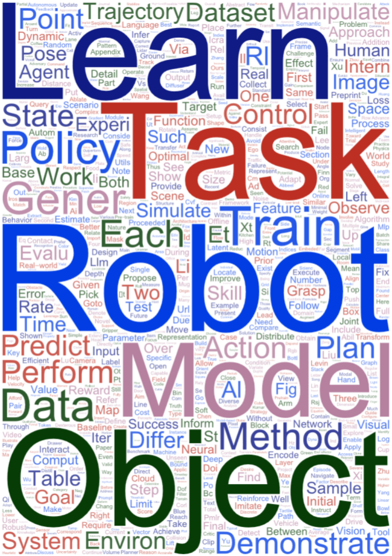



# Robot Learning

## Dynamics Learning

*(aka. System Identification, Model Learning)*

Marc Toussaint & Wolfgang Hönig
Technical University of Berlin
Summer 2024

# Outline

- I. What is learned?
  - Incl. which mapping exactly, model assumption, parameterization, loss function

- II. How is the data generated?

- III. Multirotor Examples

# I. What is learned?

# I. What is learned?

**environment/task parameters**

instructions/lang./goal info $g$
physics parameters $\Theta$

**state evaluations**

rewards $r_t$
value $V(x)$
Q-value $Q(x, u)$
constraint $\phi(x)$

state
$x_t$

controls
$u_t$

observations
$y_t$

**plans/anticipation**

waypoints/subgoals $x_{t_{1:K}}$
trajectory $x_{[t,t+H]}$
action plan $a_{1:K}$

# Dynamics Learning – State-based view

- Learning the *state-based* dynamics:

$$x_t = f(x_{t\text{-}1}, u_{t\text{-}1}) \qquad \text{or} \qquad p(x_t \,|\, x_{t\text{-}1}, u_{t\text{-}1})$$

# Dynamics Learning – State-based view

- Learning the *state-based* dynamics:

$$x_t = f(x_{t\text{-}1}, u_{t\text{-}1}) \qquad \text{or} \qquad p(x_t \,|\, x_{t\text{-}1}, u_{t\text{-}1})$$

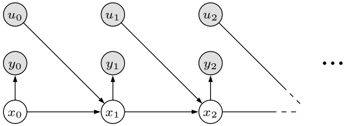
- Distinguish three cases:
  - **Parameter Estimation:** $f$ is assumed physics with unknown physics parameters $\Theta$
  - **Full Regression:** $f$ is learned as regression model
  - **Residual Dynamics:** learn the difference to a nominal physics model

# Dynamics Learning – Observation-based view

- $x_t$ is the system *state*

  [Markov Property: We call a variable *state* if the future is conditionally independent on the past when conditioned on state; $I(\text{future}, \text{past} \mid \text{state}) = 0$.]

- Sometimes the true state is not observed (or unknown), only observations $y_t$ are available ($y_t$: sensor readings, or *state estimates* from sensors)

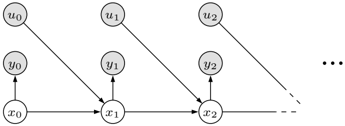


# Dynamics Learning – Observation-based view

- $x_t$ is the system *state*

  [Markov Property: We call a variable *state* if the future is conditionally independent on the past when conditioned on state; $I(\text{future}, \text{past} \mid \text{state}) = 0$.]

- Sometimes the true state is not observed (or unknown), only observations $y_t$ are available ($y_t$: sensor readings, or *state estimates* from sensors)




- We need to use the **history** of observed $y_t, u_t$ to predict next $y_t$!
- Distinguish three cases:
  - **Autoregression:** Learn a direct history-based model $y_t = f(y_{t-H:t}, u_{t-H:t})$
  - **Recurrent Model:** Learn a recurrent model with latent state $h_t$ (e.g. LSTM)
  - **State-space Model:** Jointly learn embedding/decoding $x \mapsto y$ and latent dynamics $x, u \mapsto x'$ (is also a recurrent model)

- In summary, six cases we'll discuss more concretely:
  - state-based dynamics
    - physical parameter estimation
    - full regression
    - residual dynamics
  - observation-based dynamics
    - autoregression (NARX)
    - observation-based dynamics – recurrent model
    - observation-based dynamics – state-space model

- Why learn the dynamics?
  - Given learned dynamics, we can use planning (MPC) or RL against the learned model to generate controllers
  - Examples in literature: Schaal'02, Deisenroth'15 (PILCO!), Finn'17, Driess'23, Schubert'23

