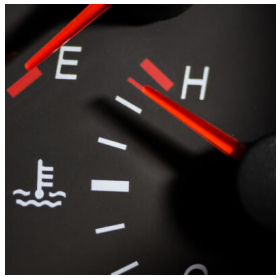
Computational Control

Data-Driven Predictive Control Part II

Saverio Bolognani and Keith Moffat

Automatic Control Laboratory (IfA)
ETH Zurich

Recap from last week:









ECB raises interest rates across eurozone by record margin

European Central Bank sets aside recession fears to increase rates by 0.75 of a percentage point to 1.25%

The ECB president, Christine Lagarde, in Frankfurt on Thursday. The bank had to respond forcefully to criticism of falling behind the curve, said one investment strategist of the rates decision. Photograph: Kai Pfaffenbach/Reuters

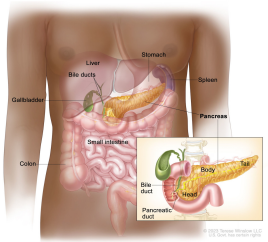
The European Central Bank has raised interest rates across the eurozone by a record margin to combat soaring inflation that has reached double figures in some of the currency bloc's 19 member countries.

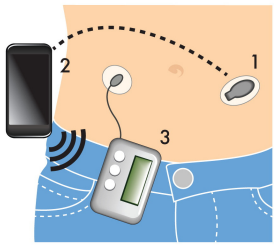




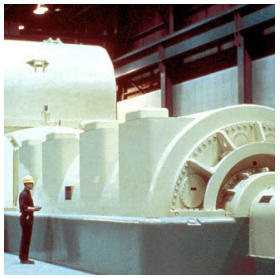








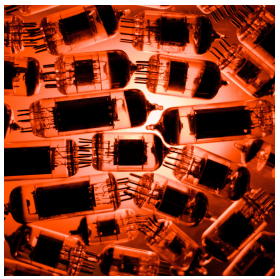












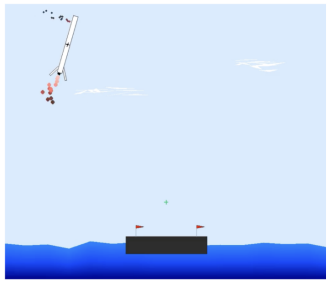
How can we control these systems if we do not have a system model?

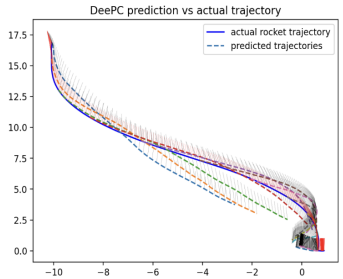
Purpose 1 of this week's lecture:

Getting Data-Driven Predictive Control to work in practice.

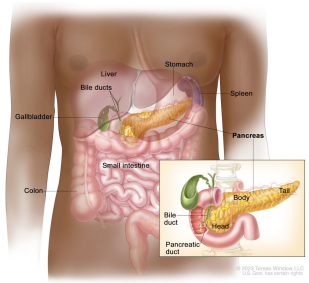
Example 1: Rocket landing:

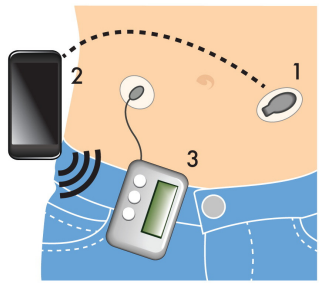




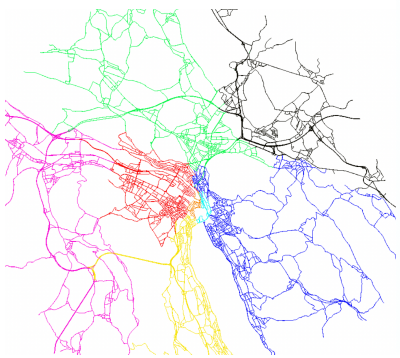


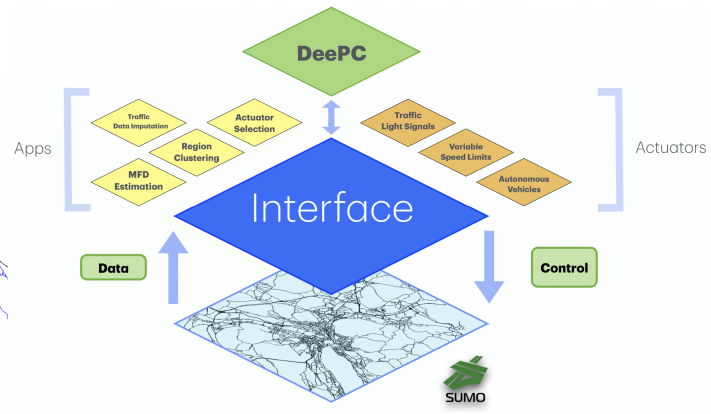
Example 2: Artificial pancreas control:





Example 3: Traffic control:





Purpose 2 of this week's lecture:

Demonstrating different LTI representations

State Space

$$x_{t+1} = Ax_t + Bu_t$$
$$y_t = Cx_t$$

Subspace

$$\text{span}\{\Lambda_L\}$$

or

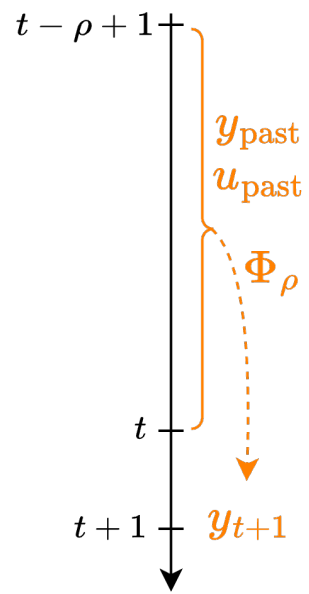
$$\text{span}\left\{\begin{bmatrix}\mathcal{H}_L(\mathbf{u}_{\text{data}})\\ \mathcal{H}_L(\mathbf{y}_{\text{data}})\end{bmatrix}\right\}$$

or

$$\text{span}\left\{\begin{bmatrix}I_{L+\rho}\\ \widehat{S}\end{bmatrix}\right\}$$

ARX

$$y_k = \Phi_\rho \begin{bmatrix}y_{\text{past}}\\ u_{\text{past}}\end{bmatrix}$$



Any of these representations can be used for data-driven predictive control.

Recap from last week: LTI System Response

$$y_t = \underbrace{CA^t x_0}_{\text{free response}} + \underbrace{\sum_{j=0}^{t-1} CA^{t-1-j} B u_j}_{\text{convolution}}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{L-1} \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{L-1} \end{bmatrix}}_{\mathcal{O}_L} x_0 + \underbrace{\begin{bmatrix} 0 & 0 & \dots & & & \\ CB & 0 & \dots & & & \\ CAB & CB & 0 & \dots & & \\ \vdots & & & & & \\ CA^{L-2}B & CA^{L-3}B & \dots & CAB & CB & 0 \end{bmatrix}}_{\mathcal{G}_L} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{L-1} \end{bmatrix}$$

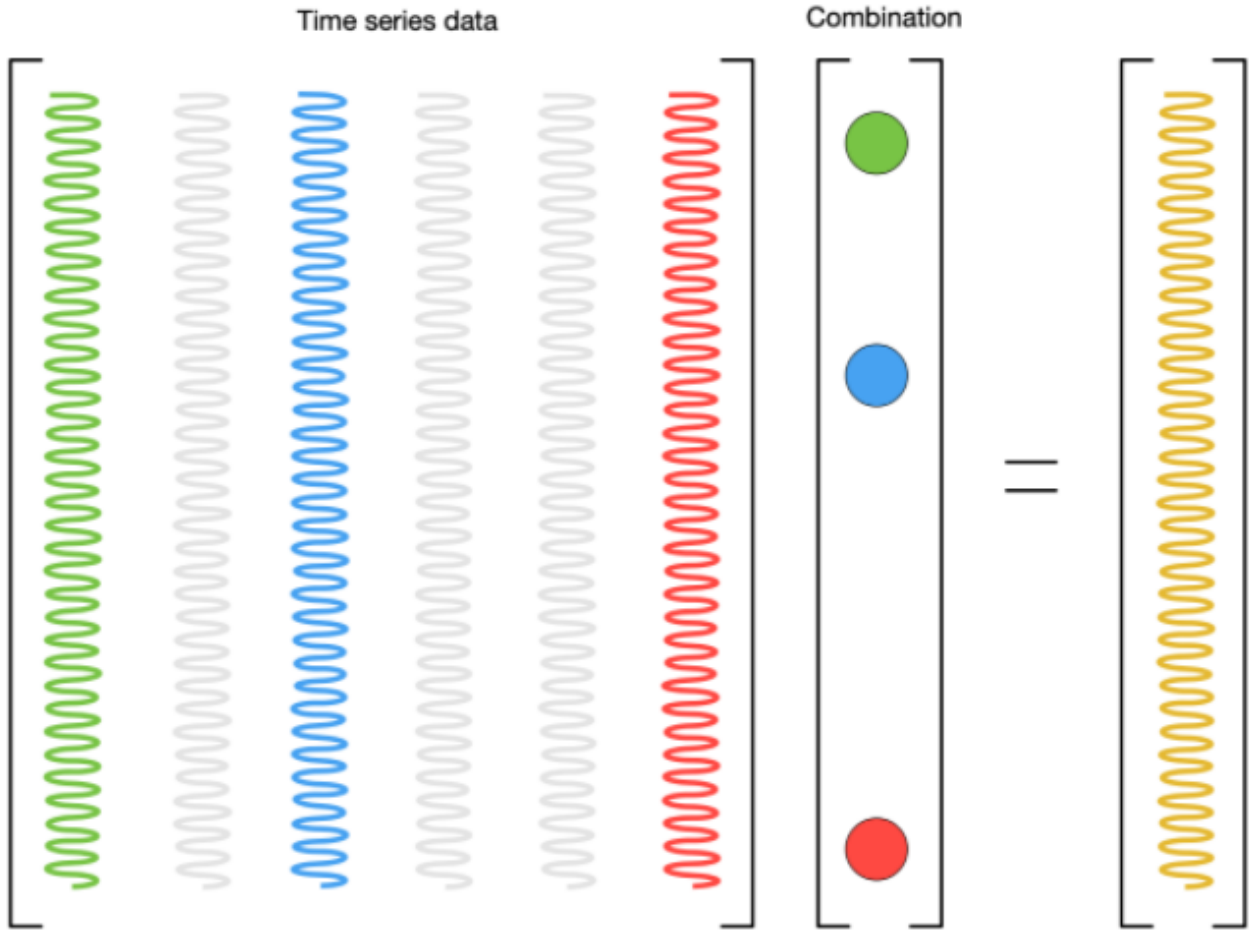
- $\mathcal{O}_L \in \mathbb{R}^{L \times n}$ (tall) is the **extended observability matrix**
- $\mathcal{G}_L \in \mathbb{R}^{L \times L}$ (square) is the **convolution matrix**

