Data-Driven MPC

Why bother with a model?

Alexandre F.B.

2A SG8 MPC – CentraleSupélec

2nd May 2024

Latin Notations

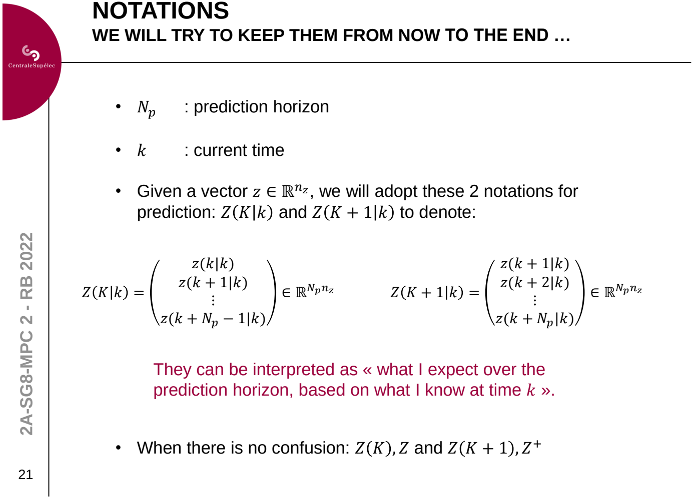


Figure 1: Recall from Romain's slides

Notations and algebra for today

- L: trajectory length, t: current time, N_p: prediction horizon
- Given a vector $z \in \mathbb{R}^{n_z}$, we will adopt the following notations:
  - z_k the value of z at time k
  - $z_{[k,k+\ell]} = \begin{pmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+\ell} \end{pmatrix} \in \mathbb{R}^{(\ell+1)n_z}$ the "stacked window"

Notations and algebra for today

- L: trajectory length, t: current time, N_p: prediction horizon
- Given a vector $z \in \mathbb{R}^{n_z}$, we will adopt the following notations:
  - z_k the value of z at time k
  - $z_{[k,k+\ell]} = \begin{pmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+\ell} \end{pmatrix} \in \mathbb{R}^{(\ell+1)n_z}$ the "stacked window"
  - $z_k(t)$ predicted/calculated value of z_{t+k}, from information available at time t

$\hookrightarrow$ $z_{[0,\ell]}(t)$ "what I expect on the interval $[t+0, t+\ell]$, based on what I know at time t"

Notations and algebra for today

- L: trajectory length, t: current time, N_p: prediction horizon
- Given a vector $z \in \mathbb{R}^{n_z}$, we will adopt the following notations:
  - z_k the value of z at time k
  - $z_{[k,k+\ell]} = \begin{pmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+\ell} \end{pmatrix} \in \mathbb{R}^{(\ell+1)n_z}$ the "stacked window"
  - $z_k(t)$ predicted/calculated value of z_{t+k}, from information available at time t

  $\hookrightarrow$ $z_{[0,\ell]}(t)$ "what I expect on the interval $[t+0, t+\ell]$, based on what I know at time t"
- Given a matrix M, we have its Moore-Penrose inverse $M^\dagger$
  - If $Mx = b$ has solution(s) for x, then $M^\dagger b$ is a solution
- Given a matrix $M \succ 0$ and a vector z: $\|z\|_M^2 := z^\top M z$

Model-Based State Prediction

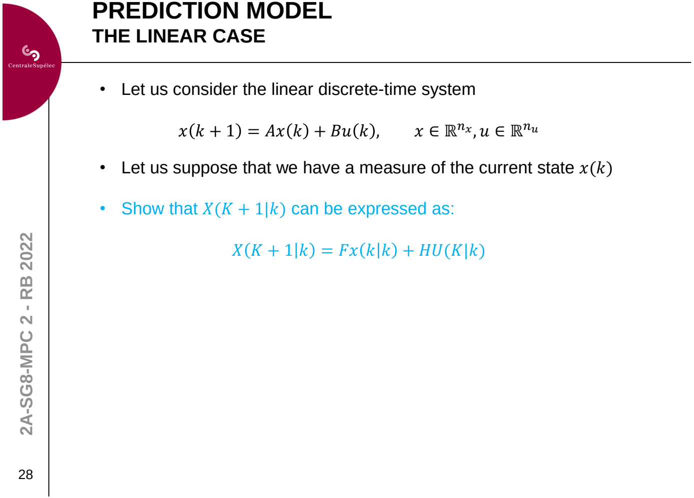


Figure 2: Recall from Romain's slides

Brick 1: Model-Based Output Prediction

- Linear discrete-time system:

