# Control without models

## Data-driven control and its applications in robotics



Jeremy Coulson

University of Wisconsin–Madison

MaGIC 2026

Machine–Ground Interaction Consortium

# Collaborators



Florian Dörfler
ETH Zürich



John Lygeros
ETH Zürich



Ivan Markovsky
ICREA & CIMNE



Henk J. van Waarde
University of Groningen



Alberto Padoan
University of British Columbia



Joel Stevens, Dian Jin
UW–Madison

# The systems in this room



**What we would like**

A model good enough to design and certify a controller

# The systems in this room



**What we would like**

A model good enough to design and certify a controller

**What we have**

- systems that are changing
- systems that are expensive to simulate and hard to identify
- plenty of measured trajectories

# The systems in this room



**What we would like**

A model good enough to design and certify a controller

**What we have**

- systems that are changing
- systems that are expensive to simulate and hard to identify
- plenty of measured trajectories

**Modelling is the bottleneck, but the data is not**

# Today




unknown system, measured trajectories

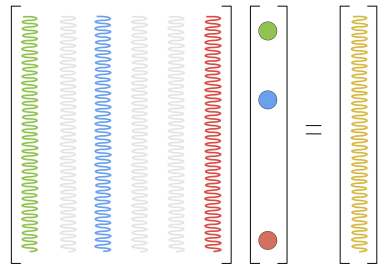
# Today




Can we design a controller directly from data, with guarantees, and never build a model?

# Menu

- Foundations
- Data-enabled predictive control (DeePC)
- Applications in robotics



# Sneak preview



12 ton machine: nonlinear, noisy, actuation delays, dead zones

Goal: accurate trajectory tracking for the arm, without a model

# Sneak preview



12 ton machine: nonlinear, noisy, actuation delays, dead zones

Goal: accurate trajectory tracking for the arm, without a model

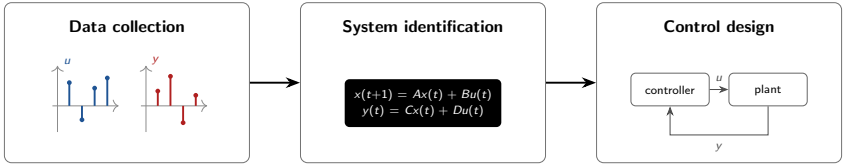
**RL approach** [Egli, Hutter '20]

- $\sim$2 hours of data
- 10 hours to train a model

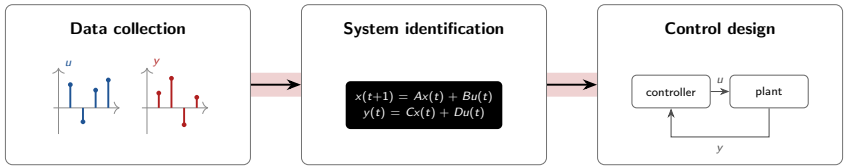
**DeePC** (today's approach)

- $\sim$20 seconds of data
- no model training

# What is data-driven control?



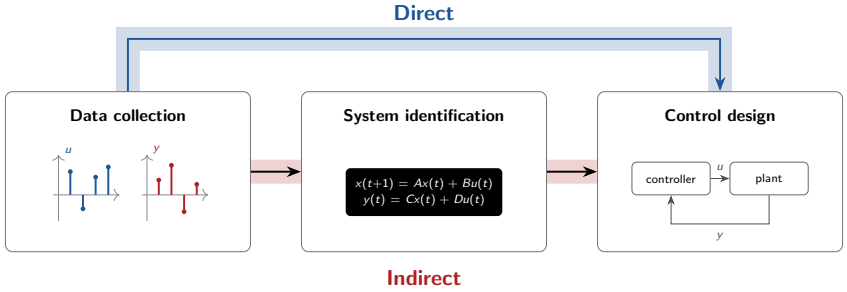
# What is data-driven control?




