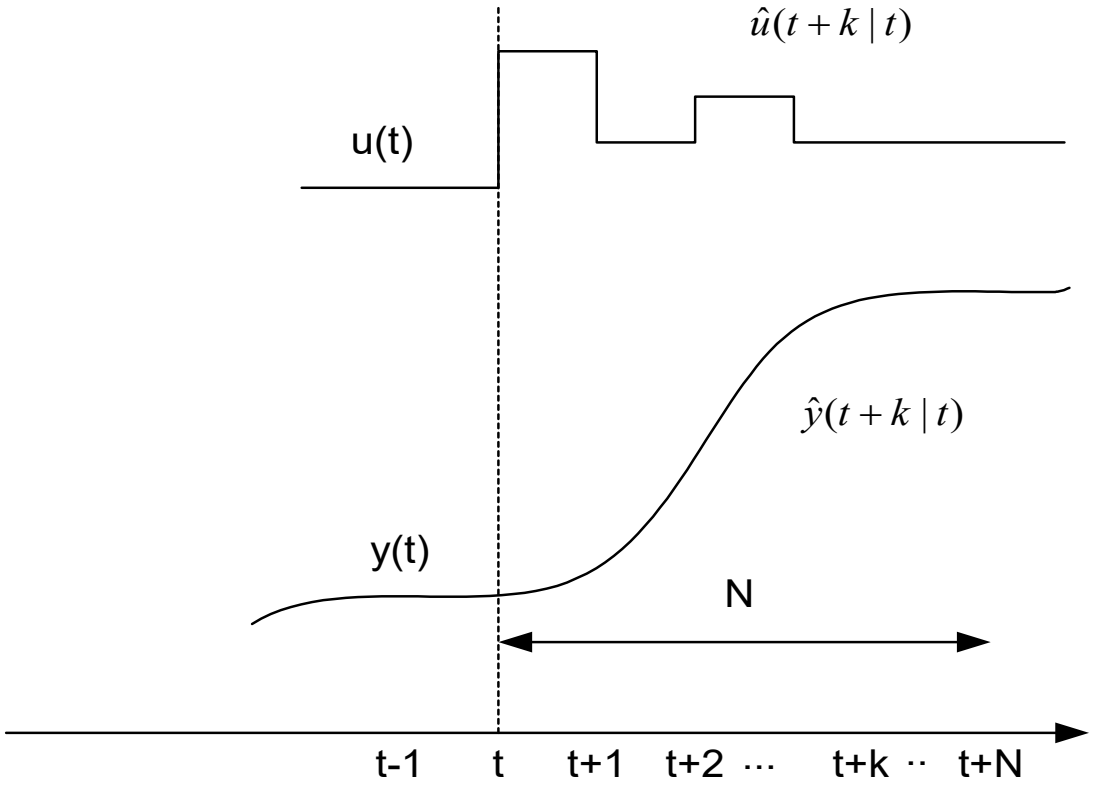
The main characteristics in MPC

- An *internal model* capable of fast simulation
- A *reference trajectory* which defines the desired closed-loop behaviour
- The *receding horizon* principle
- Future *input trajctory* by a finite number of *control moves*
- On-line *optimization* (*possibly constrained*)

Model Predictive Control



The Receding horizon principle



The output is predicted over the *prediction horizon*. Control *moves* are calculated over the *control horizon* by optimizing a criterion. Only the first move is realized; then the process is repeated.

- A lot of different formulations can be found in the literature (MPHC, MAC, DMC, EHAC, EPSAC, GPC etc. etc.)
- Maciejowski's book has information on commercial MPC products, e.g. DMCPlus, RMPCT, Connoisseur, PFC, HIECON, 3dMPC, Process Perfecter.

Model predictive control (MPC)

- Note that there are different formalisms to pose the MPC problem.
- Also, there exist software packages to do the job. The problem in using software packages ”blindly” is the lack of insight and analysis possibilities.
- For example: Matlab’s MPC toolbox is good in posing and solving problems at a reasonably high level. It is somewhat difficult to use it in research though.
- It is good to make one formalism yourself to get insight. The software packages then become easier to deal with.