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{cases} \quad ; \quad x \in \mathbb{R}^{n_x}, u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y}$$

- Suppose x_k (and u) is known:

$$y_{[k,k+L-1]} = \mathcal{O}_L \, x_k + \mathcal{H}_L \, u_{[k,k+L-1]}$$

$$\begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+L-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{L-1} \end{bmatrix} x_k + \begin{bmatrix} D & 0 & \dots & 0 \\ CB & D & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{L-2}B & CA^{L-3}B & \dots & D \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+L-1} \end{bmatrix}$$

About Linearity

- We know 2 "trajectories":
  - $x_0^1, u_{[0,L-1]}^1, y_{[0,L-1]}^1$
  - $x_0^2, u_{[0,L-1]}^2, y_{[0,L-1]}^2$
- We want to compute the output sequence $y_{[0,L-1]}^3$, corresponding to
  - state $x_0^3 = x_0^1 + x_0^2$
  - input sequence $u_{[0,L-1]}^3 = u_{[0,L-1]}^1 + u_{[0,L-1]}^2$

  but we don't know $\mathcal{O}_L$ nor $\mathcal{H}_L$...

About Linearity

- We know 2 "trajectories":
  - $x_0^1, u^1_{[0,L-1]}, y^1_{[0,L-1]}$
  - $x_0^2, u^2_{[0,L-1]}, y^2_{[0,L-1]}$
- We want to compute the output sequence $y^3_{[0,L-1]}$, corresponding to
  - state $x_0^3 = x_0^1 + x_0^2$
  - input sequence $u^3_{[0,L-1]} = u^1_{[0,L-1]} + u^2_{[0,L-1]}$

  but we don't know $\mathcal{O}_L$ nor $\mathcal{H}_L$...
- Thanks to linearity, we have:

$$\begin{aligned}
y^3_{[0,L-1]} &= \mathcal{O}_L x_0^3 + \mathcal{H}_L u^3_{[0,L-1]} \\
&= \mathcal{O}_L (x_0^1 + x_0^2) + \mathcal{H}_L (u^1_{[0,L-1]} + u^2_{[0,L-1]}) \\
&= (\mathcal{O}_L x_0^1 + \mathcal{H}_L u^1_{[0,L-1]}) + (\mathcal{O}_L x_0^2 + \mathcal{H}_L u^2_{[0,L-1]}) \\
&= y^1_{[0,L-1]} + y^2_{[0,L-1]}
\end{aligned}$$

Brick 2: An alternative to the State Observer

- Suppose now that $\mathcal{S}$ is observable, which by definition means:

$$\forall n \geqslant n_x, \operatorname{rank}(\mathcal{O}_n) = \operatorname{rank}\left(\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n_x$$

$\hookrightarrow$ $\mathcal{O}_n x_0 = y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}$ is a system of linear equations, with a unique solution [1] x_0

[1]Assuming $y_{[0,n-1]}$ is a possible output of the system

Brick 2: An alternative to the State Observer

- Suppose now that $\mathcal{S}$ is observable, which by definition means:

$$\forall n \geqslant n_x, \operatorname{rank}(\mathcal{O}_n) = \operatorname{rank}\left(\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n_x$$

↪ $\mathcal{O}_n x_0 = y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}$ is a system of linear equations, with a unique solution [1] $x_0 = \mathcal{O}_n^{\dagger}\left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}\right)$

↪ This is a "static" alternative to dynamic state observers: it is called MHE (Moving Horizon Estimator).

↪ It is linear too!