- Why learn the dynamics?
  - Given learned dynamics, we can use planning (MPC) or RL against the learned model to generate controllers
  - Examples in literature: Schaal'02, Deisenroth'15 (PILCO!), Finn'17, Driess'23, Schubert'23

- Quick terminology:
  - Dynamics Learning $\leftrightarrow$ System Identification (in control theory), Model Learning (in model-based RL)
  - In control theory $u_t$ are called **inputs** and the *observations/measurements* $y_t$ are called **outputs**

## State Dynamics – Parameter Estimation

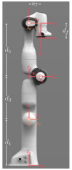
- Assume that dynamics $x_t = f_\Theta(x_{t\text{-}1}, u_{t\text{-}1})$ has unknown physical parameters $\Theta$,

# State Dynamics – Parameter Estimation

- Assume that dynamics $x_t = f_\Theta(x_{t\text{-}1}, u_{t\text{-}1})$ has unknown physical parameters $\Theta$,e.g.:

**Dynamic Identification of the Franka Emika Panda Robot with Retrieval of Feasible Parameters Using Penalty-based Optimization**

Claudio Gaz[1] Marco Cognetti[2] Alexander Oliva[3] Paolo Robuffo Giordano[2] Alessandro De Luca[1]



| $i$ | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | $d_1$ | $q_1$ |
| 2 | 0 | $-\pi/2$ | 0 | $q_2$ |
| 3 | 0 | $\pi/2$ | $d_3$ | $q_3$ |
| 4 | $a_4$ | $\pi/2$ | 0 | $q_4$ |
| 5 | $a_5$ | $-\pi/2$ | $d_5$ | $q_5$ |
| 6 | 0 | $\pi/2$ | 0 | $q_6$ |
| 7 | $a_7$ | $\pi/2$ | 0 | $q_7$ |
| 8 | 0 | 0 | $d_f$ | 0 |

Fig. 1. Denavit-Hartenberg frames and table of parameters for the Franka Emika Panda. The reference frames follow the modified Denavit-Hartenberg convention. In the figure, $d_1 = 0.333$ m, $d_3 = 0.316$ m, $d_5 = 0.384$ m, $d_f = 0.107$ m, $a_4 = 0.0825$ m, $a_5 = -0.0825$ m, $a_7 = 0.088$ m.

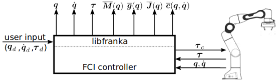



Fig. 2. Signal flows from and to the controller. The user sends a command to the libfranka interface that communicates with the FCI controller. This input is then converted to a commanded torque $\boldsymbol{\tau}_c$ to the robot that returns the measured joint torque $\boldsymbol{\tau}$, as well as the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$. The FCI controller computes the numerical values for the inertia matrix $\overline{\boldsymbol{M}}(\boldsymbol{q})$, as well for the gravity vector $\overline{\boldsymbol{g}}(\boldsymbol{q})$, the Jacobian $\overline{\boldsymbol{J}}(\boldsymbol{q})$, and the Coriolis term $\overline{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. These data are sent back to the user through the libfranka interface. A more detailed description of the FCI can be found at: https://frankaemika.github.io/docs/index.html.

consistency of the parameters. The identification procedure

Claudio Gaz, Marco Cognetti, Alexander Oliva, Paolo Robuffo Giordano, and Alessandro De Luca, (2019). Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization.
*IEEE Robotics and Automation Letters*, 4(4):4147–4154

# State Dynamics – Parameter Estimation

- Given data $D = \{(x_t, x_{t\text{-}1}, u_{t\text{-}1})\}_{t=1}^T$, find parameters

$$\min_\Theta \sum_t \|x_t - f_\Theta(x_{t\text{-}1}, u_{t\text{-}1})\|^2$$

## State Dynamics – Parameter Estimation

- Given data $D = \{(x_t, x_{t\text{-}1}, u_{t\text{-}1})\}_{t=1}^T$, find parameters

$$\min_\Theta \sum_t \|x_t - f_\Theta(x_{t\text{-}1}, u_{t\text{-}1})\|^2$$

- Sometimes, it is possible to describe $f_\Theta$ as linear in $\Theta$. See Gaz'19!
  - Then finding optimal $\Theta$ leads to a linear least squares problem.
  - Otherwise: Black-box optimization (CMA-ES) or gradient-based (SGD, Gauss-Newton)