Cost function

$$V(k) = \sum_{i=H_w}^{H_p} \left\| \hat{z}(k+i|k) - r(k+i|k) \right\|^2_{Q(i)} + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}(k+i|k) \right\|^2_{R(i)}$$

Prediction horizon H_p ,$Q(i), R(i)$ positive semidefinite
Control horizon H_u
Control move Δu it is assumed that the penalty is on the control moves, not controls as such

Note that if $H_w > 1$ there is no penalty immediately at time k.

The states are usually not measurable; instead we have predictions $\hat{x}(k+1|k)$ meaning that we estimate the state by using data up to time k.

Features of constrained predictive control

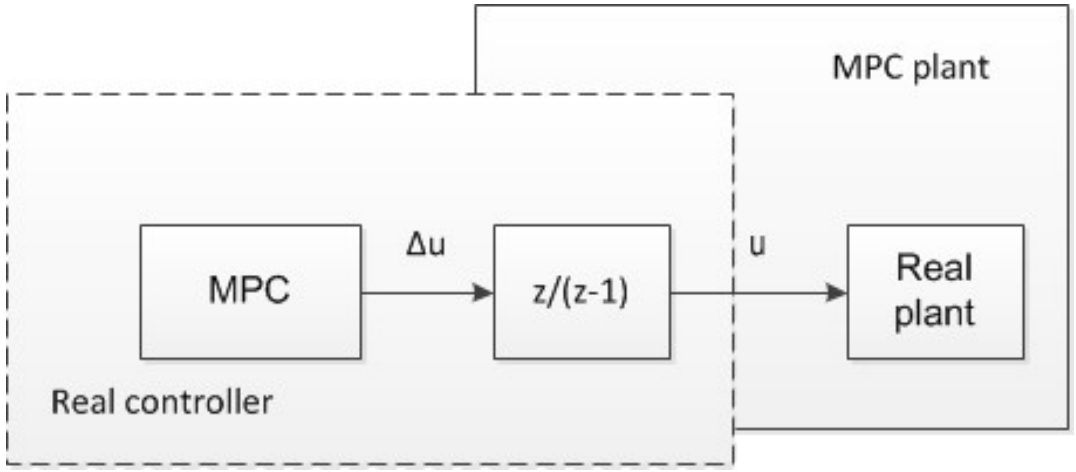
- Constraints cause MPC be *nonlinear*. But most of the time (when constraints are not near to be active) the controller operates in a linear way.
- In practice, meeting a hard constraint can be dangerous for the system. An MPC might do hazardous actions (in ”panic”); usually a supervisory mode is used to prevent such actions.
- We consider only *time-invariant* MPC. The system has then constant coefficient matrices. In

$$V(k) = \sum_{i=H_w}^{H_p} \left\| \hat{z}(k+i|k) - r(k+i|k) \right\|^2_{Q(i)} + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}(k+i|k) \right\|^2_{R(i)}$$

Q(i) and $R(i)$ can vary with i , but they must not change with k

Alternative state variable choices

- Usually the MPC gives the *control move* as its output, whereas the project model uses absolute values. The *integration* in the state space model is then needed (to create *u* from Δu



Prediction

- Predictions of the controlled variables $\hat{z}(k+i|k)$ must be obtained to solve the control problem. They are based on the best estimates of $\hat{x}(k|k)$ and the assumed future inputs (or the latest input $u(t\text{-}1)$)
- The predictor can be seen as a ”tuning parameter” in the MPC problem, because it plays a key role in the performance of the controller.
- We are actually specifying a model of the environment in which the plant is operating