**Indirect: identify, then design**

- \+ mature, and the model is reusable
- – the identification objective is not the control objective
- – you must pick a model class and order (bias)

# What is data-driven control?




**Indirect: identify, then design**

- \+ mature, and the model is reusable
- − the identification objective is not the control objective
- − you must pick a model class and order (bias)

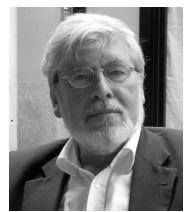
**Direct: data straight to controller**

- \+ optimizes the objective you care about
- \+ easy to implement (no experts needed)
- ± No bias...no prior knowledge
- − sometimes less tractable

# Behavioral system theory [Willems '86]

**Dynamical system**

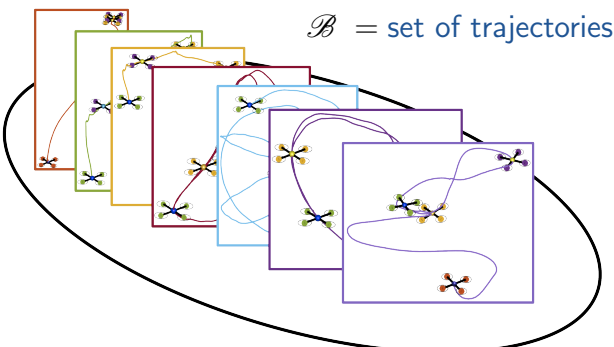
A discrete-time dynamical system is a triple $\Sigma = (\mathbb{Z}_+, \mathbb{R}^{m+p}, \mathscr{B})$, where the behavior $\mathscr{B} \subseteq \{ w = (u, y) : \mathbb{Z}_+ \to \mathbb{R}^{m+p} \}$ is the set of trajectories the system can produce.



# Behavioral system theory [Willems '86]

**Dynamical system**

A discrete-time dynamical system is a triple $\Sigma = (\mathbb{Z}_+, \mathbb{R}^{m+p}, \mathscr{B})$, where the behavior $\mathscr{B} \subseteq \{ w = (u, y) : \mathbb{Z}_+ \to \mathbb{R}^{m+p} \}$ is the set of trajectories the system can produce.

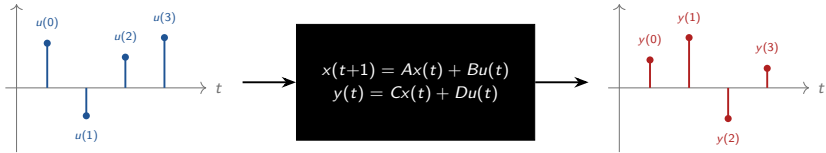





**The behavior contains every trajectory the machine can fly**

# Data matrix as a model

[Willems et al. '05] [Markovsky, Dörfler '21]

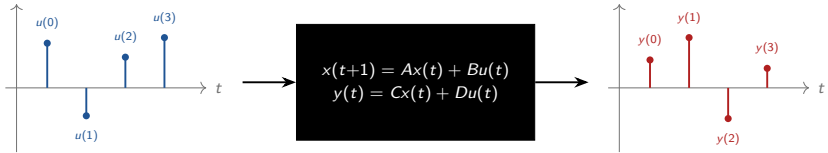



Given $(u_{[0,T-1]}, y_{[0,T-1]})$ and complexity parameters order $n$ and lag $\ell$:

$$\text{im} \underbrace{\begin{bmatrix} u(0) & u(1) & \cdots & u(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ u(L-1) & u(L) & \cdots & u(T-1) \\ y(0) & y(1) & \cdots & y(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ y(L-1) & y(L) & \cdots & y(T-1) \end{bmatrix}}_{=:\, H} \subseteq \mathscr{B}|_{[0,L-1]}$$

# Data matrix as a model

[Willems et al. '05] [Markovsky, Dörfler '21]

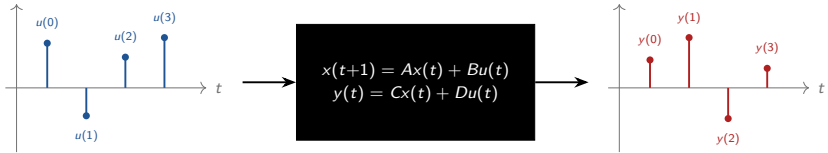



Given $(u_{[0,T-1]}, y_{[0,T-1]})$ and complexity parameters order $n$ and lag $\ell$:

$$\text{im} \underbrace{\begin{bmatrix} u(0) & u(1) & \cdots & u(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ u(L-1) & u(L) & \cdots & u(T-1) \\ y(0) & y(1) & \cdots & y(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ y(L-1) & y(L) & \cdots & y(T-1) \end{bmatrix}}_{=:\, H} = \mathscr{B}|_{[0,L-1]} \quad \Longleftrightarrow \quad \begin{array}{c} \text{rank}\, H = mL + n \\ L \geq \ell \end{array}$$