# State Dynamics – Full Regression

- Learn $f_\theta$ directly, using some ML regression, e.g. (old-fashioned LWR):

**Scalable Techniques from Nonparametric Statistics for Real Time Robot Learning**


STEFAN SCHAAL
*Computer Science and Neuroscience, HNB-103, University of Southern California, Los Angeles, CA 90089-2520, USA; Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan*
sschaal@usc.edu; www.clmc.usc.edu

CHRISTOPHER G. ATKESON
*College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280, USA; ATR Human Information Processing Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan*
cga@cc.gatech.edu; www.cc.gatech.edu/fac/Chris.Atkeson

SETHU VIJAYAKUMAR
*Computer Science and Neuroscience, HNB-103, University of Southern California, Los Angeles, CA 90089-2520, USA; Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan*
sethu@usc.edu; www.clmc.usc.edu

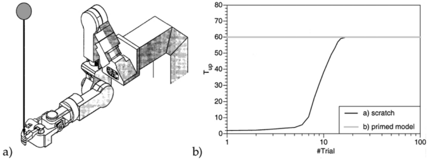




*Figure 3.* (a) Sarcos Dexterous Robot Arm; (b) Smoothed average of 10 learning curves of the robot for pole balancing. The trials were aborted after successful balancing of 60 seconds. We also tested long term performance of the learning system by running pole balancing for over an hour—the pole was never dropped.


Stefan Schaal, Christopher G. Atkeson, and Sethu Vijayakumar, (2002). Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60

# State Dynamics – Full Regression

- Given data $D = \{(x_t, x_{t\text{-}1}, u_{t\text{-}1})\}_{i=1:n,t=1:T_i}$, find parameters

$$\min_\theta \sum_t \|x_t - f_\theta(x_{t\text{-}1}, u_{t\text{-}1})\|^2$$

$\to$ same formulation as parameter estimation, really.

## State Dynamics – Full Regression

- Given data $D = \{(x_t, x_{t\text{-}1}, u_{t\text{-}1})\}_{i=1:n, t=1:T_i}$, find parameters

$$\min_\theta \sum_t \|x_t - f_\theta(x_{t\text{-}1}, u_{t\text{-}1})\|^2$$

$\to$ same formulation as parameter estimation, really.

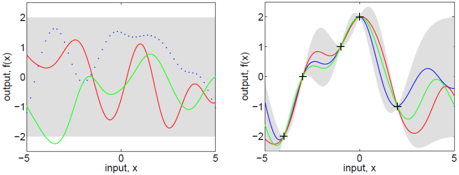
- Use supervised ML to minimize regression error

# State Dynamics – Full Regression (probabilistic)

- Given data $D = \{(x_t, x_{t\text{-}1}, u_{t\text{-}1})\}_{i=1:n, t=1:T_i}$, find parameters

$$\min_\theta - \sum_t \log p_\theta(x_t \,|\, x_{t\text{-}1}, u_{t\text{-}1})$$

where $p_t(x_t \,|\, x_{t\text{-}1}, u_{t\text{-}1})$ is a probabilistic regression, e.g. Gaussian Process:



(a), prior (b), posterior

(from Rasmussen & Williams)

[Marc Deisenroth's PICLO paper had huge impact: Using learned GP dynamics to derive optimal controls.]

## State Dynamics – Residual Dynamics

- Given a nominal dynamics $f_M$ (e.g., assumed physics), learn a residual model $f_\theta$ to minimze

$$\min_\theta \sum_t \|x_t - [f_M(x_{t\text{-}1}, u_{t\text{-}1}) + f_\theta(x_{t\text{-}1}, u_{t\text{-}1})]\|^2$$

# State Dynamics – Residual Dynamics

- Given a nominal dynamics $f_M$ (e.g., assumed physics), learn a residual model $f_\theta$ to minimze

$$\min_\theta \sum_t \|x_t - [f_M(x_{t\text{-}1}, u_{t\text{-}1}) + f_\theta(x_{t\text{-}1}, u_{t\text{-}1})]\|^2$$