Question: Do we need A, B, and C for multistep prediction?

Answer (from last lecture): No, we can also predict multiple steps into the future using the *subspace* defined by the training data.

(Noiseless) LTI System Subspace Representation

Any SISO I/O sequence of length $L > n$ belongs to an $\mathbb{R}^{L+n}$ subspace of $\mathbb{R}^{2L}$



- **Only** valid I/O sequences are linear combinations of (noise-free) system data
- If $\mathbf{u}_{\text{data}}$ is *persistently exciting* ($\text{rank}\left(\mathcal{H}_{L+n}(\mathbf{u}_{\text{data}})\right) = L + n$), then **all** valid I/O sequences are linear combinations of system data

(Noiseless) LTI System Subspace Representation

Any SISO I/O sequence of length $L > n$ belongs to an $\mathbb{R}^{L+n}$ subspace of $\mathbb{R}^{2L}$

For control, we split the L-length trajectories ($L = \rho + K$) into

- **past** data: the "initial condition" at the time of the control decision, and
- **future** data: the planned inputs and outputs at the time of the control decision.

$$\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix}$$

$$\mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) = \begin{bmatrix} u_1 & u_2 & \dots & u_N \\ u_2 & u_3 & \dots & u_{N+1} \\ \vdots & \vdots & & \vdots \\ u_\rho & u_{\rho+1} & \dots & u_{N+\rho-1} \end{bmatrix} \quad \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) = \begin{bmatrix} y_1 & y_2 & \dots & y_N \\ y_2 & y_3 & \dots & y_{N+1} \\ \vdots & \vdots & & \vdots \\ y_\rho & y_{\rho+1} & \dots & y_{N+\rho-1} \end{bmatrix}$$

$$\mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) = \begin{bmatrix} u_{\rho+1} & u_{\rho+2} & \dots & u_{N+\rho} \\ u_{\rho+2} & u_{\rho+3} & \dots & u_{N+\rho+1} \\ \vdots & \vdots & & \vdots \\ u_L & u_{L+1} & \dots & u_{N+L-1} \end{bmatrix} \quad \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) = \begin{bmatrix} y_{\rho+1} & y_{\rho+2} & \dots & y_{N+\rho} \\ y_{\rho+2} & y_{\rho+3} & \dots & y_{N+\rho+1} \\ \vdots & \vdots & & \vdots \\ y_L & y_{L+1} & \dots & y_{N+L-1} \end{bmatrix}$$

(Noiseless) LTI System Subspace Representation

Any SISO I/O sequence of length $L > n$ belongs to an $\mathbb{R}^{L+n}$ subspace of $\mathbb{R}^{2L}$

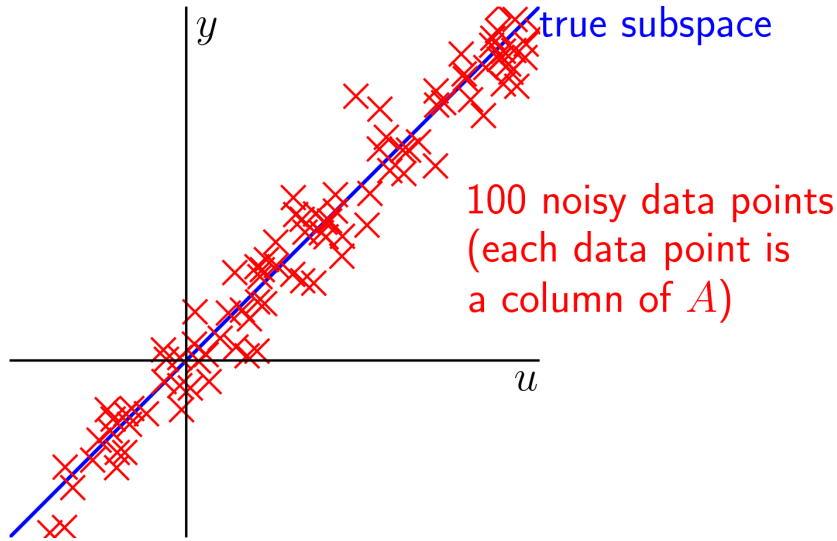
For control, we split the L-length trajectories into

- **past** data: the initial condition at the time of the control decision, and
- **future** data: the planned inputs and outputs at the time of the control decision.

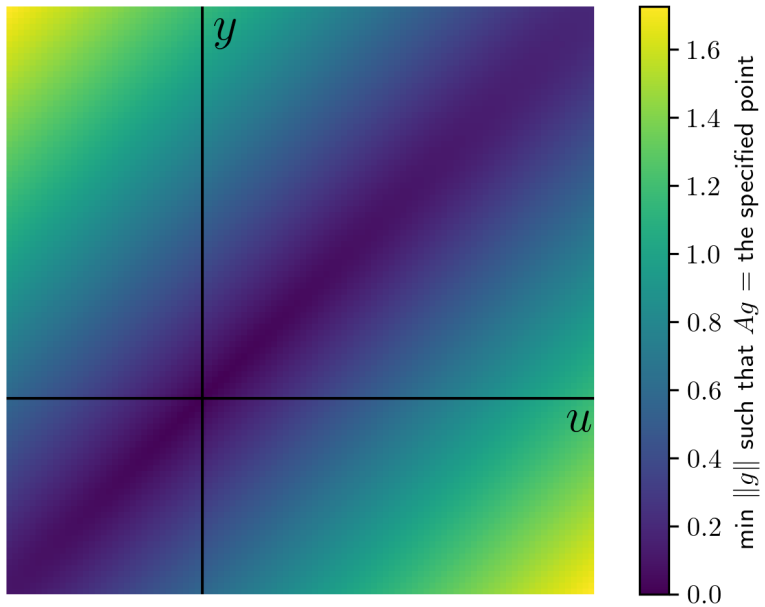
$$\begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \\ y_\rho \\ \vdots \\ y_{L-1} \end{bmatrix} = \underbrace{\begin{bmatrix} I_L & 0 \\ \mathcal{G}_L & \mathcal{O}_L \end{bmatrix}}_{\Lambda_L} \begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ x_0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \\ y_\rho \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g$$

Noisy LTI System Representation

Noisy I/O sequences do not belong to a subspace of $\mathbb{R}^{2L}$



But the "true" system behavior corresponds to using g vectors with small norms:



DeePC with noisy data

(for now, one more regularization to come in this lecture)

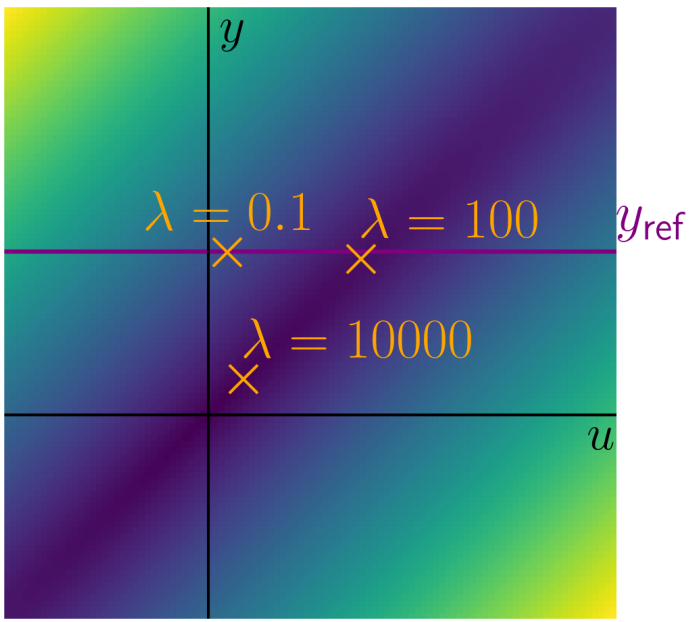
$$\min_{\mathbf{u},\mathbf{y},g,\sigma} \quad y^T Q y + u^T R u + \lambda \|g\|$$

$$\text{subject to} \quad \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix}$$

$$u_k \in \mathcal{U}_k \ \forall k$$

$$y_k \in \mathcal{Y}_k \ \forall k$$

Cartoon demonstrating the "Goldilocks" property of λ:



DeePC Implementation Steps

Offline

1. Gather the offline/"training" data
2. Organize the training data in a Hankel matrix:

$$\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix}$$

Online, at each timestep

1. collect the ρ recent "initial condition" measurements into $\mathbf{u}_{\text{past}}$ and $\mathbf{y}_{\text{past}}$
2. run the DeePC optimization to determine u

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y},g} \quad & y^T Q y + u^T R u + \lambda \|g\| \\ \text{subject to} \quad & \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

DeePC

$$\min_{\mathbf{u},\mathbf{y},g} \quad y^T Q y + u^T R u + \lambda \|g\| + \lambda_\sigma \|\sigma\|$$

$$\text{s. t.} \quad \underbrace{\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix}}_{\text{wide matrix}} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} + \sigma \\ \mathbf{y} \end{bmatrix}$$

$$u \in \mathcal{U}$$

$$y \in \mathcal{Y}$$

Benefits

- No system model needed
- No state estimator (or even state dimension) needed
- Flexibility. e.g., $\|g\|_1$ regularization and the σ slack variable

Downsides

- Computationally expensive (optimizing over a **large vector** g)
- Larger memory footprint (**data** $\mathcal{H}$ instead of **parametric model** A, B, C)

End of recap from last week

General Data-Driven Predictive Control Implementation Steps

Offline

1. Gather the offline/"training" data
2. Organize/process the training data in some way , e.g. putting the data in a Hankel matrix:

$$\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix}$$

Online, at each timestep

1. collect the ρ recent "initial condition" measurements into u_{past} and $\mathbf{y}_{\text{past}}$
2. run an optimization to determine u

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{s.t.} \quad & \text{[insert dynamics constraints here]} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

$\rightarrow$ Are there alternatives to DeePC that do not add g to the optimization?

Subspace Predictive Control (SPC)

uses the Subspace (Multistep, Output) Predictor, estimated from data:

$$\min_{\mathbf{u},\mathbf{y}} \quad y^T Q y + u^T R u$$

$$\text{subject to} \quad \mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$$

$$u \in \mathcal{U}$$

$$y \in \mathcal{Y}$$

The Subspace Multistep Predictor $\widehat{S}$ is the "least-squares $\mathbf{y}$-predictor"

$$\widehat{S} := \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger$$

$$= \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \underbrace{\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^T \left(\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix} \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^T \right)^{-1}}_{\text{the pseudo-inverse of a wide matrix}}$$

Why $\widehat{S} := \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger$?

Answer:

$$\widehat{S} = \underset{X}{\operatorname{argmin}} \underbrace{\left\| \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) - X \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix} \right\|_F}_{\text{Frobenius Norm} \leftrightarrow \text{Least-squares Error}}$$

Thus, $\widehat{S}$ is the best* predictor of $\mathcal{H}_K(\mathbf{y}_{\text{data,fut}})$ for $\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}$.

*in the least-squares sense

Proof

Goal: $\underset{X}{\operatorname{argmin}} f(X)$, where $f(X) = \|B - XA\|_F = (B - XA)(B - XA)^T$

We know that at the optimal X, $\nabla f(X) = 0$ (∇ indicates the gradient).

We also know (from linear algebra textbooks) that $\nabla f(X) = 2(XAA^T - BA^T)$.

Thus, $X^{\text{opt}} = BA^T(AA^T)^{-1}$.

(This is one of the core results of Optimization Theory, see Ch. 12 of this book.)

Why "Subspace" Predictive Control?

For noiseless SISO data, the stacked L-long trajectories

$$\begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \\ y_\rho \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} I_{L+\rho} \\ \widehat{S} \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \end{bmatrix}$$

are all in the $\mathbb{R}^{L+\rho}$ subspace of $\mathbb{R}^{2L}$ defined by the column-span of $\begin{bmatrix} I_{L+\rho} \\ \widehat{S} \end{bmatrix}$.

Given u_{past}, u, and y_{past}, the subspace uniquely defines a y.

Subspace Predictive Control (SPC) Implementation Steps

Offline

1. Gather the training data
2. Organize the training data in Hankel matrices and estimate

$$\widehat{S} := \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger$$

Online, at each timestep

1. collect the ρ recent "initial condition" measurements into $\mathbf{u}_{\text{past}}$ and $\mathbf{y}_{\text{past}}$
2. run the following optimization to determine u

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{s.t.} \quad & \mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

Subspace Predictive Control (SPC)

uses the Subspace Predictor, estimated from data:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{subject to} \quad & \mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

Equivalence:

- In the noiseless case, $\widehat{S}$ gives the same prediction of $\mathbf{y}$ as DeePC

Benefits:

- Resolves DeePC's computational issues by removing g

SPC and DeePC equivalences

Subspace Predictive Control (SPC)

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{subject to} \quad & \mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

is equivalent* to DeePC with (only) the "Projection Regularizer" $\lambda_p = \infty$:

(Kernel) Projection Matrix:

$$\Pi = \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}$$

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y},g} \quad & y^T Q y + u^T R u + \lambda_p \|(I - \Pi) g\|_2^2 \\ \text{s. t.} \quad & \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix} \end{aligned}$$

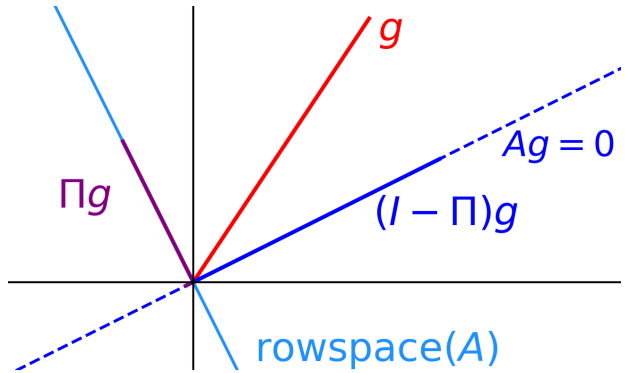
*assuming there is at least one g such that

$$\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$$

What is the (Kernel) Projection Matrix Π ?

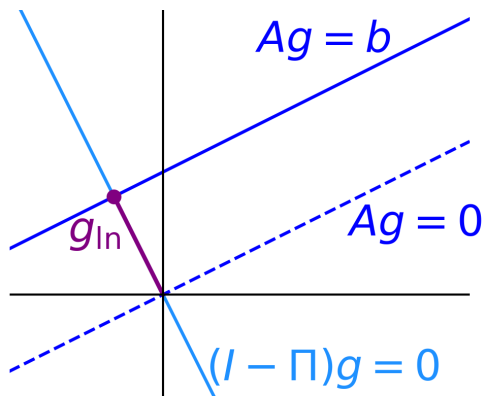
For a wide matrix A, consider the linear equation: $\underbrace{Ag = b}_{\text{there are many } g\text{s for which this holds}}$.

Defining $\Pi = A^T \left(AA^T\right)^{-1} A$, Πg is the projection of g onto the rowspace of A:



Thus, $(I - \Pi)g$ is the portion of g in the nullspace of A.

The "least-norm" solution g_{ln} is the *only* solution to both $Ag = b$ and $(I - \Pi)g = 0$.

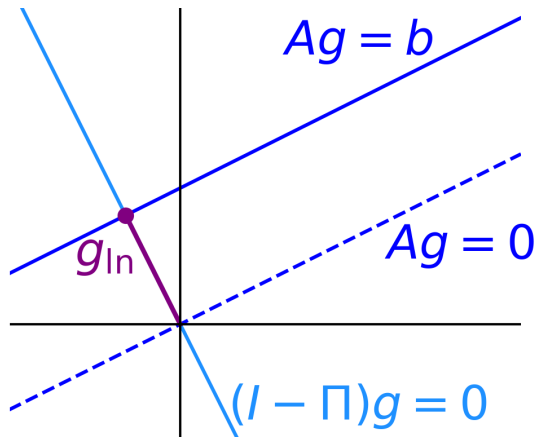


g_{ln} is also given by $g_{\text{ln}} = A^{\dagger} b$.

Why is SPC equivalent to DeePC with (just) $\lambda_p = \infty$?

If $\lambda_p = \infty$, penalizing $\lambda_p \|(I - \Pi)g\|_2^2$ ensures that $(I - \Pi)g = 0$.

$(I - \Pi)g = 0$ with $\underbrace{\begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}}_{\text{there are many } g\text{s for which this holds}}$ gives $\underbrace{g_{\text{ln}} = \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}}_{\text{the least-norm } g\text{, of the many } g\text{s}}$.



Plugging g_{ln} into the $\mathbf{y}$ prediction gives $\mathbf{y} = \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$,

which is exactly the Subspace Prediction given by $\widehat{S}$.

What does SPC lose, relative to DeePC?

DeePC, with the Projection Regularizer, has more regularization flexibility:

$$\min_{\mathbf{u},\mathbf{y},g} \quad y^T Q y + u^T R u + \lambda_g \|g\| + \lambda_p \|(I - \Pi)g\|_2^2$$

$$\text{subject to} \quad \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix}$$

$$u \in \mathcal{U}$$

$$y \in \mathcal{Y}$$

λ_g and λ_p are hyperparameters which can/must be tuned for the control task.