- Assuming that the states are measurable and there are *no disturbances* we get

Prediction

$$\hat{x}(k+1|k) = Ax(k) + B\hat{u}(k|k)$$

$$\hat{x}(k+2|k) = A\hat{x}(k+1|k) + B\hat{u}(k+1|k) = A^2x(k) + AB\hat{u}(k|k) + B\hat{u}(k+1|k)$$

$$\vdots$$

$$\hat{x}(k+H_p|k) = A\hat{x}(k+H_p-1|k) + B\hat{u}(k+H_p-1|k)$$

$$= A^{H_p}x(k) + A^{H_p-1}B\hat{u}(k|k) + \cdots + B\hat{u}(k+H_p-1|k)$$

But $\hat{u}(k+i|k) = \hat{u}(k+H_u-1|k), \quad H_u \le i \le H_p - 1$ and earlier control moves will be studied only. So

$$\hat{u}(k|k) = \Delta\hat{u}(k|k) + u(k-1)$$

$$\hat{u}(k+1|k) = \Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1)$$

$$\vdots$$

$$\hat{u}(k+H_u-1|k) = \Delta\hat{u}(k+H_u-1|k) + \ldots + \Delta\hat{u}(k|k) + u(k-1)$$

Prediction

- Hence we get

$$\hat{x}(k+1|k) = Ax(k) + B\left[\Delta\hat{u}(k|k) + u(k-1)\right]$$

$$\hat{x}(k+2|k) = A^2x(k) + AB\left[\Delta\hat{u}(k|k) + u(k-1)\right] + B\underbrace{\left[\Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1)\right]}_{\hat{u}(k+1|k)}$$

$$= A^2x(k) + (A+I)B\Delta\hat{u}(k|k) + B\,\Delta\hat{u}(k+1|k) + (A+I)Bu(k-1)$$

$$\vdots$$

$$\hat{x}(k+H_u|k) = A^{H_u}x(k) + (A^{H_u-1} + \ldots + A + I)B\Delta\hat{u}(k|k)$$

$$\ldots + B\Delta\hat{u}(k+H_u-1|k) + (A^{H_u-1} + \ldots + A + I)Bu(k-1)$$

Prediction

$$\hat{x}(k+H_u+1|k) = A^{H_u+1}x(k) + (A^{H_u}+\ldots+A+I)B\Delta\hat{u}(k|k)$$
$$\ldots+(A+I)B\Delta\hat{u}(k+H_u-1|k)+(A^{H_u}+\ldots+A+I)Bu(k-1)$$
$$\vdots$$
$$\hat{x}(k+H_p|k) = A^{H_p}x(k) + (A^{H_p-1}+\ldots+A+I)B\Delta\hat{u}(k|k)$$
$$\ldots+(A^{H_p-H_u}+\ldots+A+I)B\Delta\hat{u}(k+H_u-1|k)+(A^{H_p-1}+\ldots+A+I)Bu(k-1)$$

We can collect everything in a matrix-vector form

Prediction

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_u|k) \\ \hat{x}(k+H_u+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} A \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix} x(k) + \begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^i B \\ \sum_{i=0}^{H_u} A^i B \\ \vdots \\ \sum_{i=0}^{H_p-1} A^i B \end{bmatrix} u(k-1) +$$

← past

$$\begin{bmatrix} B & \cdots & 0 \\ AB+B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^i B & \cdots & B \\ \sum_{i=0}^{H_u} A^i B & \cdots & AB+B \\ \vdots & \vdots & \vdots \\ \sum_{i=0}^{H_p-1} A^i B & \cdots & \sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix} \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}$$