- Examples: Gaz'19, Multirotor Examples

# Observation-based Dynamics – Autoregression (NARX)

208 IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 27, NO. 2, APRIL 1997

## Computational Capabilities of Recurrent NARX Neural Networks

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, *Senior Member, IEEE*

***Abstract*—Recently, fully connected recurrent neural networks have been proven to be computationally rich—at least as powerful as Turing machines. This work focuses on another network which is popular in control applications and has been found to be very effective at learning a variety of problems. These networks are based upon Nonlinear AutoRegressive models with eXogenous Inputs (NARX models), and are therefore called *NARX networks*. As opposed to other recurrent networks, NARX networks have a limited feedback which comes only from the output neuron rather than from hidden states. They are formalized by**

$$y(t) = \Psi(u(t-n_u), \cdots, u(t-1), u(t), y(t-n_y), \cdots, y(t-1))$$

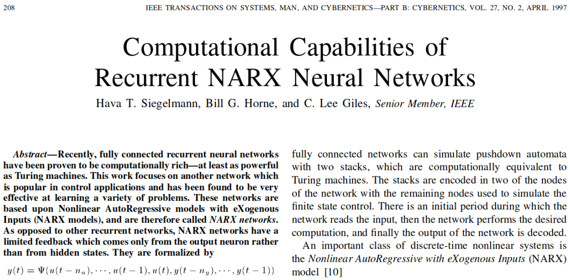
fully connected networks can simulate pushdown automata with two stacks, which are computationally equivalent to Turing machines. The stacks are encoded in two of the nodes of the network with the remaining nodes used to simulate the finite state control. There is an initial period during which the network reads the input, then the network performs the desired computation, and finally the output of the network is decoded.

An important class of discrete-time nonlinear systems is the *Nonlinear AutoRegressive with eXogenous Inputs* (NARX) model [10]

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, (1997). Computational capabilities of recurrent NARX neural networks.
*IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215

- NARX="Autoregression with controls" our notation: $y_t = f_\theta(y_{t\text{-}H:t\text{-}1}, u_{t\text{-}H:t\text{-}1})$
- developed in time-series modelling, sequence modelling

# Observation-based Dynamics – Autoregression (NARX)



208 IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 27, NO. 2, APRIL 1997

## Computational Capabilities of Recurrent NARX Neural Networks

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, *Senior Member, IEEE*

***Abstract*—Recently, fully connected recurrent neural networks have been proven to be computationally rich—at least as powerful as Turing machines. This work focuses on another network which is popular in control applications and has been found to be very effective at learning a variety of problems. These networks are based upon Nonlinear AutoRegressive models with eXogenous Inputs (NARX models), and are therefore called *NARX networks*. As opposed to other recurrent networks, NARX networks have a limited feedback which comes only from the output neuron rather than from hidden states. They are formalized by**

$$y(t) = \Psi(u(t-n_u), \cdots, u(t-1), u(t), y(t-n_y), \cdots, y(t-1))$$

fully connected networks can simulate pushdown automata with two stacks, which are computationally equivalent to Turing machines. The stacks are encoded in two of the nodes of the network with the remaining nodes used to simulate the finite state control. There is an initial period during which the network reads the input, then the network performs the desired computation, and finally the output of the network is decoded.

An important class of discrete-time nonlinear systems is the *Nonlinear AutoRegressive with eXogenous Inputs* (NARX) model [10]

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, (1997). Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215

- – NARX="Autoregression with controls" our notation: $y_t = f_\theta(y_{t\text{-}H:t\text{-}1}, u_{t\text{-}H:t\text{-}1})$
- – developed in time-series modelling, sequence modelling

- How long does the history $H$ have to be?

# Observation-based Dynamics – Autoregression (NARX)

208 IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 27, NO. 2, APRIL 1997

## Computational Capabilities of Recurrent NARX Neural Networks

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, *Senior Member, IEEE*

***Abstract*—Recently, fully connected recurrent neural networks have been proven to be computationally rich—at least as powerful as Turing machines. This work focuses on another network which is popular in control applications and has been found to be very effective at learning a variety of problems. These networks are based upon Nonlinear AutoRegressive models with eXogenous Inputs (NARX models), and are therefore called *NARX networks*. As opposed to other recurrent networks, NARX networks have a limited feedback which comes only from the output neuron rather than from hidden states. They are formalized by**

$$y(t) = \Psi(u(t-n_u), \cdots, u(t-1), u(t), y(t-n_y), \cdots, y(t-1))$$

fully connected networks can simulate pushdown automata with two stacks, which are computationally equivalent to Turing machines. The stacks are encoded in two of the nodes of the network with the remaining nodes used to simulate the finite state control. There is an initial period during which the network reads the input, then the network performs the desired computation, and finally the output of the network is decoded.