[1] Assuming $y_{[0,n-1]}$ is a possible output of the system

Model-Based Input-Output Representation

- Up until now, we have (with $L > n \geqslant n_x$):
  - Output prediction: $y_{[0,L-1]} = \mathcal{O}_L x_0 + \mathcal{H}_L u_{[0,L-1]}$
  - State estimation: $x_0 = \mathcal{O}_n^{\dagger} \left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]} \right)$

Model-Based Input-Output Representation

- Up until now, we have (with $L > n \geqslant n_x$):
  - Output prediction: $y_{[0,L-1]} = \mathcal{O}_L x_0 + \mathcal{H}_L u_{[0,L-1]}$
  - State estimation: $x_0 = \mathcal{O}_n^\dagger \left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}\right)$
- Let's plug all this together:

$$y_{[0,L-1]} = \mathcal{O}_L \mathcal{O}_n^\dagger \left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}\right) + \mathcal{H}_L u_{[0,L-1]}$$

- For convenience we can write (same n; $N_p = L - n \geqslant 1$):

$$y_{[-n,N_p-1]}(t) = \mathcal{O}_{N_p+n} \mathcal{O}_n^\dagger \left(y_{[-n,-1]}(t) - \mathcal{H}_n u_{[-n,-1]}(t)\right) + \mathcal{H}_L u_{[-n,N_p-1]}(t)$$

Model-Based Input-Output Representation

- Up until now, we have (with $L > n \geqslant n_x$):
  - Output prediction: $y_{[0,L-1]} = \mathcal{O}_L x_0 + \mathcal{H}_L u_{[0,L-1]}$
  - State estimation: $x_0 = \mathcal{O}_n^\dagger \left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}\right)$
- Let's plug all this together:

$$y_{[0,L-1]} = \mathcal{O}_L \mathcal{O}_n^\dagger \left(y_{[0,n-1]} - \mathcal{H}_n u_{[0,n-1]}\right) + \mathcal{H}_L u_{[0,L-1]}$$

- For convenience we can write (same n; $N_p = L - n \geqslant 1$):

$$y_{[-n,N_p-1]}(t) = \mathcal{O}_{N_p+n} \mathcal{O}_n^\dagger \left(y_{[-n,-1]}(t) - \mathcal{H}_n u_{[-n,-1]}(t)\right) + \mathcal{H}_L u_{[-n,N_p-1]}(t)$$

- For convenience (again) we rewrite:

$$\begin{aligned} y_{[0,N_p-1]}(t) &= \mathcal{A}_L^u u_{[-n,-1]}(t) + \mathcal{A}_L^y y_{[-n,-1]}(t) + \mathcal{B}_L u_{[0,N_p-1]}(t) \\ &= \mathcal{A}_L \begin{bmatrix} u_{[-n,-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix} + \mathcal{B}_L u_{[0,N_p-1]}(t) \end{aligned}$$

Back onto Linearity

- From before:

$$y_{[0,N_p-1]}(t) = \mathcal{A}_L \begin{bmatrix} u_{[-n,-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix} + \mathcal{B}_L u_{[0,N_p-1]}(t)$$

- Given initial conditions $\begin{bmatrix} u_{[-n,-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix}$ and "future" inputs $u_{[0,N_p-1]}(t)$, we want to compute future output $y_{[0,N_p-1]}(t)$.

Back onto Linearity

- From before:

$$y_{[0,N_p-1]}(t) = \mathcal{A}_L \begin{bmatrix} u_{[-n,-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix} + \mathcal{B}_L u_{[0,N_p-1]}(t)$$

- Given initial conditions $\begin{bmatrix} u_{[-n,-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix}$ and "future" inputs $u_{[0,N_p-1]}(t)$, we want to compute future output $y_{[0,N_p-1]}(t)$.
- If we know input-output trajectories $(u^i_{[-n,N_p-1]}, y^i_{[-n,N_p-1]})_{i=1,2}$ such that

$$\begin{cases} u_{[-n,-1]}(t) & = u^1_{[-n,-1]} + u^2_{[-n,-1]} \\ y_{[-n,-1]}(t) & = y^1_{[-n,-1]} + y^2_{[-n,-1]} \end{cases} \quad ; \quad u_{[0,N_p-1]}(t) = u^1_{[0,N_p-1]} + u^2_{[0,N_p-1]}$$

$\hookrightarrow$ Future output is (similarly as before):

$$y_{[0,N_p-1]}(t) = y^1_{[0,N_p-1]} + y^2_{[0,N_p-1]}$$

A Non-Parametric Model?