# Data matrix as a model

[Willems et al. '05] [Markovsky, Dörfler '21]

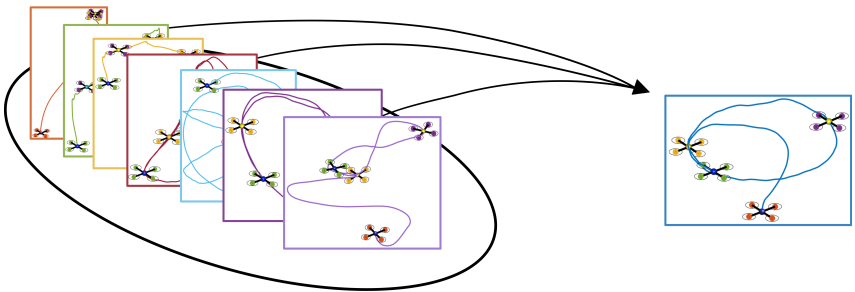



Given $(u_{[0,T-1]}, y_{[0,T-1]})$ and complexity parameters order $n$ and lag $\ell$:

$$\text{im} \underbrace{\begin{bmatrix} u(0) & u(1) & \cdots & u(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ u(L-1) & u(L) & \cdots & u(T-1) \\ y(0) & y(1) & \cdots & y(T-L) \\ \vdots & \vdots & \ddots & \vdots \\ y(L-1) & y(L) & \cdots & y(T-1) \end{bmatrix}}_{=:\, H} = \mathscr{B}|_{[0,L-1]} \quad \Longleftrightarrow \quad \begin{array}{c} \text{rank}\, H = mL + n \\ L \geq \ell \end{array}$$

**Data matrix is a model!**

# Data-enabled Predictive Control (DeePC)



# **D**ata-**e**nabl**e**d **P**redictive **C**ontrol (DeePC) [Coulson et al. '19]

Given data, reference trajectory $y^{\mathrm{ref}}$, and constraints $\mathcal{U}, \mathcal{Y}$:

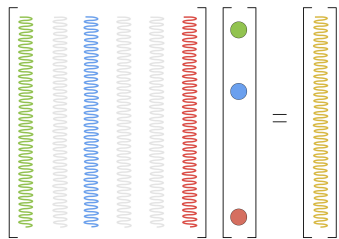
$$\min_{g,u,y} \sum_{k=0}^{T_{\mathrm{f}}-1} \|y_k - y^{\mathrm{ref}}_{t+k}\|^2_Q + \|u_k\|^2_R \quad \Big\} \text{ performance}$$

$$\text{s.t. } \mathcal{H}\, g = \begin{bmatrix} u_{\mathrm{ini}} \\ y_{\mathrm{ini}} \\ u \\ y \end{bmatrix} \quad \Big\} \text{ model}$$

$$u_k \in \mathcal{U},\ y_k \in \mathcal{Y} \quad \Big\} \text{ safety}$$

$\mathcal{H}$ is the data matrix built from offline data

$(u_{\mathrm{ini}}, y_{\mathrm{ini}})$ is the most recent $T_{\mathrm{ini}}$-length trajectory, updated online



# **D**ata-**e**nabl**e**d **P**redictive **C**ontrol (DeePC) [Coulson et al. '19]

Given data, reference trajectory $y^{\mathrm{ref}}$, and constraints $\mathcal{U}, \mathcal{Y}$:

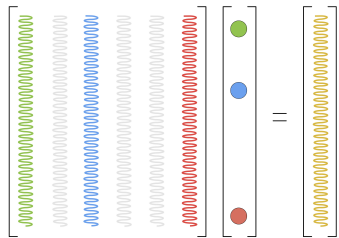
$$\min_{g,u,y} \sum_{k=0}^{T_{\mathrm{f}}-1} \|y_k - y^{\mathrm{ref}}_{t+k}\|^2_Q + \|u_k\|^2_R \quad \Big\} \text{ performance}$$

$$\text{s.t. } \mathcal{H}\, g = \begin{bmatrix} u_{\mathrm{ini}} \\ y_{\mathrm{ini}} \\ u \\ y \end{bmatrix} \quad \Big\} \text{ model}$$

$$u_k \in \mathcal{U},\ y_k \in \mathcal{Y} \quad \Big\} \text{ safety}$$

$\mathcal{H}$ is the data matrix built from offline data

$(u_{\mathrm{ini}}, y_{\mathrm{ini}})$ is the most recent $T_{\mathrm{ini}}$-length trajectory, updated online



**Equivalent to Model Predictive Control (MPC), but the model is replaced by data.**

What about noisy data?

. . . and systems that are not linear?

What about noisy data?

. . . and systems that are not linear?

We need a robustified approach

# Regularization makes it work [Coulson et al. '22]

$$\min_{g,u,y} \quad \sum_{k=0}^{T_f-1} \|y_k - y_{t+k}^{\mathrm{ref}}\|_Q^2 + \|u_k\|_R^2$$

$$\text{s.t.} \quad \mathcal{H}\, g = \begin{bmatrix} u_{\mathsf{ini}} \\ y_{\mathsf{ini}} \\ u \\ y \end{bmatrix} \qquad , \qquad u_k \in \mathcal{U}, \quad y_k \in \mathcal{Y}$$

# Regularization makes it work [Coulson et al. '22]

$$\min_{g,u,y,\sigma_y} \quad \sum_{k=0}^{T_\mathrm{f}-1} \|y_k - y^{\mathrm{ref}}_{t+k}\|_Q^2 + \|u_k\|_R^2 \; + \; \lambda_y \|\sigma_y\|_2^2$$

$$\text{s.t.} \;\; \mathcal{H}\, g = \begin{bmatrix} u_{\mathrm{ini}} \\ y_{\mathrm{ini}} \\ u \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{bmatrix}, \qquad u_k \in \mathcal{U}, \;\; y_k \in \mathcal{Y}$$

**measurement noise**

$(u_{\mathrm{ini}}, y_{\mathrm{ini}})$ is inconsistent with the data, so *no* $g$ *exists*. Add the slack (estimation error) $\sigma_y$.

# Regularization makes it work [Coulson et al. '22]

$$\min_{g,u,y,\sigma_y} \quad \sum_{k=0}^{T_f-1} \|y_k - y_{t+k}^{\text{ref}}\|_Q^2 + \|u_k\|_R^2 \; + \; \lambda_y \|\sigma_y\|_2^2 \; + \; \lambda_g r(g)$$

$$\text{s.t.} \;\; \mathcal{H}\, g = \begin{bmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{bmatrix}, \qquad u_k \in \mathcal{U}, \;\; y_k \in \mathcal{Y}$$

**measurement noise**

$(u_{\text{ini}}, y_{\text{ini}})$ is inconsistent with the data, so *no* $g$ *exists*. Add the slack (estimation error) $\sigma_y$.

**noisy or nonlinear data**

the data matrix is *full rank*, so every $(u, y)$ looks feasible. Add regularizer $\lambda_g r(g)$ (needs tuning)

$r(g) = \|g\|_1 \Rightarrow$ sparse $g \Rightarrow$ motion primitives (low rank)

# Regularization makes it work [Coulson et al. '22]

$$\min_{g,u,y,\sigma_y} \quad \sum_{k=0}^{T_f-1} \|y_k - y^{\text{ref}}_{t+k}\|_Q^2 + \|u_k\|_R^2 \; + \; \lambda_y \|\sigma_y\|_2^2 \; + \; \lambda_g r(g)$$

$$\text{s.t.} \;\; \mathcal{H}\, g = \begin{bmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{bmatrix}, \qquad u_k \in \mathcal{U}, \;\; y_k \in \mathcal{Y}$$

**measurement noise**

$(u_{\text{ini}}, y_{\text{ini}})$ is inconsistent with the data, so *no* $g$ *exists*. Add the slack (estimation error) $\sigma_y$.

