Chapter 8

- **Introduction to Model Predictive Control**

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 DONT 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.

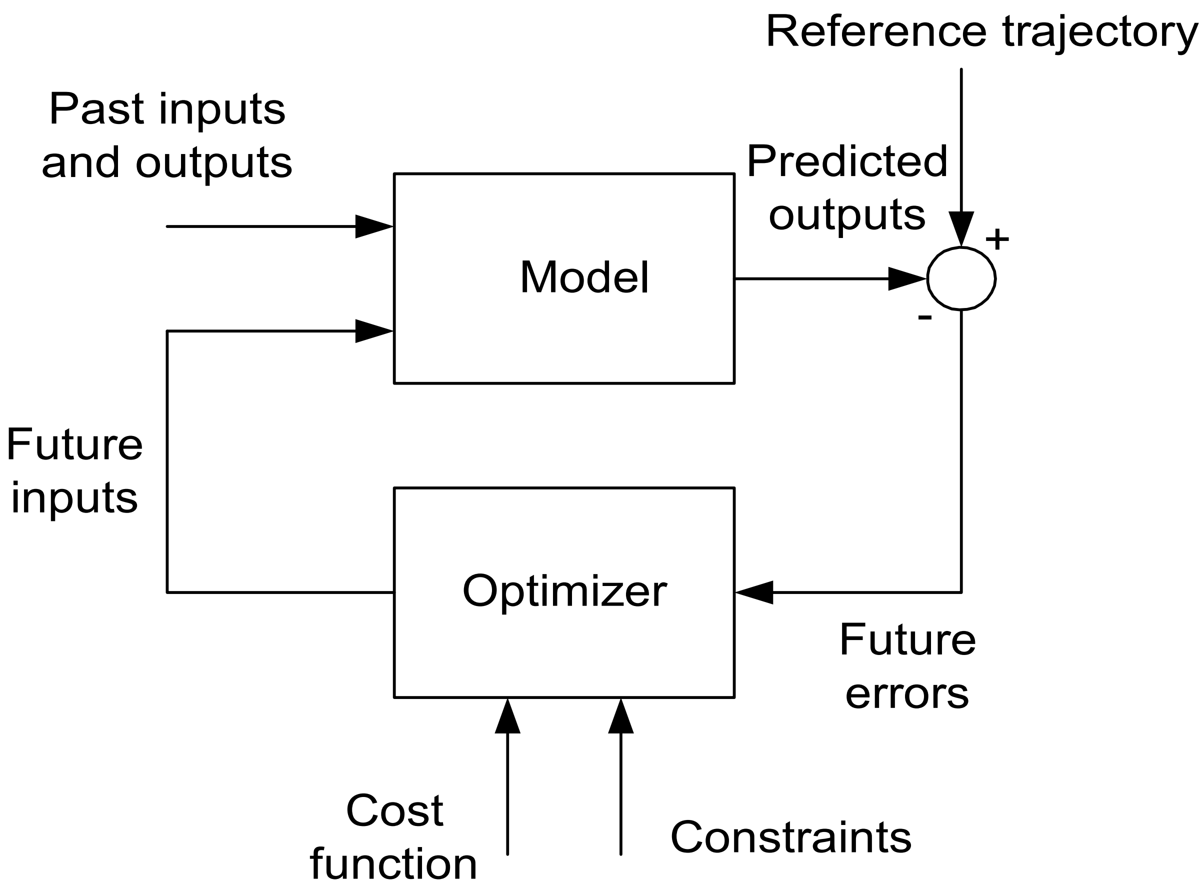
Model Predictive Control

- 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

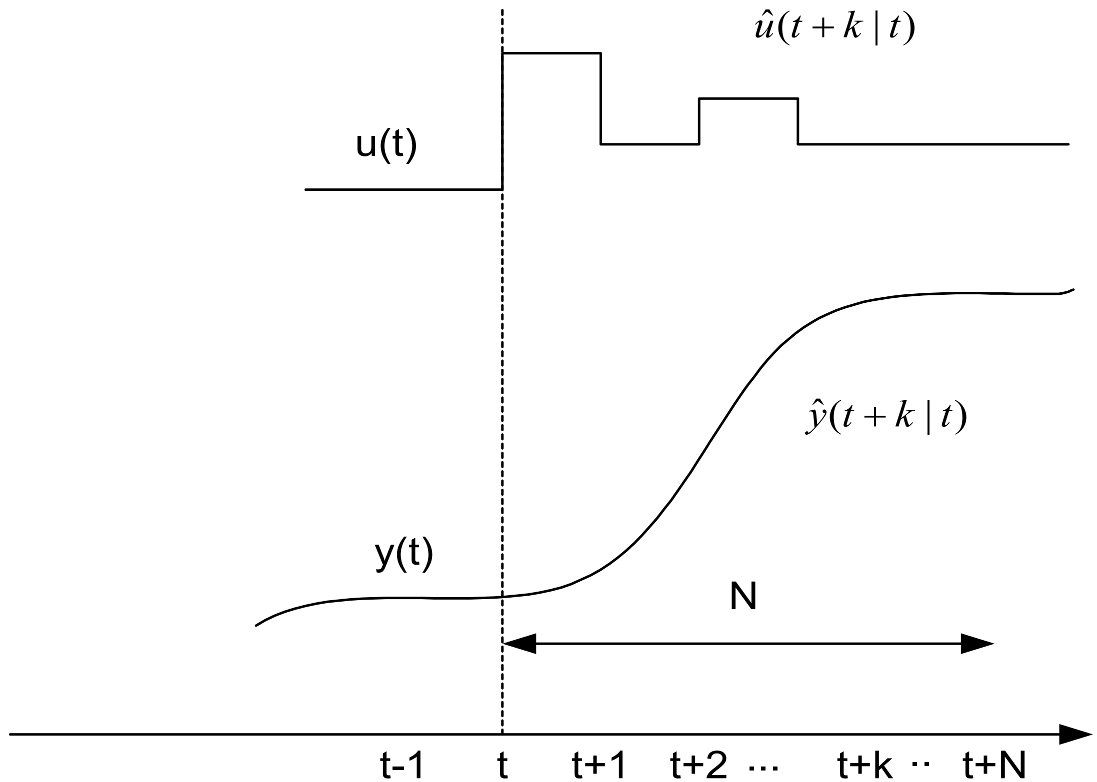
- An *internal model* capable of fast simulation
- A *reference trajectory* which defines the desired closed-loop behaviour
- The *receding horizon* principle
- Future *input trajctory* by a finite number of *control moves*
- On-line *optimization* (*possibly constrained*)

Model Predictive Control