- Say we have a lot of input-output trajectories $(u^i_{[-n,N_p-1]}, y^i_{[-n,N_p-1]})_{i=1:K}$, we can store them in data matrices:

$$U = \begin{bmatrix} u^1_{[-n,N_p-1]} & u^2_{[-n,N_p-1]} & \cdots & u^K_{[-n,N_p-1]} \end{bmatrix}$$
$$Y = \begin{bmatrix} y^1_{[-n,N_p-1]} & y^2_{[-n,N_p-1]} & \cdots & y^K_{[-n,N_p-1]} \end{bmatrix}$$

$\hookrightarrow$ For any vector α, if

$$\begin{cases} u_{[-n,N_p-1]}(t) & = U\alpha \\ y_{[-n,N_p-1]}(t) & = Y\alpha \end{cases} \iff \begin{bmatrix} U \\ Y \end{bmatrix} \alpha = \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix}$$

then $(u_{[-n,N_p-1]}(t), y_{[-n,N_p-1]}(t))$ is a valid trajectory.

A Non-Parametric Model?

- Say we have a lot of input-output trajectories $(u^i_{[-n,N_p-1]}, y^i_{[-n,N_p-1]})_{i=1:K}$, we can store them in data matrices:

$$U = \begin{bmatrix} u^1_{[-n,N_p-1]} & u^2_{[-n,N_p-1]} & \cdots & u^K_{[-n,N_p-1]} \end{bmatrix}$$
$$Y = \begin{bmatrix} y^1_{[-n,N_p-1]} & y^2_{[-n,N_p-1]} & \cdots & y^K_{[-n,N_p-1]} \end{bmatrix}$$

$\hookrightarrow$ For any vector α, if

$$\begin{cases} u_{[-n,N_p-1]}(t) & = U\alpha \\ y_{[-n,N_p-1]}(t) & = Y\alpha \end{cases} \iff \begin{bmatrix} U \\ Y \end{bmatrix} \alpha = \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix}$$

then $(u_{[-n,N_p-1]}(t), y_{[-n,N_p-1]}(t))$ is a valid trajectory.

- When is the reverse true? If we have a trajectory, when can we find a corresponding α?

$\hookrightarrow$ Depends on the "diversity" of data...

One More Notation...

The Hankel Matrix

$z \in \mathbf{R}^{n_z}$

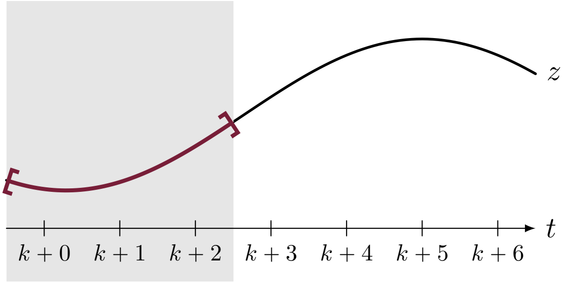
$$H_L(z_{[k,k+N-1]}) = \begin{bmatrix} z_{[k,k+L-1]} & z_{[k+1,k+L]} & \cdots & z_{[k+N-L,k+N-1]} \end{bmatrix}$$
$$= \begin{pmatrix} z_k & z_{k+1} & \cdots & z_{k+N-L} \\ z_{k+1} & z_{k+2} & \cdots & z_{k+N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{k+L-1} & z_{k+L} & \cdots & z_{k+N-1} \end{pmatrix} \in \mathcal{M}_{(Ln_z)\times(N-L+1)}(\mathbb{R})$$

One More Notation...