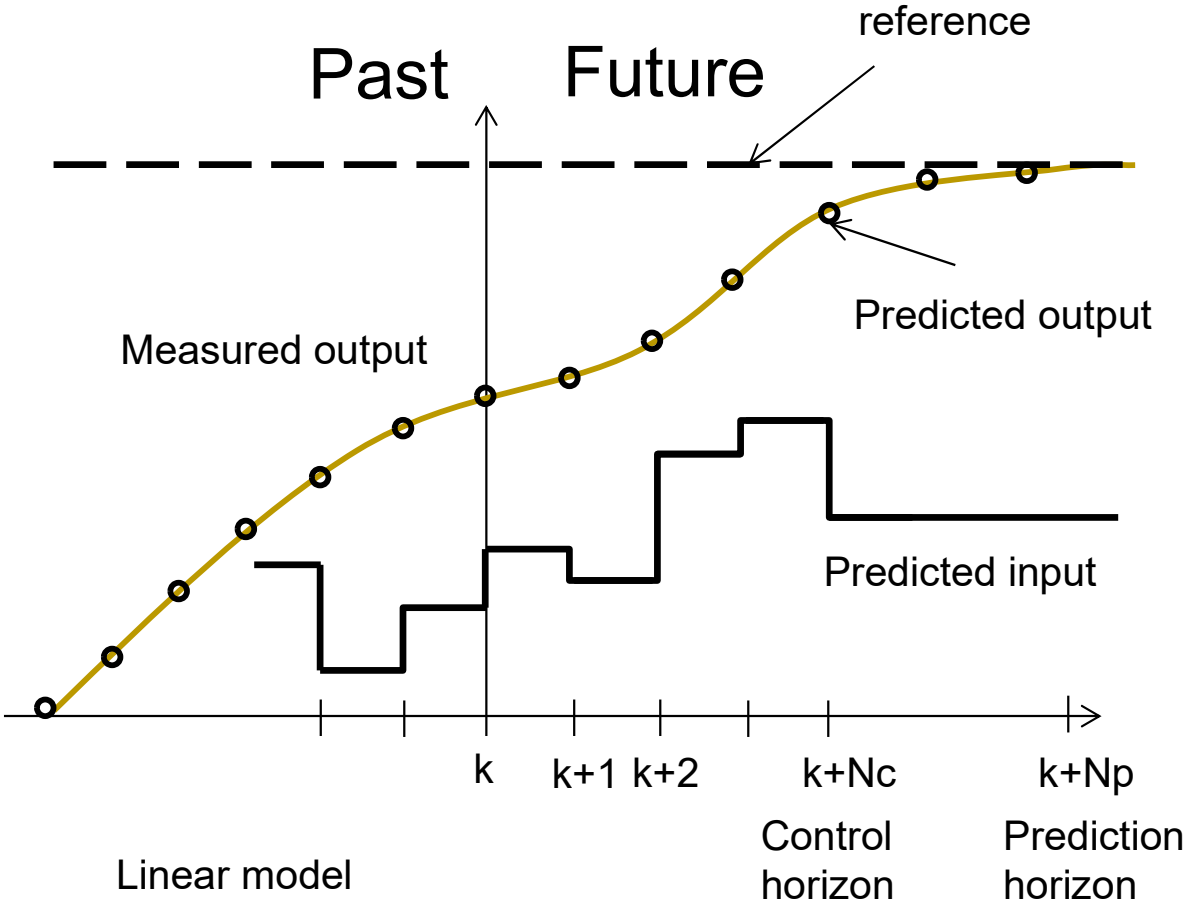
← future

Prediction

- The predictions are now obtained simply as

$$\begin{bmatrix} \hat{z}(k+1|k) \\ \vdots \\ \hat{z}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} C_z & 0 & \cdots & 0 \\ 0 & C_z & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_z \end{bmatrix} \begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix}$$