**noisy or nonlinear data**

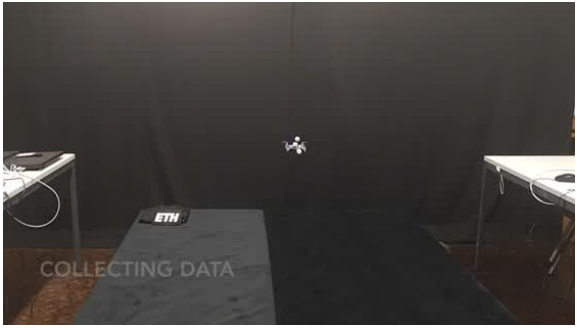
the data matrix is *full rank*, so every $(u, y)$ looks feasible. Add regularizer $\lambda_g r(g)$ (needs tuning)

$r(g) = \|g\|_1 \Rightarrow$ sparse $g \Rightarrow$ motion primitives (low rank)

**regularization $\Leftrightarrow$ robustness**

Does it work on real machines?

# Flying and digging with data

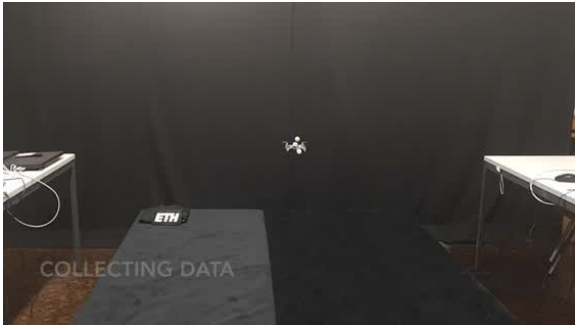



Quadcopter [Elokda et al. '21]



Autonomous excavator [Wegner et al. '21]

# Flying and digging with data




Quadcopter [Elokda et al. '21]



Autonomous excavator [Wegner et al. '21]

**Requires only seconds of data**

# DeePC in the wild



Quadcopter



Excavator



Quadruped [Fawcett]




Exoskeleton [Li]



Soft robot [Wang]



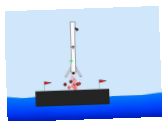
Robot arm [Vahidi-M.]



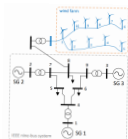
Traffic [Wang]



Tesla



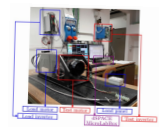
Rocket landing



Wind farm [Markovsky]



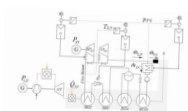
Buildings [Lian]



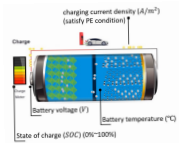
Motor drives [Carlet]



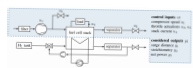
Greenhouse [Kerkhof]



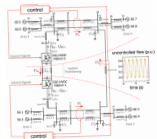
Power plant [Mahdavipour]




Batteries [Chen]

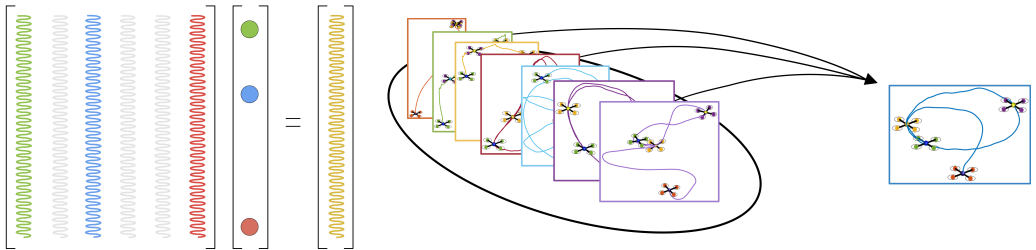


Fuel cell [Schmitt]



Power systems

# Summary



- A matrix of time series is a system representation: no identification step
- regularization $\Leftrightarrow$ robustification
- Deployed on quadcopters, excavators, legged robots, grids, buildings, growing tomatoes, ...
- DeePC is simple and it works

Ongoing: adaptation with streaming data, contact-rich and hybrid systems, nonlinear DeePC

# Thanks!

Jeremy Coulson

jeremy.coulson@wisc.edu

University of Wisconsin–Madison

Happy to talk about putting this on your system