The Hankel Matrix, visually

With $L = 3$ and $N = 7$:

$$H_3(z_{[k,k+6]}) = \begin{pmatrix} z_{k+0} & z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} \\ z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} \\ z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} & z_{k+6} \end{pmatrix}$$

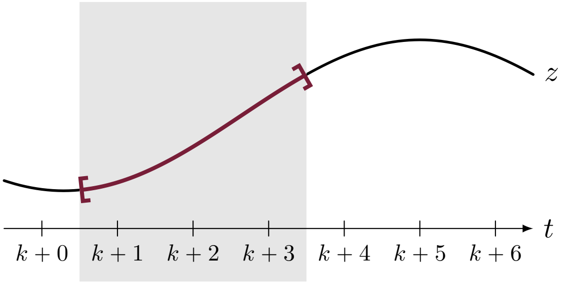


One More Notation...

The Hankel Matrix, visually

With $L = 3$ and $N = 7$:

$$H_3(z_{[k,k+6]}) = \begin{pmatrix} z_{k+0} & z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} \\ z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} \\ z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} & z_{k+6} \end{pmatrix}$$

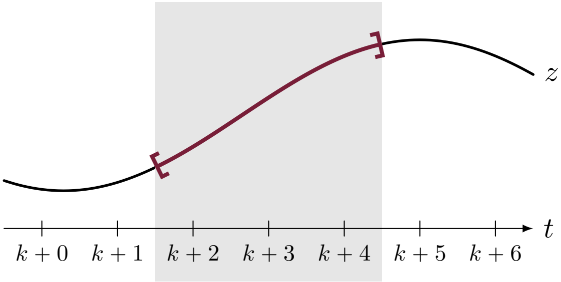


One More Notation...

The Hankel Matrix, visually

With $L = 3$ and $N = 7$:

$$H_3(z_{[k,k+6]}) = \begin{pmatrix} z_{k+0} & z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} \\ z_{k+1} & z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} \\ z_{k+2} & z_{k+3} & z_{k+4} & z_{k+5} & z_{k+6} \end{pmatrix}$$



Willems' Lemma

Theorem

Let $\mathcal{S}$ a linear time-invariant, controllable and observable system. Let u^d, y^d a trajectory of $\mathcal{S}$ of length N (u^d input, y^d output), such that u^d is Persistently Exciting of order $L+n$. Then, with again u input, y output:

$$\forall \begin{bmatrix} u \\ y \end{bmatrix} \text{ trajectory of } \mathcal{S}, \quad \exists \alpha \in \mathbb{R}^{N-L+1}, \begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha$$

Willems' Lemma

Theorem

Let $\mathcal{S}$ a linear time-invariant, controllable and observable system. Let u^d, y^d a trajectory of $\mathcal{S}$ of length N (u^d input, y^d output), such that u^d is Persistently Exciting of order $L+n$. Then, with again u input, y output:

$$\forall \begin{bmatrix} u \\ y \end{bmatrix} \text{ trajectory of } \mathcal{S}, \quad \exists \alpha \in \mathbb{R}^{N-L+1}, \begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha$$

Definition (Persistence of Excitation)

$u_{[k,k+N-1]}$, such that $u_k \in \mathbb{R}^{n_u}$, is Persistently Exciting of order $L+n$ if:

$$\mathrm{rank}(H_{L+n}(u_{[k,k+N-1]})) = n_u \times (L+n)$$

Consequence : $N \geqslant (n_u+1) \times (L+n) - 1$.