A different fotmulation



Linear model

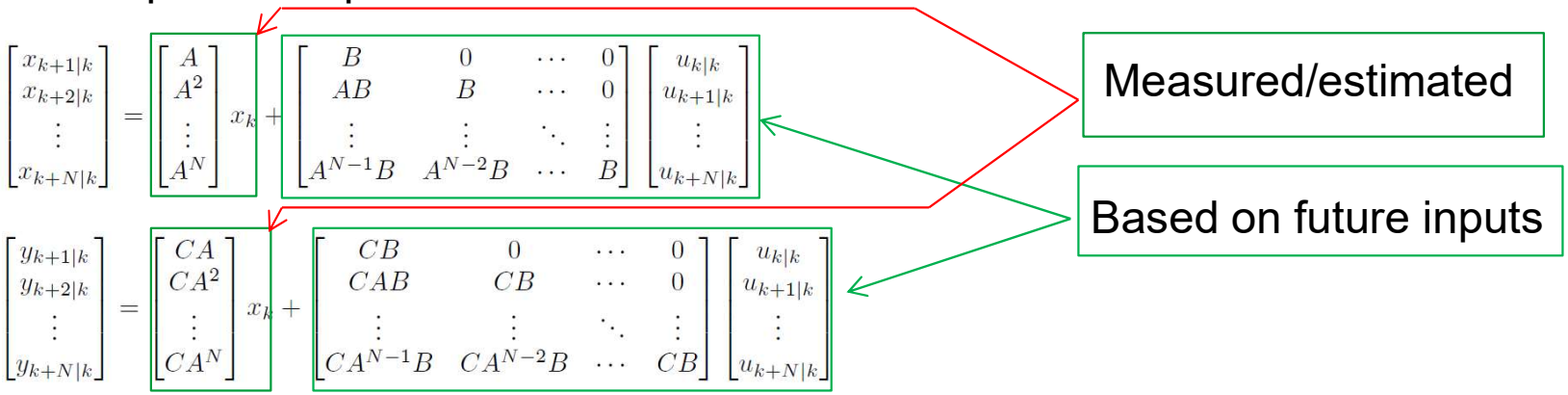
$$x[k+1] = Ax[k] + Bu[k] + Gw[k]$$
$$y[k] = Cx[k] + v[k]$$
$$E\{w[k]w[k]^T\} = Q, E\{v[k]v[k]^T\} = R, E\{w[k]v[k]^T\} = 0,$$

- Design steps:

1. Process – first principles nonlinear
2. Linear model of that process (ss, tf, etc.)
3. N-steps ahead prediction model
4. State estimator
5. Performance index
6. Optimization

Model predictive control

3. N-steps ahead prediction model



$$\begin{bmatrix} x_{k+1|k} \\ x_{k+2|k} \\ \vdots \\ x_{k+N|k} \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x_k + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+N|k} \end{bmatrix}$$

$$\begin{bmatrix} y_{k+1|k} \\ y_{k+2|k} \\ \vdots \\ y_{k+N|k} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x_k + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+N|k} \end{bmatrix}$$

$$x_{k+1} = P_x x_k + H_x u_k$$
$$y_{k+1} = P x_k + H u_k$$

4. Kalman observer

$$\hat{x}[k|k] = \hat{x}[k|k-1] + M(y_m[k] - \hat{y}_m[k])$$
$$\hat{x}[k+1|k] = A\hat{x}[k|k] + Bu[k]$$
$$\hat{y}[k] = C\hat{x}[k|k-1]$$
$$M = PC^T(CPC^T + R)^{-1}$$

Model predictive control

Tracking error

Input penalty

Input rate penalty

5. Performance index

$$J = \sum_{k=1}^{n_y} \|W_y e_k\|_2^2 + \sum_{k=1}^{n_u} \|W_u(u_k - u_{ss})\|_2^2 + \|R_u \Delta u_k\|_2^2$$

6. Optimization problem

$$\min_{u_{k|k} \dots u_{k+N-1|k}} \sum_{i=0}^{N-1} \left(\sum_{j=1}^{n_y} \|W_y(y_j(k+i+1|k) - r_j(k+i+1))\|_2^2 + \sum_{j=1}^{n_u} \|W_u(u_j(k+i|k) - u_{ss,j}(k+i))\|_2^2 \right)$$

$$\text{s.t.} \quad u_{j,min}(i) < u_j(k+i|k) < u_{j,max}(i)$$

J

To continue:

- Read chapters 1 and 2 in Wang's book to become convenient with one formalism.
- The rest of Wang's book is interesting and useful. To continue studies in MPC, I would start from it.