Aalto University
School of Science
and Technology

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

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

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

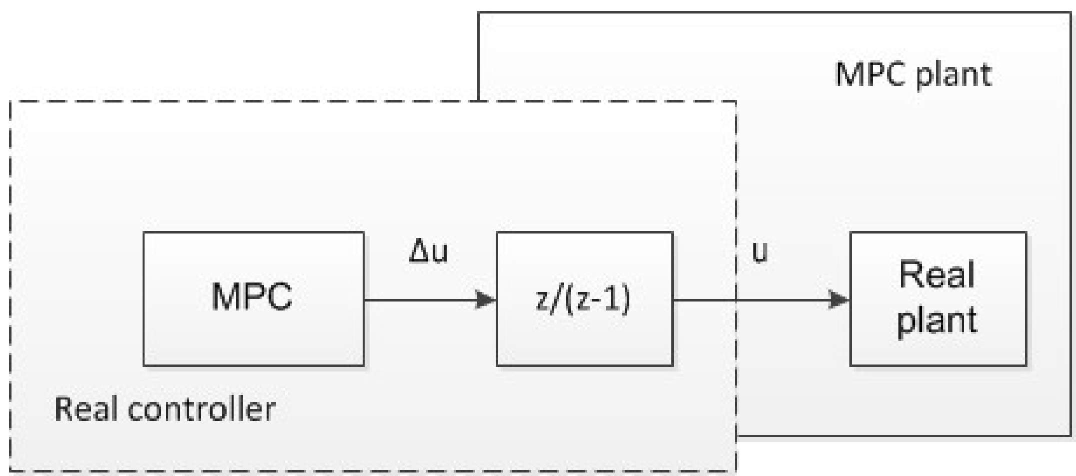
- Constraints cause MPC be *nonlinear*. But most of the time (when constraints are not near to be active) the controller operates in a linear way.
- In practice, meeting a hard constraint can be dangerous for the system. An MPC might do hazardous actions (in ”panic”); usually a supervisory mode is used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