A Non-Parametric Model

- It is now established that: if u^d is such that $\mathrm{rank}(H_{L+n}(u^d)) = (L+n)n_u$, and we take $U = H_L(u^d), Y = H_L(y^d)$ then

$$\exists \alpha, \begin{bmatrix} U \\ Y \end{bmatrix} \alpha = \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix} \iff \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix} \text{ is a trajectory.}$$

A Non-Parametric Model

- It is now established that: if u^d is such that $\mathrm{rank}(H_{L+n}(u^d)) = (L+n)n_u$, and we take $U = H_L(u^d), Y = H_L(y^d)$ then

$$\exists\alpha, \begin{bmatrix} U \\ Y \end{bmatrix} \alpha = \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix} \iff \begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,N_p-1]}(t) \end{bmatrix} \text{ is a trajectory.}$$

- We can "cut" the data matrix in two parts:

$$Y = \begin{bmatrix} Y_p \\ Y_f \end{bmatrix} = \begin{bmatrix} y^1_{[-n,-1]} & y^2_{[-n,-1]} & \cdots & y^K_{[-n,-1]} \\ y^1_{[0,N_p-1]} & y^2_{[0,N_p-1]} & \cdots & y^K_{[0,N_p-1]} \end{bmatrix}$$

- Then, using linearity (for prediction *and* MHE):

$$\begin{bmatrix} u_{[-n,N_p-1]}(t) \\ y_{[-n,-1]}(t) \end{bmatrix} = \begin{bmatrix} U \\ Y_p \end{bmatrix} \alpha \implies y_{[0,N_p-1]}(t) = Y_f \alpha$$

Model Predictive Control

Problem (Control)

At each time t, compute $\bar{u}(t)$ that minimizes cost:

$$J(\bar{u}_{[0,N_p-1]}(t), \bar{y}_{[0,N_p-1]}(t))$$

such that

$$\text{Dynamics: } \begin{cases} \bar{x}_{i+1}(t) & = A\bar{x}_i(t) + B\bar{u}_i(t) \\ \bar{y}_i(t) & = C\bar{x}_i(t) + D\bar{u}_i(t) \end{cases}, \quad 0 \leqslant i < N_p$$

Initial state: $\bar{x}_0(t) = x_t$

Then, we apply $u_t = \bar{u}_0(t)$.

Data-Driven Model Predictive Control

Problem (Data-driven control)

At each time t, compute $\alpha(t)$ that minimizes cost:

$$J(\bar{u}_{[0,N_p-1]}(t), \bar{y}_{[0,N_p-1]}(t))$$

such that

$$\text{Behavior: } \begin{bmatrix} \bar{u}_{[-n,N_p-1]}(t) \\ \bar{y}_{[-n,N_p-1]}(t) \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha(t)$$

$$\text{Initial conditions: } \begin{bmatrix} \bar{u}_{[-n,-1]}(t) \\ \bar{y}_{[-n,-1]}(t) \end{bmatrix} = \begin{bmatrix} u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \end{bmatrix}$$

Then, we apply $u_t = \bar{u}_0(t)$.

Recall, $L = N_p + n$.

Data-Driven Model Predictive Control

Problem (Data-driven control)

At each time t, compute $\alpha(t)$ that minimizes cost:

$$J(\bar{u}_{[0,N_p-1]}(t), \bar{y}_{[0,N_p-1]}(t)) + \lambda_\varepsilon \|\varepsilon(t)\|_2^2$$

such that

$$\text{Behavior: } \begin{bmatrix} \bar{u}_{[-n,N_p-1]}(t) \\ \bar{y}_{[-n,N_p-1]}(t) \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha(t)$$

$$\text{Noisy Initial conditions: } \begin{bmatrix} \bar{u}_{[-n,-1]}(t) \\ \bar{y}_{[-n,-1]}(t) + \varepsilon(t) \end{bmatrix} = \begin{bmatrix} u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \end{bmatrix}$$

Then, we apply $u_t = \bar{u}_0(t)$.

Recall, $L = N_p + n$.

Data-Driven Model Predictive Control

Problem (Data-driven control)

At each time t, compute $\alpha(t)$ that minimizes regularized cost:

$$J(\bar{u}_{[0,N_p-1]}(t), \bar{y}_{[0,N_p-1]}(t)) + \lambda_\varepsilon \|\varepsilon(t)\|_2^2 + \lambda_\alpha \|\alpha(t)\|_2^2$$

such that

$$\text{Behavior: } \begin{bmatrix} \bar{u}_{[-n,N_p-1]}(t) \\ \bar{y}_{[-n,N_p-1]}(t) \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha(t)$$

$$\text{Noisy Initial conditions: } \begin{bmatrix} \bar{u}_{[-n,-1]}(t) \\ \bar{y}_{[-n,-1]}(t) + \varepsilon(t) \end{bmatrix} = \begin{bmatrix} u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \end{bmatrix}$$

Then, we apply $u_t = \bar{u}_0(t)$.