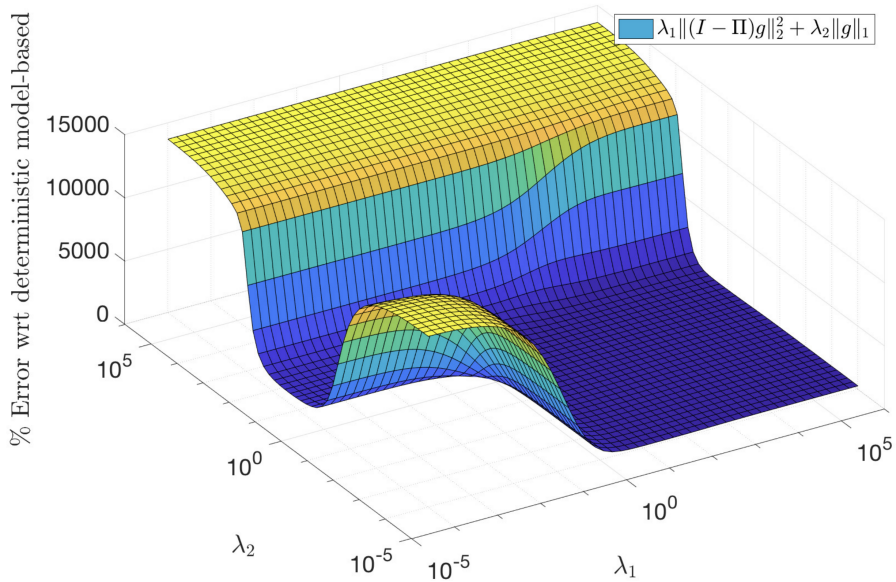


Figure: Dörfler et al.'s *Bridging Direct and Indirect Data-Driven Control Formulations via Regularizations and Relaxations."*

Benefits of DeePC

DeePC provides more regularization flexibility (e.g., 1-norm g regularization), and the option to put a slack variable on the initial condition:

$$\min_{\mathbf{u},\mathbf{y},g,\sigma} \quad y^T Q y + u^T R u + \lambda_{g_1} \|g\|_1 + \lambda_p \|(I - \Pi)g\|_2^2 + \lambda_\sigma \|\sigma\|$$

For nonlinear systems, or data sets with erroneous outlier data, this additional flexibility may be desirable.

Subspace Predictive Control (SPC)

uses the Subspace Predictor, estimated from data:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{subject to} \quad & \mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

Equivalence:

- In the noiseless case, $\widehat{S}$ gives the same prediction of $\mathbf{y}$ as DeePC

Benefits:

- Resolves DeePC's computational issues by removing g

Shortcomings:

- Less flexibility than DeePC: no $\|g\|_1$ regularization or σ slack variable

Are there downsides to subspace-based methods such as DeePC and SPC?

SPC and DeePC are Subspace Methods

For noiseless SISO data (and a valid y_{past}), the stacked L-long trajectories

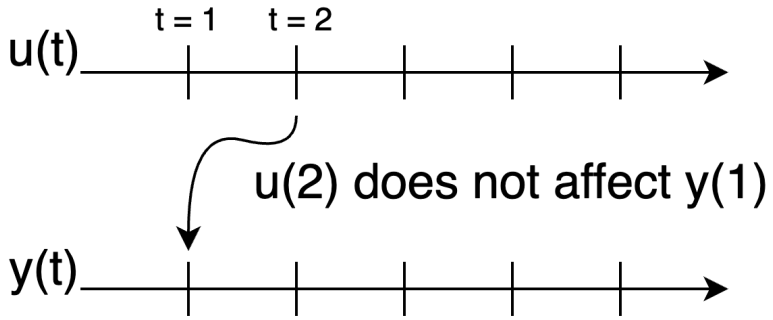
$$\begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \\ y_\rho \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}}) \end{bmatrix} g \qquad \text{and} \qquad \begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \\ y_\rho \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} I_{L+\rho} \\ \widehat{S} \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{\rho-1} \\ u_\rho \\ \vdots \\ u_{L-1} \\ y_0 \\ \vdots \\ y_{\rho-1} \end{bmatrix}$$

occupy the same $\mathbb{R}^{L+n}$ subspace of $\mathbb{R}^{2L}$.

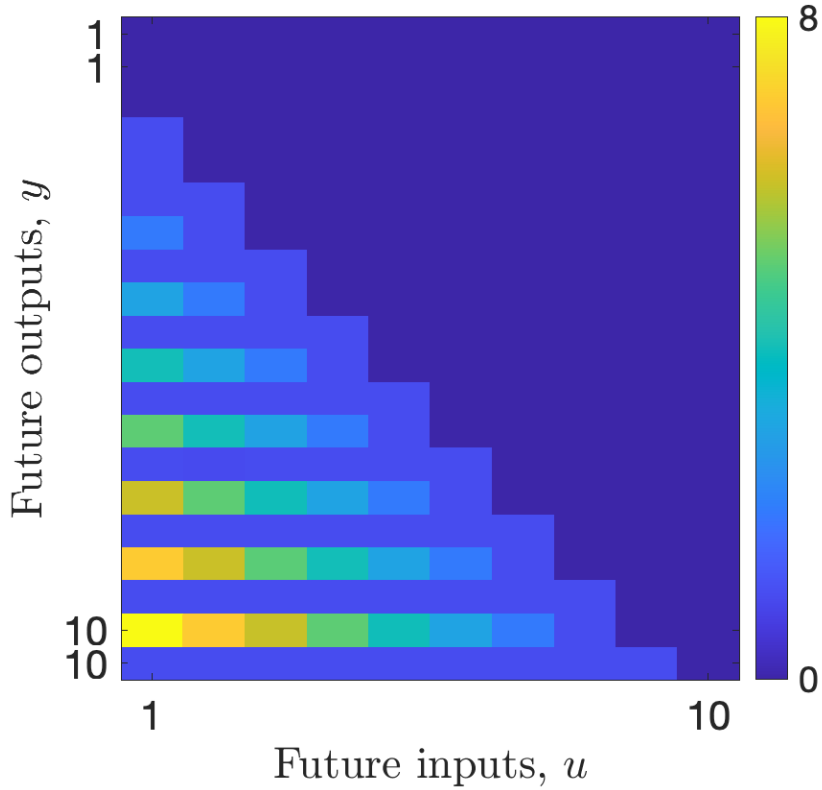
$\rightarrow$ if u_{past}, u, and y_{past} match, then y also matches.

Technicality: this is true even though $\begin{bmatrix} I_{L+\rho} \\ \widehat{S} \end{bmatrix} \in \mathbb{R}^{L+\rho}$, not $\mathbb{R}^{L+n}$, because valid $\begin{matrix} u_{\text{past}} \\ y_{\text{past}} \end{matrix} \in \mathbb{R}^{\rho+n}$, not $\mathbb{R}^{2\rho}$.

Causality



Thus, the true relation between u and y is lower-triangular:

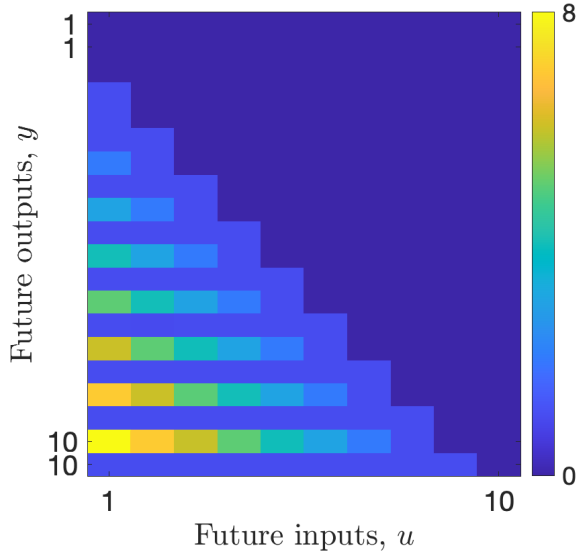


Subspace Methods and Causality

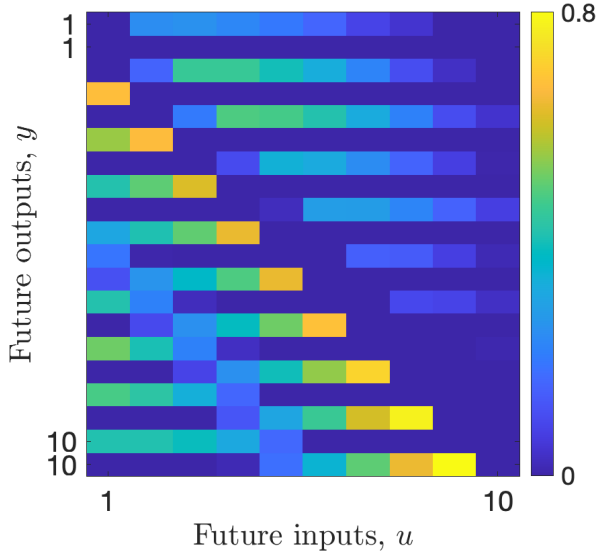
Subspace methods do not assert causality.

E.g., for 100 noisy open-loop data points from a double integrator:

True Predictor: $\mathbf{y} = H \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$

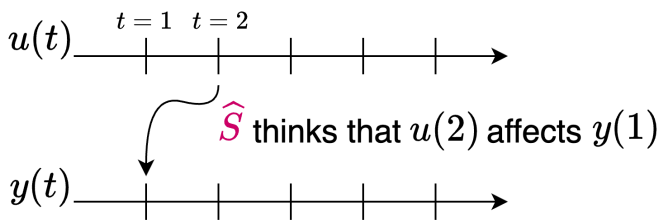


Subspace Predictor: $\mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$



The lack of causality degrades control performance.