The formalism in Wang's book

Process model

$$\dot{x}_m(t) = Ax_m(t) + Bu(t)$$
$$y(t) = Cx_m(t)$$

Discretized form

$$x_m(k+1) = A_m x_m(k) + B_m u(k)$$
$$y(k) = C_m x_m(k)$$

Form the difference

$$x_m(k+1) - x_m(k) = A_m\left[x_m(k) - x_m(k-1)\right] + B_m\left[u(k) - u(k-1)\right]$$

and define the variables

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k)$$
$$\Delta x_m(k) = x_m(k) - x_m(k-1)$$
$$\Delta u(k) = u(k) - u(k-1)$$

It follows that

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k)$$

Denote $x(k) = \begin{bmatrix} \Delta x_m^T(k) & y(k) \end{bmatrix}^T$ which leads first to

$$y(k+1) - y(k) = C_m \left[x_m(k+1) - x_m(k) \right] = C_m \Delta x_m(k+1)$$
$$= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k)$$

and finally to

$$\underbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}_{x(k+1)} = \underbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}_{x(k)} + \underbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}_{B} \Delta u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}_{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}; \qquad o_m = \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}}_{n_1}$$

Now, remember

$$\begin{bmatrix} y_{k+1|k} \\ y_{k+2|k} \\ \vdots \\ y_{k+N|k} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x_k + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+N|k} \end{bmatrix}$$

or

$$Y = Fx(k_i) + \Phi \Delta U$$

Written generally in the MIMO case

$$\underbrace{\begin{bmatrix}\Delta x(k+1)\\ y(k+1)\end{bmatrix}}_{x_a(k+1)} = \underbrace{\begin{bmatrix}A & \mathbb{0}_{n_s\times n_y}\\ CA & \mathbb{1}_{n_y\times n_y}\end{bmatrix}}_{A_a}\underbrace{\begin{bmatrix}\Delta x(k)\\ y(k)\end{bmatrix}}_{x_a(k)} + \underbrace{\begin{bmatrix}B\\ CB\end{bmatrix}}_{B_a}\Delta u(k)$$

$$y(k) = \begin{bmatrix}\mathbb{0}_{n_s\times n_y} & \mathbb{1}_{n_y\times n_y}\end{bmatrix}\underbrace{\begin{bmatrix}\Delta x(k)\\ y(k)\end{bmatrix}}_{C_a}$$

$$Y = \begin{bmatrix}y(k_i+1|k_i) & y(k_i+2|k_i) & \dots & y(k_i+N_p|k_i)\end{bmatrix}$$

$$\Delta U = \begin{bmatrix}\Delta u(k_i) & \Delta u(k_i+1) & \dots & \Delta u(k_i+N_c-1)\end{bmatrix}$$

$$Y = Fx_a(k_i) + \Phi\Delta U$$

$$\Phi = \begin{bmatrix} C_a B_a & 0 & \cdots & 0 \\ C_a A_a B_a & C_a B_a & \cdots & 0 \\ C_a A_a^2 B_a & C_a A_a B_a & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ C_a A_a^{N_p-1} B_a & C_a A_a^{N_p-2} B_a & \cdots & C_a A_a^{N_p-N_u} B_a \end{bmatrix} \qquad F = \begin{bmatrix} C_a A_a \\ C_a A_a^2 \\ \vdots \\ C_a A_a^{N_p} \end{bmatrix}$$

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U \quad R_s^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} r(k_i) = \bar{R}_s^{\,T} r(k_i)$$

$$J = \left[R_s - Fx(k_i)\right]^T \left[R_s - Fx(k_i)\right] - 2\Delta U^T \Phi^T \left(R_s - Fx(k_i)\right) + \Delta U^T \left(\Phi^T \Phi + \bar{R}\right) \Delta U$$

To find the minimum without constraints

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T \left[R_s - Fx(k_i)\right] + 2\left[\Phi^T \Phi + \bar{R}\right] \Delta U = 0$$

$$\Rightarrow \Delta U = \left(\Phi^T \Phi + \bar{R}\right)^{-1} \Phi^T \left(R_s - Fx(k_i)\right)$$

- Generally (with constraints)

$$\min_{\Delta u_{k|k} \ldots \Delta u_{k+N_u|k}} J,$$
$$\text{s.t. } A_{ineq}\Delta U \leq b_{ineq}$$

leads to a numerical optimization problem, for which efficient algorithms exist.