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

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

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

$\leftarrow$ past

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

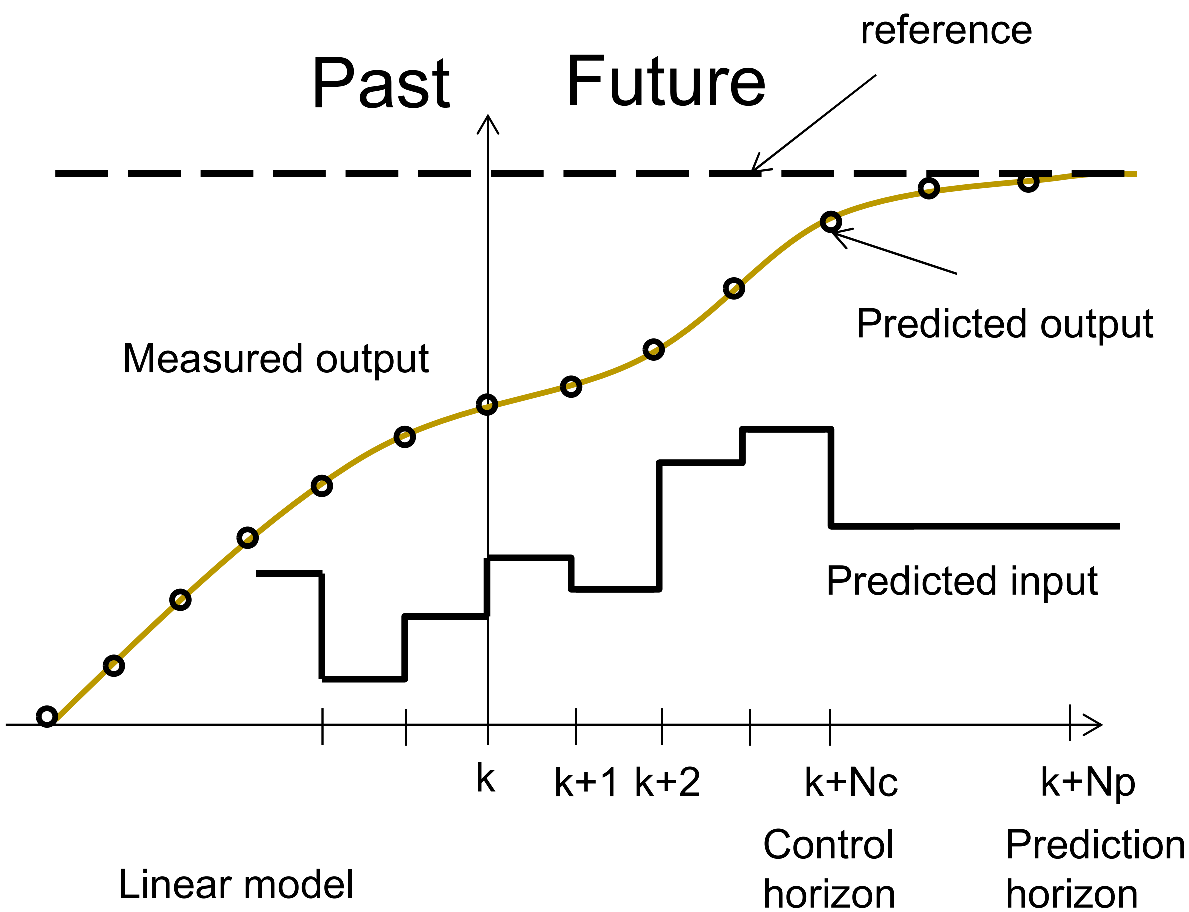
$\leftarrow$ future

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

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

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

Measured/estimated

Based on future inputs

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

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 $x(k) = \left[\Delta x_m^T(k) \quad y(k) \right]^T$ 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