An important class of discrete-time nonlinear systems is the *Nonlinear AutoRegressive with eXogenous Inputs* (NARX) model [10]

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, (1997). Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215

  - NARX="Autoregression with controls" our notation: $y_t = f_\theta(y_{t\text{-}H:t\text{-}1}, u_{t\text{-}H:t\text{-}1})$
  - developed in time-series modelling, sequence modelling

- How long does the history $H$ have to be?
- What's the modern version of autoregression?

# Observation-based Dynamics – Autoregression (Transformers)

DeepMind 2023-9-26

## A Generalist Dynamics Model for Control

Ingmar Schubert[*,1], Jingwei Zhang[2], Jake Bruce[2], Sarah Bechtle[2], Emilio Parisotto[2], Martin Riedmiller[2], Jost Tobias Springenberg[2], Arunkumar Byravan[2], Leonard Hasenclever[2] and Nicolas Heess[2]
[1]TU Berlin, [2]DeepMind, [*]Work done at DeepMind

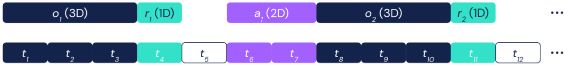



Figure 2 | Illustration of the tokenization for $n = 3$ and $m = 2$. Starting from $o_1$, performing action $a_1$ will result in the next observation $o_2$ and the reward $r_2$. The constant separator tokens $t_5$ and $t_{12}$ are inserted to indicate the start of a new environment step.

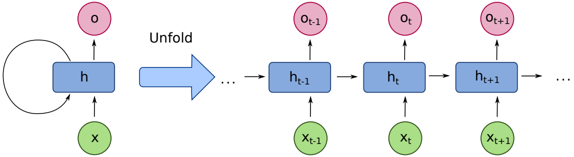
Ingmar Schubert, Jingwei Zhang, Jake Bruce, Sarah Bechtle, Emilio Parisotto, Martin Riedmiller, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, and Nicolas Heess, (2023). A generalist dynamics model for control

# Observation-based Dynamics – Recurrent Model

- Rather than giving the model a history as input, it should *learn* to memorize relevant information, i.e., learn a latent representation for relevant information $\rightarrow$ recurrent NN

# Observation-based Dynamics – Recurrent Model

- Rather than giving the model a history as input, it should *learn* to memorize relevant information, i.e., learn a latent representation for relevant information $\rightarrow$ recurrent NN
- Train a latent representation $h_t$ to consume history information and predict $y_t$

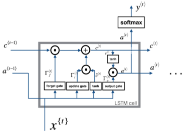



(Wikipedia; change in notation: $x \rightsquigarrow (y,u), o \rightsquigarrow y$)

- The most common NN architecture is LSTM (better: Gated Recurrent Units):

**2 - Long Short-Term Memory (LSTM) network**

This following figure shows the operations of an LSTM-cell.




$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$
$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$
$$\tilde{c}^{\langle t \rangle} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$
$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$
$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$
$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

**Figure 4**: LSTM-cell. This tracks and updates a "cell state" or memory variable $c^{\langle t \rangle}$ at every time-step, which can be different from $a^{\langle t \rangle}$.

(Hochreiter, Schmidthuber, 1997)

# Observation-based Dynamics – State-Space Model

- Also a recurrent model, but explicitly assumes latent state $x_t \in \mathbb{R}^d$

## Probabilistic Recurrent State-Space Models

**Andreas Doerr** [1 2] **Christian Daniel** [1] **Martin Schiegg** [1] **Duy Nguyen-Tuong** [1] **Stefan Schaal** [2 3] **Marc Toussaint** [4]
**Sebastian Trimpe** [2]

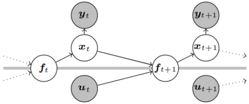



*Figure 1.* Graphical model of the PR-SSM. Gray nodes are observed variables in contrast to latent variables in white nodes. Thick lines indicate variables, which are jointly Gaussian under a GP prior.