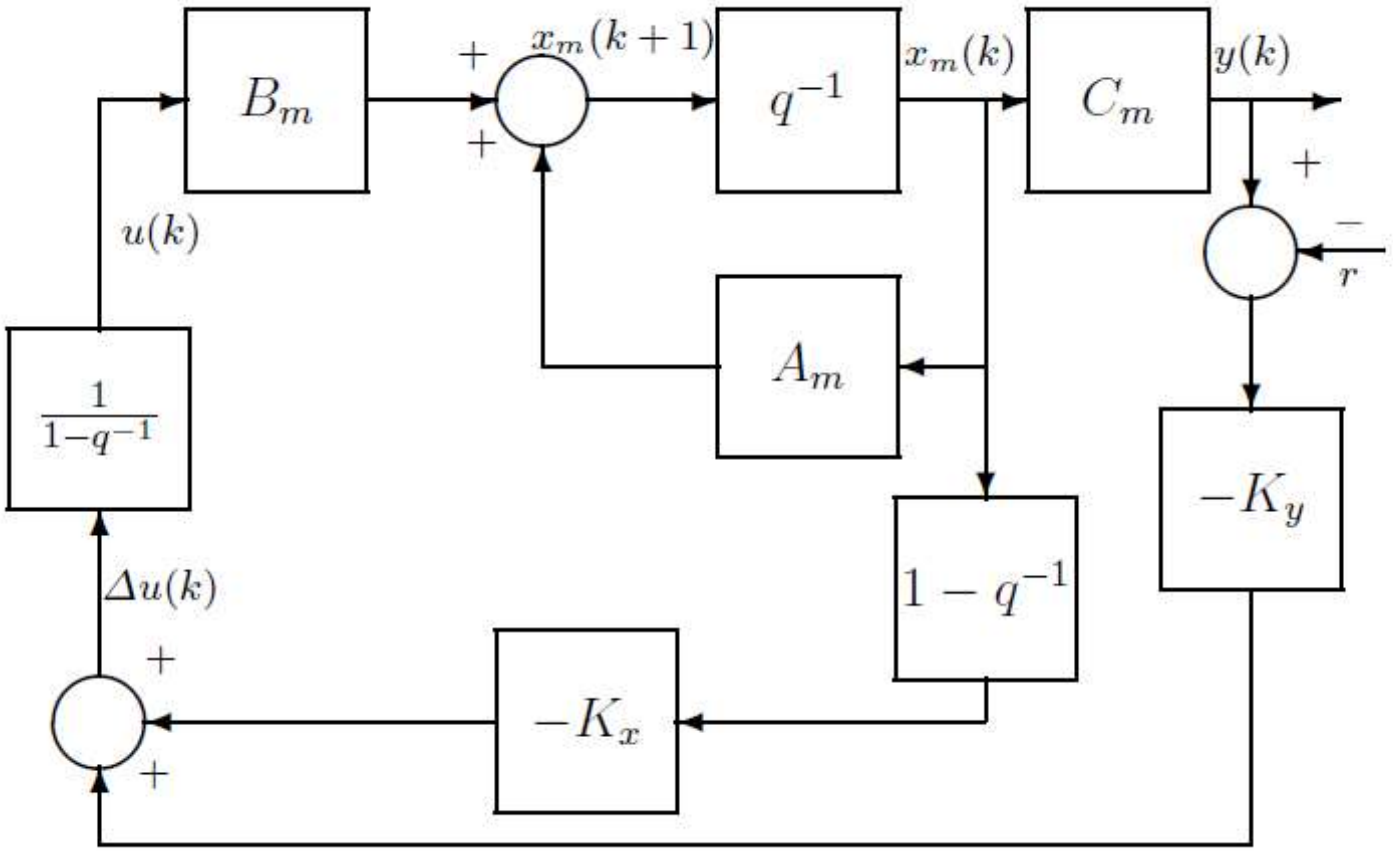
Note that the idea has been to formulate the whole MPC problem such that it can be solved by general optimization software. See e.g. the command *quadprog* in Matlab.