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

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

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

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

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

$$Y = \left[y(k_i+1|k_i) \; y(k_i+2|k_i) \; \ldots \; y(k_i+N_p|k_i) \right]$$

$$\Delta U = \left[\Delta u(k_i) \; \Delta u(k_i+1) \; \ldots \; \Delta u(k_i+N_c-1) \right]$$

$$Y = F x_a(k_i) + \Phi \Delta U$$

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

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

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

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);
```

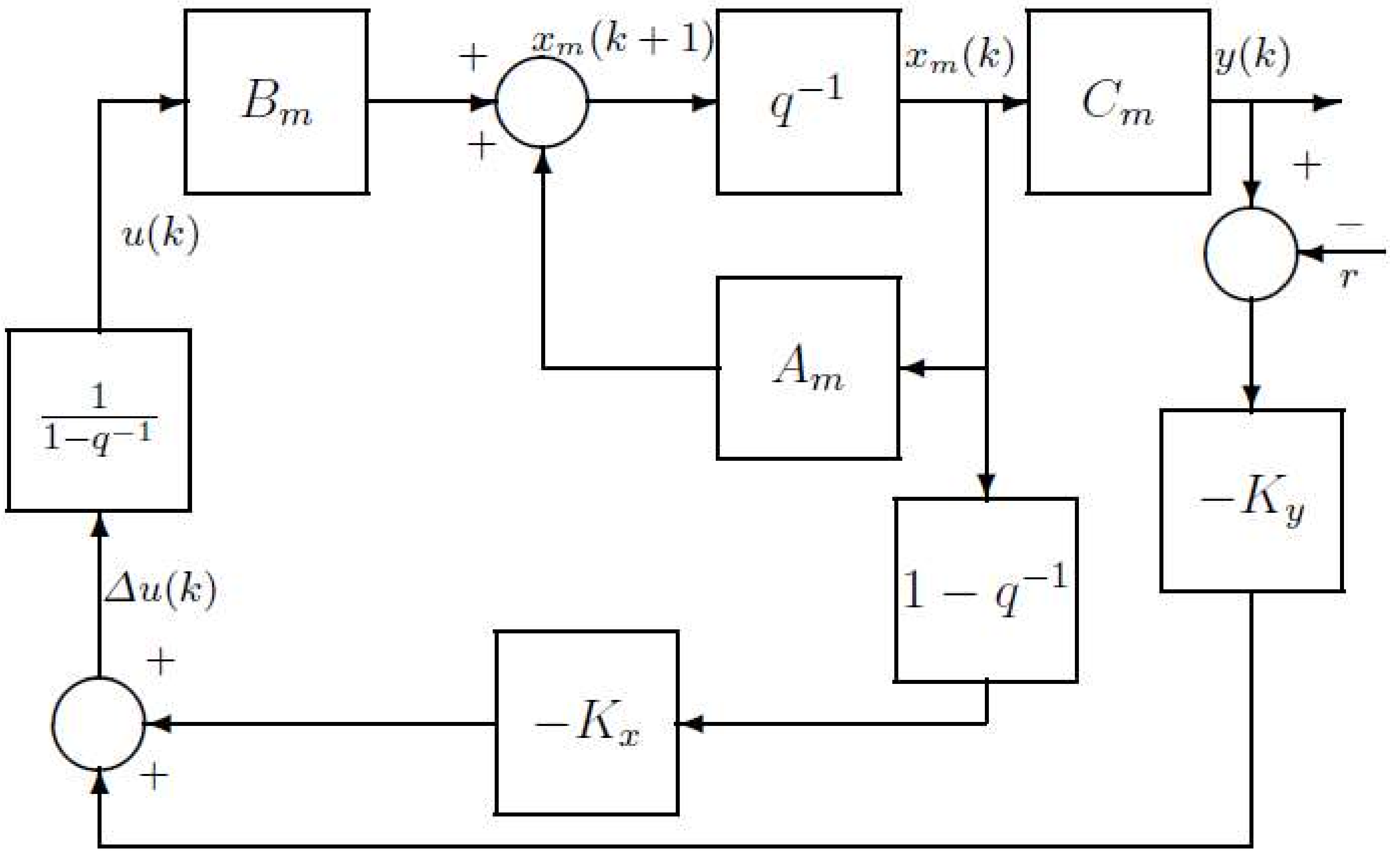
```
h(1:m1,:)=C_e;
F(1:m1,:)=C_e*A_e;
for kk=2:Np
h((kk-1)*m1+1:kk*m1,:)=h((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
F((kk-1)*m1+1:kk*m1,:)= F((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
end
v=h*B_e;
Phi=zeros(m1*Np,n_in*Nc); %declare the dimension of Phi
Phi(1:(m1*Np),1:n_in)=v; % first column of Phi


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
```

Aalto University
School of Science
and Technology

The Receding horizon solution



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

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

Aalto University
School of Science
and Technology

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.