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian, (2018). Probabilistic recurrent state-space models. In *International conference on machine learning*, pages 1280–1289

# Observation-based Dynamics – State-Space Model

- Jointly train an embedding/decoding $g : x \mapsto y$ and latent dynamics $f : x, u \mapsto x'$:

$$\begin{array}{ccc} x & , \mathbf{u} \overset{f}{\mapsto} & x' \\ g\downarrow & & g\downarrow \\ \mathbf{y} & & \mathbf{y}' \end{array}$$

- Only $u_{1:T}, y_{1:T}$ are observed! Train model to maximize data likelihood,

$$\log p(y_{1:T} \,|\, u_{1:T}) \geq \text{Evidence Lower Bound (ELBO)}$$

  - This method trains both, $g$ and $f$, and implicitly *infers* a notion of state $x_t$
  - Technically, use SGD to maximize ELBO

- More Literature for the six cases provided at the end of these slides...

# II. How is the data generated?

# II. How is the data generated?

- Importance of data generation is (mostly) under-acknowledged in papers!
- Ideas to generate *good* data may be more important than ML method details

## II. How is the data generated?

- Importance of data generation is (mostly) under-acknowledged in papers!
- Ideas to generate *good* data may be more important than ML method details
- What is *good* data?

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar x^\top \theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \ + \ \lambda \|\theta\|^2$$

  and solution

$$\theta^* = (X^\top X + \lambda \mathbf{I})^{\text{-}1} X^\top y \ .$$

- What is good data?

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar{x}^\top\theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \ + \ \lambda\|\theta\|^2$$

  and solution

$$\theta^* = (X^\top X + \lambda\mathbf{I})^{\text{-}1} X^\top y \ .$$

- What is good data?
- What is the estimator variance Var$\{\theta^*\}$?

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar{x}^\top\theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \; + \; \lambda\|\theta\|^2$$

and solution

$$\theta^* = (X^\top X + \lambda\mathbf{I})^{\text{-}1} X^\top y \; .$$

- What is good data?
- What is the estimator variance Var$\{\theta^*\}$?
  - Assume data with variance Var$\{y\} = \sigma^2\mathbf{I}_n$

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar{x}^\top \theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \ + \ \lambda \|\theta\|^2$$

  and solution

$$\theta^* = (X^\top X + \lambda \mathbf{I})^{-1} X^\top y \ .$$

- What is good data?
- What is the estimator variance Var$\{\theta^*\}$?
  - Assume data with variance Var$\{y\} = \sigma^2 \mathbf{I}_n$
  - Then Var$\{\theta^*\} = (X^\top X + \lambda I)^{-1} \sigma^2$

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar{x}^\top\theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \ + \ \lambda\|\theta\|^2$$

  and solution

$$\theta^* = (X^\top X + \lambda\mathbf{I})^{\text{-1}} X^\top y \ .$$

- What is good data?
- What is the estimator variance Var$\{\theta^*\}$?
  - Assume data with variance Var$\{y\} = \sigma^2\mathbf{I}_n$
  - Then Var$\{\theta^*\} = (X^\top X + \lambda I)^{\text{-1}}\sigma^2$
  - Smaller variance via larger $\lambda$ (but then larger bias), or **larger** $\det(X^\top X)$!

# Good Data – in Linear Regression

- Reconsider regression with linear model $f_\theta(x) = \bar x^\top \theta$, loss

$$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2 \ + \ \lambda \|\theta\|^2$$

  and solution

$$\theta^* = (X^\top X + \lambda \mathbf{I})^{\text{-}1} X^\top y \ .$$

- What is good data?
- What is the estimator variance Var$\{\theta^*\}$?
  - Assume data with variance Var$\{y\} = \sigma^2 \mathbf{I}_n$
  - Then Var$\{\theta^*\} = (X^\top X + \lambda I)^{\text{-}1} \sigma^2$
  - Smaller variance via larger $\lambda$ (but then larger bias), or **larger** $\det(X^\top X)$!
- Good data means reducing variance (=randomness) of estimated model!
  - large $\det(X^\top X) \leftrightarrow$ cover input space!

    [Large estimator variance $\leftrightarrow$ "Overfitting": Reducing variance prevents overfitting. Hastie has great section on *shrinkage* methods (=regularization)]

# Good Data – in Linear System Identification

Signals and Systems

Lecture 11: System Identification

Dr. Guillaume Ducard

Fall 2018

based on materials from: Prof. Dr. Raffaello D'Andrea

Institute for Dynamic Systems and Control

ETH Zurich, Switzerland

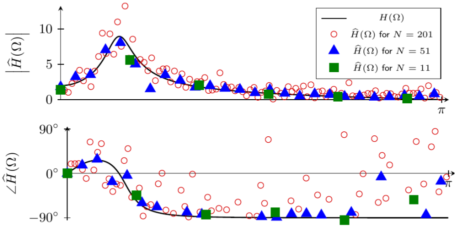
https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/Signals-and-Systems/Lectures/Fall2018/Lecture11_sigsys.pdf