Using a special MPC toolbox is possible of course, but it is impossible to see ”inside” what it really does.

```
function[F,Phi,Phi_Phi,Phi_F,Phi_R,BarRs,A_e,
B_e,C_e]=mpcgain2(Ap,Bp,Cp,Nc,Np)
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
%
A_e=eye(n1+m1,n1+m1);
%Forming the augmented model
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;

B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;

C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

```
h(1:m1,:)=C_e;
F(1:m1,:)=C_e*A_e;
for kk=2:Np
h((kk-1)*m1+1:kk*m1,:)=h((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
F((kk-1)*m1+1:kk*m1,:)= F((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
end
v=h*B_e;
Phi=zeros(m1*Np,n_in*Nc); %declare the dimension of Phi
Phi(1:(m1*Np),1:n_in)=v; % first column of Phi


for i=2:Nc
Phi(:,((i-1)*n_in+1):(i*n_in))=[zeros((i-1)*m1,n_in);v(1:(Np-(i-1))*m1,:)];%Toeplitz matrix
end


BarRs=zeros(m1*Np,m1);
for i=1:m1
BarRs(((i-1)*Np+1):i*Np,i)=1;
end
```

The Receding horizon solution



$$\Delta u(k_i) = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \left(\Phi^T \Phi + \bar{R} \right)^{-1} \left(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i) \right)$$

$$= K_y r(k_i) - K_{mpc} x(k_i) = K_y r(k_i) - \begin{bmatrix} K_x & K_y \end{bmatrix} x(k_i)$$

Note $x = \begin{bmatrix} \Delta x^T & y^T \end{bmatrix}^T$

Constraints

Constraints must be related to the control variable ΔU.

The inequalities $\Delta U^{min} \leq \Delta U \leq \Delta U^{max}$ are equal to

$$-\Delta U \leq -\Delta U^{min}$$
$$\Delta U \leq \Delta U^{max}$$

or

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}$$

Note that

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} u(k_i-1) + \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ I & I & 0 & \dots & 0 \\ I & I & I & \dots & 0 \\ \vdots & & & & \\ I & I & \dots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix}.$$

which can be written as

$$-(C_1 u(k_i - 1) + C_2 \Delta U) \leq -U^{min}$$
$$(C_1 u(k_i - 1) + C_2 \Delta U) \leq U^{max},$$

Similarly $$Y^{min} \leq Fx(k_i) + \Phi \Delta U \leq Y^{max}.$$

In short, minimize

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U,$$

under the inequality constraints

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix},$$

where the data matrices are

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix};\ N_1 = \begin{bmatrix} -U^{min} + C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix};\ M_2 = \begin{bmatrix} -I \\ I \end{bmatrix};$$

$$N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix};\ M_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix};\ N_3 = \begin{bmatrix} -Y^{min} + Fx(k_i) \\ Y^{max} - Fx(k_i) \end{bmatrix}.$$

or $M\Delta U \leq \gamma,$

Matlab's command *quadprog* can for example be used.

Final steps

- Today's lecture no 12 (1.12) is the last lecture. The 11th exercise was the last exercise. The last homework no 6 has been published.

- Second Intermediate exam (IE2) on Thursday 8.12, 14:00-16:00, room T3 (T-house).
- First full exam (Kurssitentti) on Tuesday 13.12, 13:00-16:00, hall AS1.

- No registrations are needed for these exams. You can participate in both if you wish. Intermediate exams cannot be repeated. For the full exams later (next: January 2023) you have to register.
- For the grading of the course, see lecture slides, Chapter 1. (If you participate in both intermediate and full exam, then the better of (IE1+IE2), (Full exam), counts.)

Contents

- Classical control theory: SISO-systems, linear or linearized system models
- Extension to multivariable (MIMO) systems
- Performance and limitations of control
- Uncertainty and robustness,
- IMC-control,
- LQ and LQG control
- Optimal control
- Introduction to Model Predictive Control

The end

Good luck for the future!