Can it cause other issues?

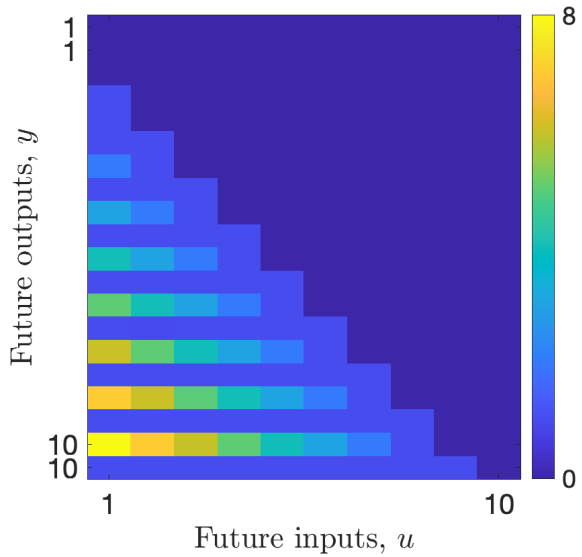


Subspace Methods and Causality

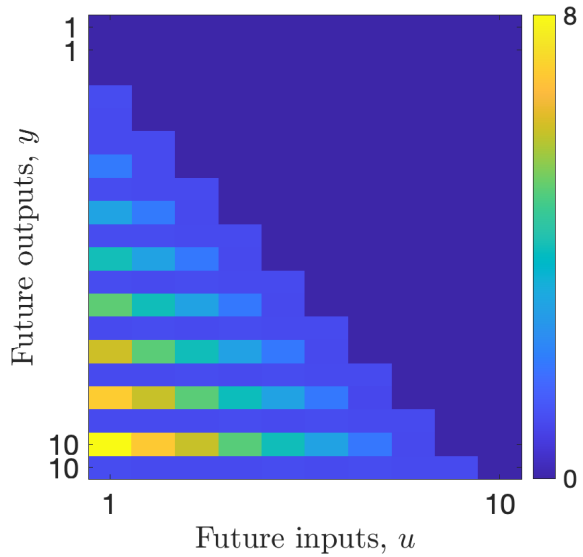
Subspace methods do not assert causality. But with enough data, causality emerges.

E.g., for 1000 noisy **open-loop** data points from a double integrator:

True Predictor: $\mathbf{y} = H \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$

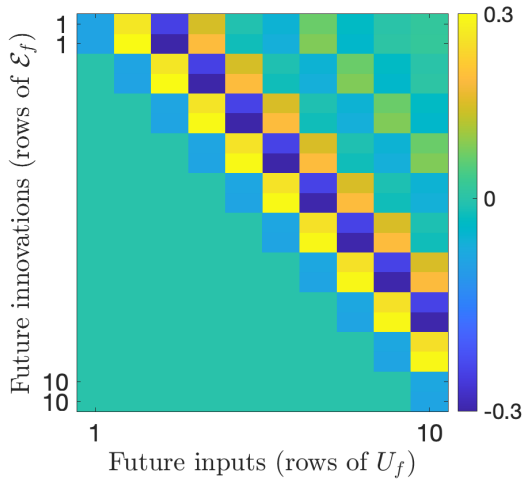


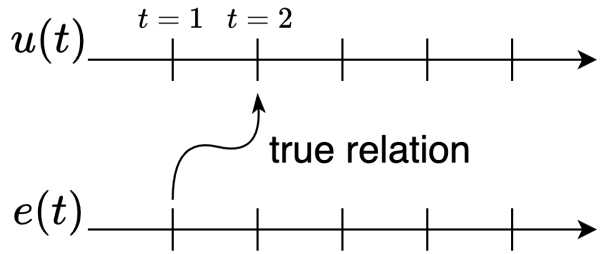
Subspace Predictor: $\mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$



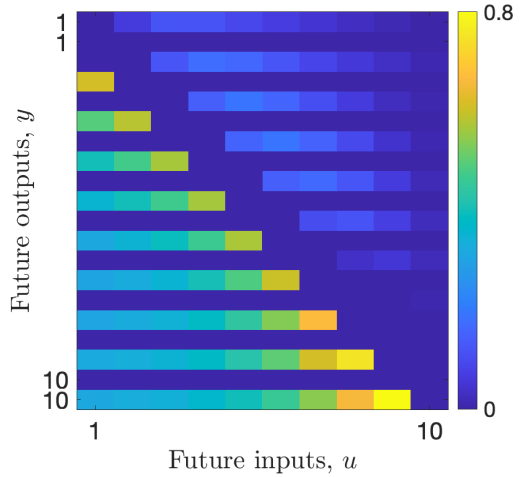
The Bias of Subspace Methods with closed-loop data

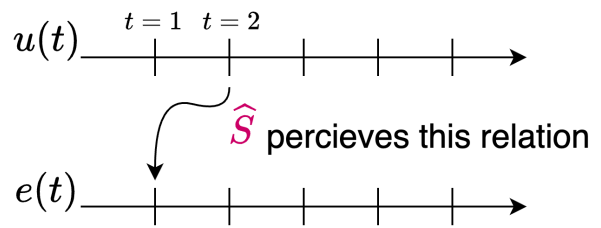
When the training data is gathered in closed-loop with a feedback controller, the innovation/noise is correlated with the input:





The Subspace Predictor mistakes this causation of this correlation, producing an acausal predictor, regardless of how much data is gathered:





The plot above is for 1000 noisy **closed-loop** data points from a double integrator.

Shortcomings of Subspace Methods:

Even with open-loop data, the lack of causality degrades prediction performance:

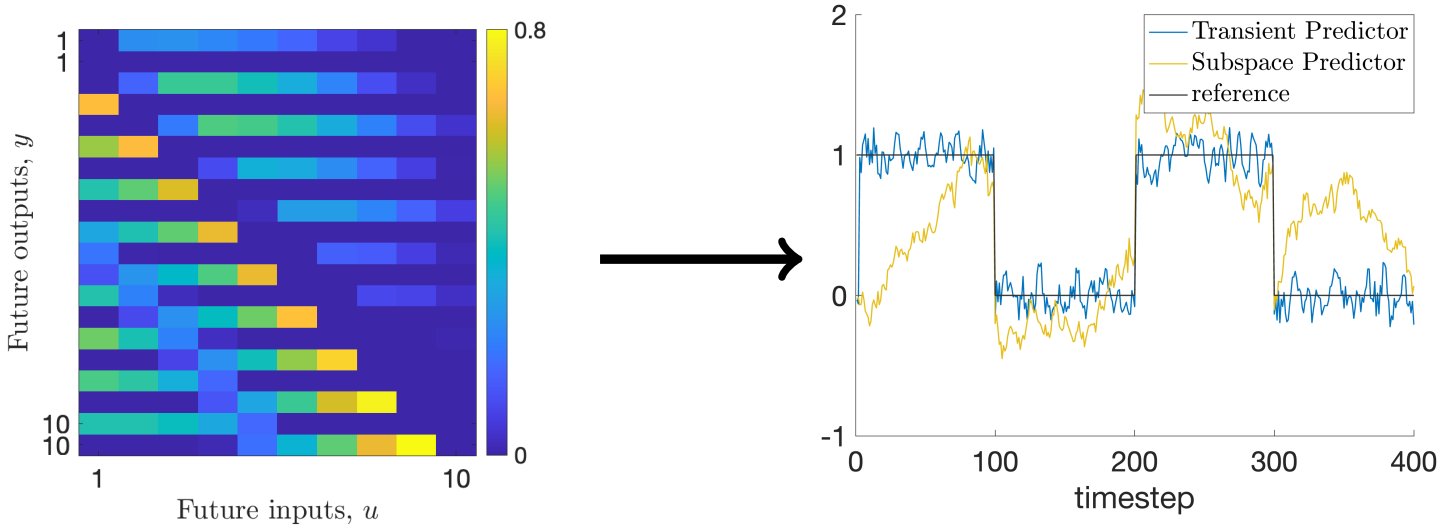


Figure: Moffat, Dorfler, Chiuso's *The Transient Predictor*

In addition, if the training data is gathered in closed-loop with a feedback controller, Subspace Methods give (aymptotically) biased output predictions:

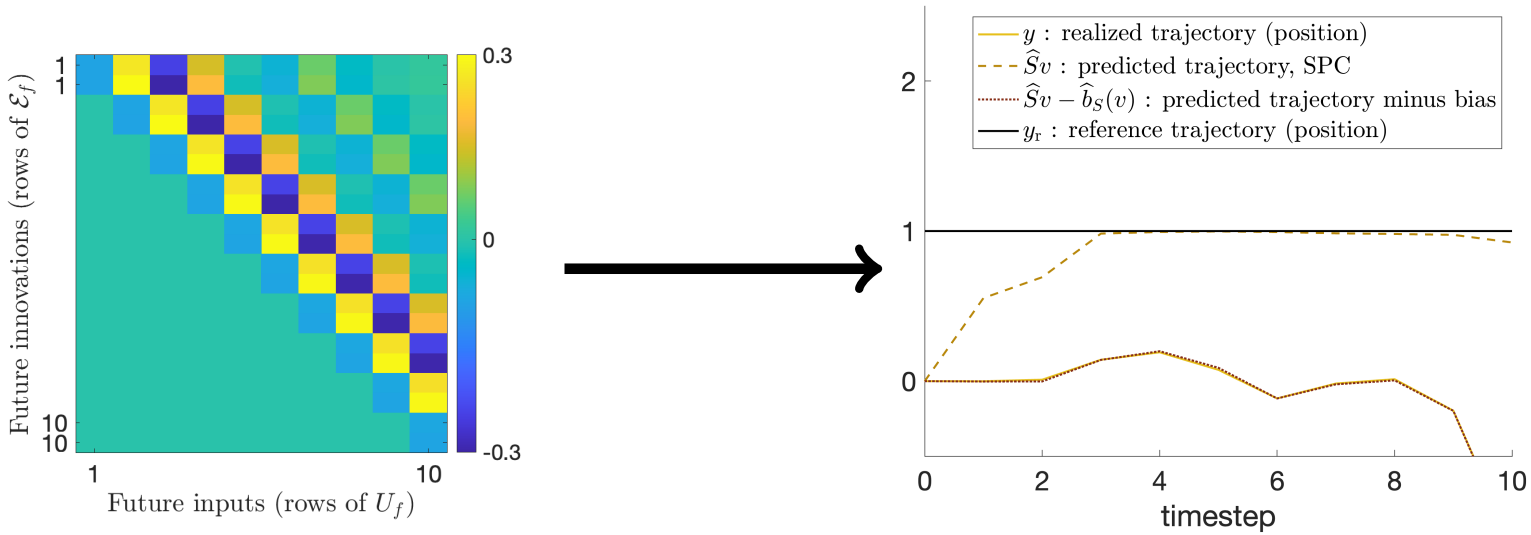


Figure: Moffat, Dorfler, Chiuso's *The BiaS of Subspace-based Data-Driven Predictive Control*

(Noiseless) LTI System Representations

LTI systems can be represented in a number of ways

State Space

$$x_{t+1} = Ax_t + Bu_t$$
$$y_t = Cx_t$$

Subspace

$$\text{span}\{\Lambda_L\}$$

or

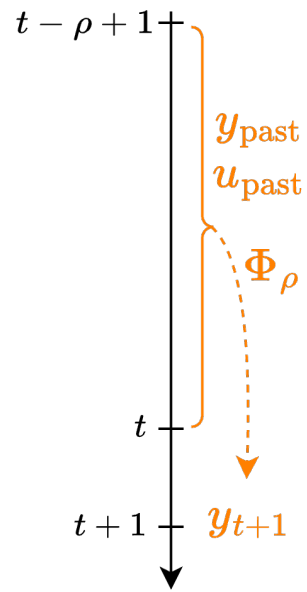
$$\text{span}\left\{\begin{bmatrix}\mathcal{H}_\rho(\mathbf{u}_{\text{data,past}})\\ \mathcal{H}_K(\mathbf{u}_{\text{data,fut}})\\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}})\\ \mathcal{H}_K(\mathbf{y}_{\text{data,fut}})\end{bmatrix}\right\}$$

or

$$\text{span}\left\{\begin{bmatrix}I_{L+\rho}\\ \widehat{S}\end{bmatrix}\right\}$$

ARX

$$y_k = \Phi_\rho \begin{bmatrix}y_{\text{past}}\\ u_{\text{past}}\end{bmatrix}$$



Any of these representations can be used for predictive control.

(Noiseless) LTI System Representations

LTI systems can be represented in a number of ways

State Space

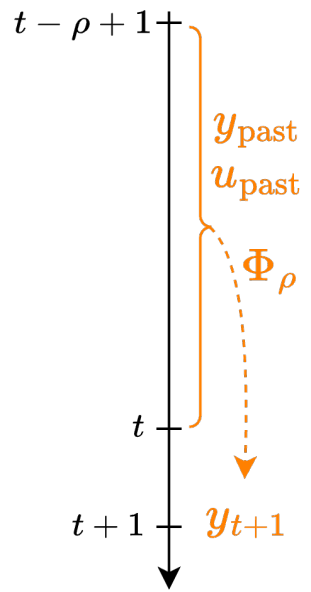
$$x_{t+1} = Ax_t + Bu_t$$
$$y_t = Cx_t$$

Subspace

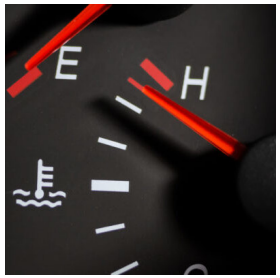
$$\text{span}\left\{ \begin{bmatrix} \mathcal{H}(u) \\ \mathcal{H}(y) \end{bmatrix} \right\}$$

ARX

$$y_k = \Phi_\rho \begin{bmatrix} y_{\text{past}} \\ u_{\text{past}} \end{bmatrix}$$



Do all systems have states?







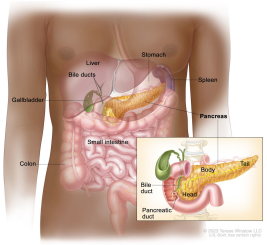


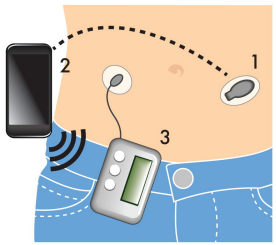




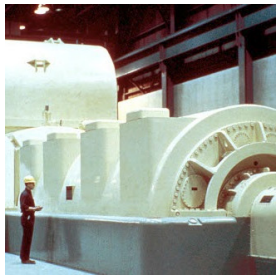








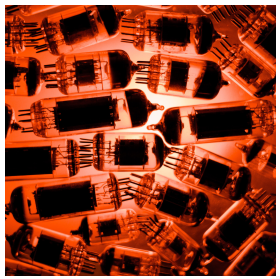












Subspace and ARX methods do not require a state definition, or even a dimension.

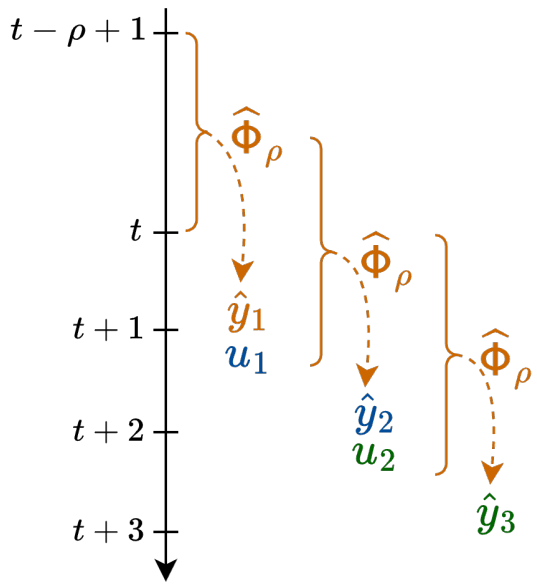
ARX Representations

$$y_{t+1} = \Phi_\rho \begin{bmatrix} y_{\text{past}} \\ u_{\text{past}} \end{bmatrix} = \overbrace{\phi_{\rho,y} y_{\text{past}}}^{\text{AutoRegressive}} + \overbrace{\phi_{\rho,u} u_{\text{past}}}^{\text{eXogenous inputs}}$$

ARX system identification is easy:

$$\widehat{\Phi}_\rho = \mathcal{H}_1(\mathbf{y}_{\text{data,past+1}}) \begin{bmatrix} \mathcal{H}_\rho(\mathbf{u}_{\text{data,past}}) \\ \mathcal{H}_\rho(\mathbf{y}_{\text{data,past}}) \end{bmatrix}^\dagger$$

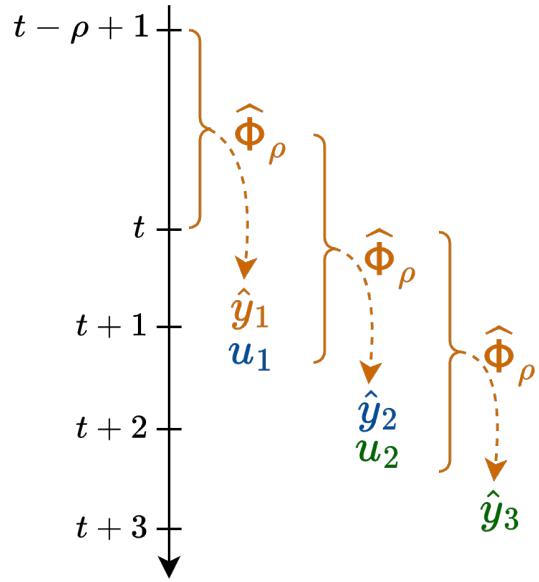
ARX models can be “stacked” to predict multiple timesteps into the future:



ARX Predictive Control

uses a single ARX model to iteratively predict future outputs

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u \\ \text{s.t.} \quad & \text{for } k = 0 \text{ to } K-1: \\ & y_{1+k} = \widehat{\Phi}_\rho \underbrace{\begin{bmatrix} y_{\text{past}+k} \\ u_{\text{past}+k} \end{bmatrix}}_{\text{length does not increase with } k} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$



Equivalence:

- In the noiseless case, iterating $\widehat{\Phi}_\rho$ K times gives the same prediction of $\mathbf{y}$ as DeePC and $\widehat{S}$

Benefits:

- Fast optimization (no g)
- Causal
- Consistent predictions, even with closed-loop data

Shortcomings:

- DeePC provides more regularization flexibility, such as $\|g\|_1$ regularization
- Can perform poorly when ρ is small