Recall, $L = N_p + n$.

Notice That...

- Only $\alpha(t)$ needs to be found! Indeed:
  - $\bar{u}(t), \bar{y}(t)$ are defined from $\alpha(t)$
  - $\varepsilon(t)$ is defined from $\bar{y}(t)$ and $y_{[t-n,t-1]}$

Notice That...

- Only $\alpha(t)$ needs to be found! Indeed:
  - $\bar{u}(t), \bar{y}(t)$ are defined from $\alpha(t)$
  - $\varepsilon(t)$ is defined from $\bar{y}(t)$ and $y_{[t-n,t-1]}$
- About the number of data samples:
  - Data u^d must be PE of order $L + n = N_p + 2n$

  $\hookrightarrow$ $N \geqslant (n_u + 1) \times (N_p + 2n) - 1$
- We have to obtain data (u^d, y^d)! There are mainly two ways:
  - *before* creating the controller,
  - with a controller in two "phases": data-gathering, then control.

Control Scheme Example

Control of a linear system with n_u inputs and n_y outputs.

- Offline: Define parameters (and cost):
  - Choose n.
  - Choose prediction horizon N_p, in conjunction with T_e the sample time.
  - Choose number of data samples $N \geqslant (n_u + 1) \times (N_p + 2n) - 1$.
  - Choose costs $J, \lambda_\alpha, \lambda_\varepsilon$.
  - Generate $u^d \in \mathbb{R}^{N n_u}$ such that $\text{rank}(H_{N_p+2n}(u^d)) = n_u \times (N_p + 2n)$: random values often work well.
- Initialization: For every time t in $[0, N-1]$:
  - Apply random input $u_t \leftarrow u_t^d$
  - Record output $y_t^d \leftarrow y_t$
- Online: As usual with MPC, for every time $t \geqslant N$:
  - Solve the control problem
  - Apply the result $\bar{u}(t)$ for 1 time step

Exercice: Data-Driven Tracking

Consider the tracking objective:

$$J(\bar{u}(t), \bar{y}(t)) = \sum_{k=0}^{N_p-1} \|\bar{y}_k(t) - y_k^r(t)\|_Q^2 + \|\bar{u}_k(t) - \bar{u}_{k-1}(t)\|_R^2$$

with constraints

$$u_{\min} \leqslant \bar{u}_k(t) \leqslant u_{\max} \quad \forall k \geqslant 0$$

where $y^r(t)$ is the reference and $\bar{u}_{-1}(t) = u_{t-1}$ is the previous input.

1. Write the data-driven control problem in terms of $\alpha(t)$.
2. Implement with these values:
   - $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; R = 0.1Q$
   - $\lambda_\alpha = 0.1; \lambda_\varepsilon = 10$

Choosing Parameters

- Choice of n:
  - Remember, $n \geqslant n_x$: an estimate of n_x is nice to have.
  - Contrarily to an identified system, a “large” n can be good: it gives more information to the MHE.
- Choice of N_p:
  - As for model-based MPC: a larger value increases control performance, but takes more compute time.
- Choice of N:
  - As said before: $N \geqslant (n_u + 1) \times (N_p + 2n) - 1$.
  - A larger value “filters” uncertainties, but is more expensive computationally.
- Choice of $J, \lambda_\alpha, \lambda_\varepsilon$:
  - λ_α reduces the risk of "overfit": it should increase with noise variance.
  - λ_ε reduces "misfit": it should increase with inverse of noise variance.
  - $\lambda_\alpha, \lambda_\varepsilon$ increase together with "model quality": it needs to be balanced against the control cost J.