# Good Data – in Linear System Identification

- Cover the input space $\to$ cover frequency space
  - Linear dynamics can be Laplace transformed into frequency domain:

$$Y(s) = H(s)\, U(s)$$

  - $U(s)$ are controls; $Y$ observations; $H(s)$ is called **transfer function** (complex)
  - $H(s)$ can be probed by sending a single control frequence ($U(s) = \delta_{ss'}$)



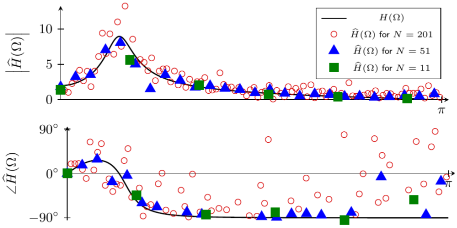

- In essence: stimulate the system with control frequencies $u(t) = \cos(kt/\tau_0)$ for $k = 0, 1, ..$

# Good Data – in Linear System Identification

- Cover the input space $\to$ cover frequency space
  - Linear dynamics can be Laplace transformed into frequency domain:

$$Y(s) = H(s)\, U(s)$$

  - $U(s)$ are controls; $Y$ observations; $H(s)$ is called **transfer function** (complex)
  - $H(s)$ can be probed by sending a single control frequence ($U(s) = \delta_{ss'}$)




- In essence: stimulate the system with control frequencies $u(t) = \cos(kt/\tau_0)$ for $k = 0, 1, ..$
- Franka SystemId paper [Gaz'19]: Sinusoidal reference motions (Eq. 31):

$$\dot{q}_{i,\text{des}(t)} = A_i \sin\left(\frac{2\pi}{T_i}\, t\right) \ , \quad i \in \{1, .., n\}$$

# Good Data – in general

- Think about good state space coverage!  (in all variants of Robot Learning)
  - Frequency coverage in control systems
  - Exploration in RL beyond $\epsilon$-greedy
  - Long-term structured variation (at least pink noise, Ornstein-Uhlenbeck) instead of Brownian motion
  - Explicit exploration: Novelty seeking, information seeking, exploration bonus, Bayesian RL

# III. Background: Multirotors

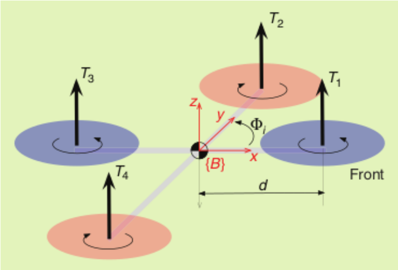
- State $\mathbf{x} = (\mathbf{p}, \mathbf{q}, \mathbf{v}, \omega)^\top$
- Control $\mathbf{u}_\Omega = (\Omega_1, \dots, \Omega_n)^\top$
- Forces $\mathbf{f} = \sum_i c_{f_i} \Omega_i \mathbf{z}_{\Omega_i} = \mathbf{F}\mathbf{u}_\Omega$,
- Torques $\boldsymbol{\tau} = \sum_i (c_{f_i} \mathbf{p}_{\Omega_i} \times \mathbf{z}_{\Omega_i} + c_{\tau_i} \mathbf{z}_{\Omega_i}) \Omega_i = \mathbf{M}\mathbf{u}_\Omega$
- Dynamics

$$\dot{\mathbf{p}} = \mathbf{v}, \qquad m\dot{\mathbf{v}} = m\mathbf{g} + \mathbf{R}(\mathbf{q})\mathbf{F}\mathbf{u}_\Omega + \mathbf{f}_a,$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \circ \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{M}\mathbf{u}_\Omega + \boldsymbol{\tau}_a,$$

[Propellers create forces and torques, rest is Newton-Euler]

[$\mathbf{f}_a, \boldsymbol{\tau}_a$ can model drag, wind, aerodynamic interactions etc.]




[Mahony, ~2012]

# Multirotors: What is learned?

- Parameters that are hard to measure: inertia $\mathbf{J}$, motor params ($c_{f_i}$, $c_{\tau_i}$, delay)

- Residuals $\mathbf{f}_a$, $\boldsymbol{\tau}_a$
  [potentially as a function of the state (e.g., drag) or environment (e.g., downwash)]
  [potentially non-Markovian, i.e., a function of a history of states]

- Full dynamics model not so much — Why?