Transient Predictive Control (TPC)

uses K-different increasing-length ARX models to iteratively predict future outputs

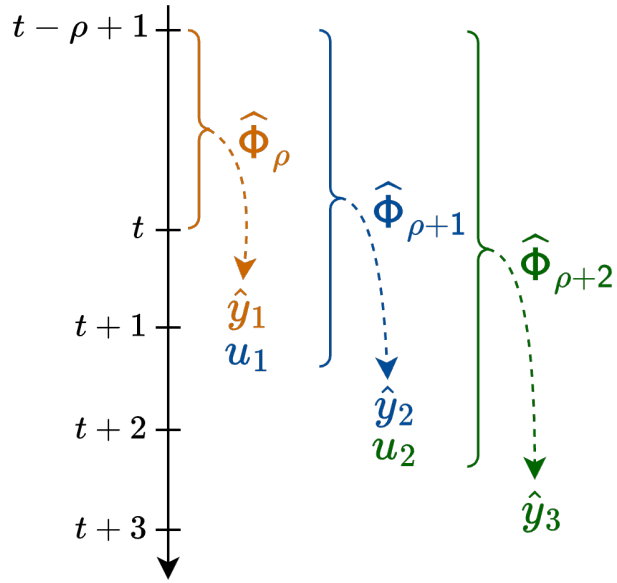
$$\min_{\mathbf{u},\mathbf{y}} \quad y^T Q y + u^T R u$$

$$\text{s.t.} \quad \text{for } k = 0 \text{ to } K-1:$$

$$y_{1+k} = \widehat{\Phi}_{\rho+k} \underbrace{\begin{bmatrix} y_{\text{past:}k} \\ u_{\text{past:}k} \end{bmatrix}}_{\text{length increases with } k}$$

$$u \in \mathcal{U}$$

$$y \in \mathcal{Y}$$



Equivalence:

- In the noiseless case, the Transient Prediction gives the same prediction of **y** as DeePC and $\widehat{S}$

Benefits:

- Fast optimization (no g)
- Causal
- Consistent predictions, even with closed-loop data
- Can outperform ARX Predictive Control when ρ is small

Shortcomings:

- DeePC provides more regularization flexibility, such as $\|g\|_1$ regularization

What about regularization?

In DeePC, regularization on g hedges against uncertainty in the prediction of y:

$$\min_{\mathbf{u},\mathbf{y},g} y^T Q y + u^T R u + \lambda_g \|g\|$$

Transient Predictive Control and ARX Predictive Control can regularize u, rather than g, to achieve the same robust performance as a 2-norm g regularization:

$$\min_{\mathbf{u},\mathbf{y}} \quad y^T Q y + u^T R u + u^T \Lambda u$$

But, like SPC and ARX Predictive Control, TPC do not have an equivalent to DeePC's 1-norm regularization on g.

Transient Predictive Control (TPC) Implementation Steps

Offline

1. Gather the training data
2. Organize the training data in Hankel matrices and estimate

$$\widehat{\Phi}_{\rho+k} = \mathcal{H}_1(\mathbf{y}_{\text{data},\rho+k+1}) \begin{bmatrix} \mathcal{H}_{\rho+k}(\mathbf{u}_{\text{data},1:\rho+k}) \\ \mathcal{H}_{\rho+k}(\mathbf{y}_{\text{data},1:\rho+k}) \end{bmatrix}^{\dagger}$$

for all k from 0 to $K-1$.
(The subscript of $\mathcal{H}$ indicates how many (block) rows the Hankel matrix has.)

Online, at each timestep

1. collect the ρ recent "initial condition" measurements into $\mathbf{u}_{\text{past}}$ and $\mathbf{y}_{\text{past}}$
2. run the following optimization to determine u

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & y^T Q y + u^T R u + u^T \Lambda u \\ \text{s.t.} \quad & \mathbf{y}_{1+k} = \widehat{\Phi}_{\rho+k} \begin{bmatrix} \mathbf{u}_{\text{past}:k} \\ \mathbf{y}_{\text{past}:k} \end{bmatrix} \forall k \in \{0, \ldots, K-1\} \\ & u \in \mathcal{U} \\ & y \in \mathcal{Y} \end{aligned}$$

Benefits of Transient Predictive Control/ARX-based methods:

Impact of causality:

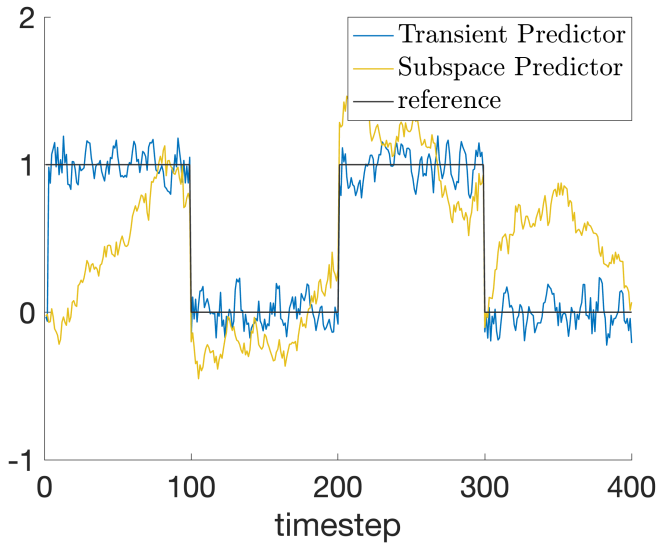
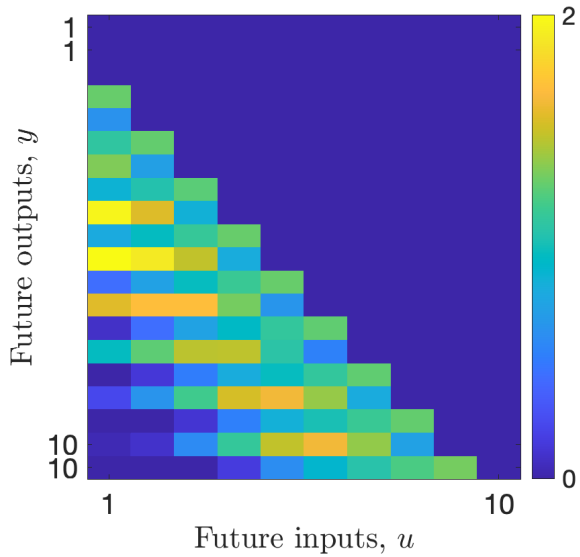
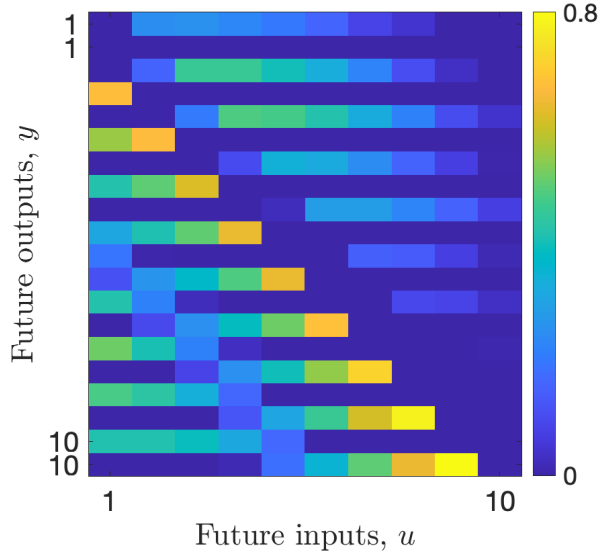


Figure: Moffat, Dorfler, Chiuso's *The Transient Predictor* (2024)

Transient Predictor: $y_{1+k} = \widehat{\Phi}_{\rho+k} \begin{bmatrix} y_{\text{past}:k} \\ u_{\text{past}:k} \end{bmatrix}$

Subspace Predictor: $\mathbf{y} = \widehat{S} \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \end{bmatrix}$

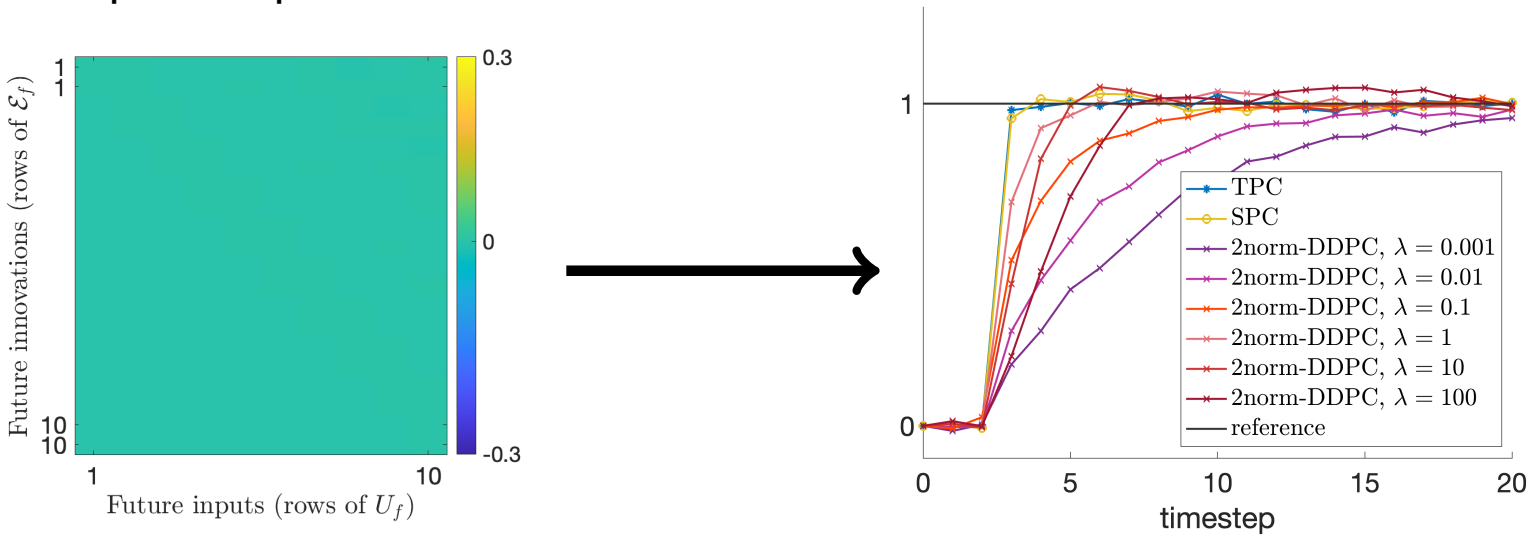




Benefits of Transient Predictive Control/ARX-based methods:

Consistent predictions, even with closed-loop training data:

With open-loop data:



With closed-loop data:

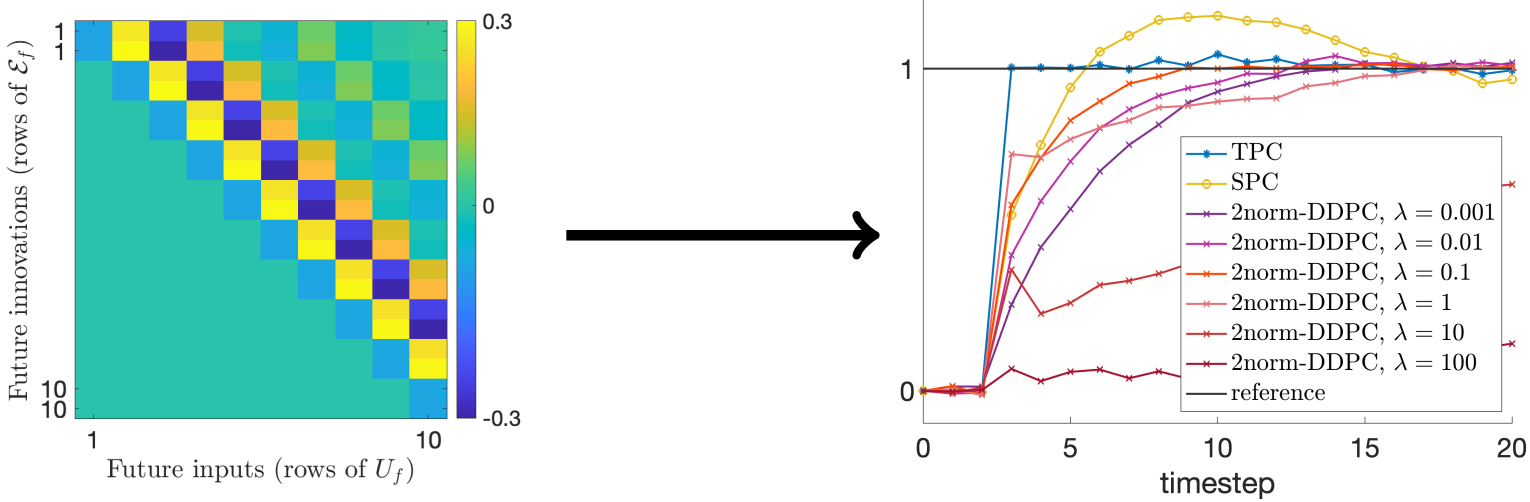
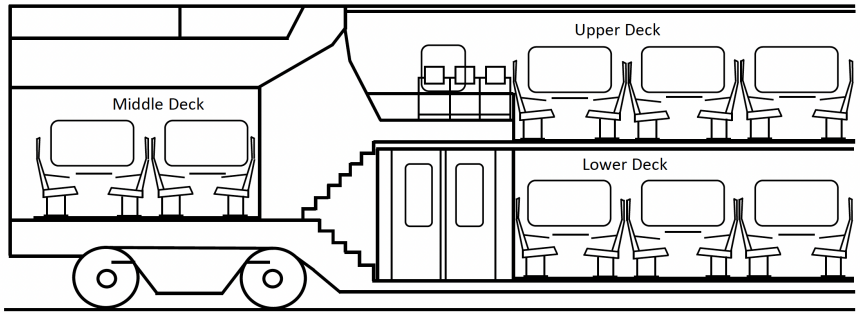
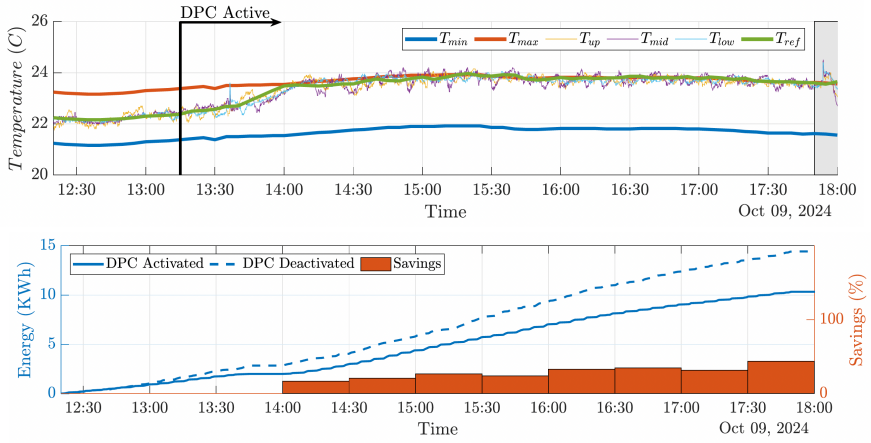


Figure: Moffat, Dorfler, Chiuso's *The BiaS of Subspace-based Data-Driven Predictive Control*

TPC has recently been demonstrated on some large systems:



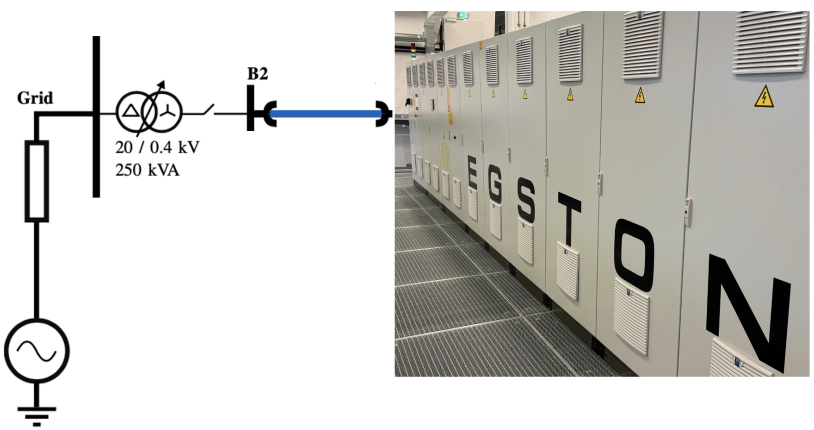
SBB Train Thermal Management

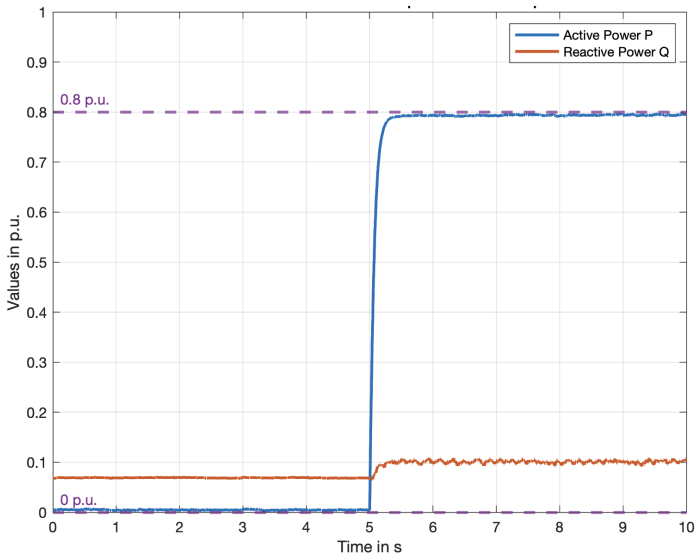


Does it work?

Transient Predictive Control on the European grid

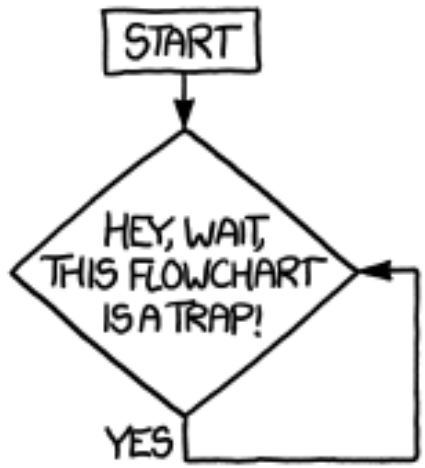
Technical University of Munich





100 kVA Grid-Connected Converter Control

The control engineer flowchart





This work is licensed under a
Creative Commons Attribution-ShareAlike 4.0 International License

https://bsaver.io/COCO