# Multirotors: What is learned?

- Parameters that are hard to measure: inertia $\mathbf{J}$, motor params ($c_{f_i}$, $c_{\tau_i}$, delay)

- Residuals $\mathbf{f}_a$, $\boldsymbol{\tau}_a$

  [potentially as a function of the state (e.g., drag) or environment (e.g., downwash)]

  [potentially non-Markovian, i.e., a function of a history of states]

- Full dynamics model not so much — Why?

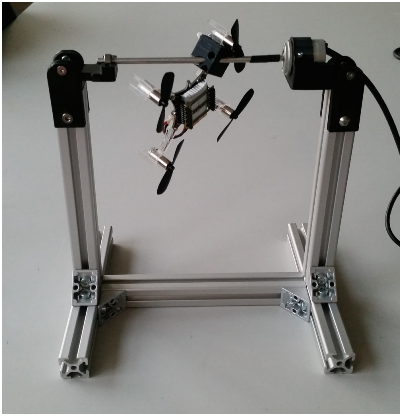
  [Impossible to gather data from all states safely]

  [Rotational symmetries are surprisingly difficult to learn]

# Multirotors: How is it “learned”? (Classic)

Estimate parameters with dedicated experiments

- Inertia: Swing body in different positions and record motion; solve an optimization problem



# Multirotors: How is it "learned"? (Classic)

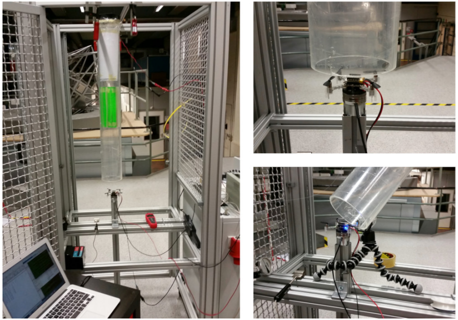
Estimate parameters with dedicated experiments

- Motors: Use thrust stand (often for a single motor + propeller) + curve fitting

# Multirotors: How is it "learned"? (Classic)

Estimate parameters with dedicated experiments

- Drag: Use wind tunnel + curve fitting with "guessed" models



Julian Förster, (2015). System identification of the crazyflie 2.0 nano quadrocopter

# Multirotors: How is it “learned”? (Classic)

Estimate parameters with dedicated experiments

- Is this learning?

# Multirotors: How is it “learned”? (Classic)

Estimate parameters with dedicated experiments

- Is this learning?

  [Yes, since curve fitting is extensively used]

- Advantages and Disadvantages?

# Multirotors: How is it “learned”? (Classic)

Estimate parameters with dedicated experiments

- Is this learning?

  [Yes, since curve fitting is extensively used]

- Advantages and Disadvantages?

  [Pros: Physics intuition (explainability); can improve “important” parameters if needed; no need to have a flying system]

  [Cons: Labor and equipment intensive; does not capture unmodeled terms; does not capture the robot as a system]

# Multirotors: How is it learned? (Parameter Estimation)

- Assumption: we have a system that can already fly; Can we do better?

  [Strong assumption, since controllers need models, too]

- Direct (analytical) optimization

  Jonas Eschmann, Dario Albani, and Giuseppe Loianno, (2024). Data-driven system identification of quadrotors subject to motor delays

  [Will skip the discussion here]

- Probabilistic formulation (Gaussian noise)

  Michael Burri, Janosch Nikolic, Helen Oleynikova, Markus W. Achtelik, and Roland Siegwart, (2016). Maximum likelihood parameter identification for MAVs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4297–4303

# Multirotors: How is it learned? (Maximum Likelihood)

- Given: Dataset with trajectory (position, orientation, motor speed), $\mathbf{Z}$; measurements (IMU data, motor commands), $\mathbf{U}$
- Goal:

$$\hat{\mathbf{X}}_{ML}, \hat{\theta}_{ML} = \underset{\hat{\mathbf{X}}, \hat{\theta}}{\operatorname{argmax}}\, p(\mathbf{Z}, \mathbf{U}, \hat{\mathbf{X}}, \hat{\theta})$$

(parameters to estimate $\hat{\theta}$; state estimates $\hat{\mathbf{X}}$; probability $p$)

# Multirotors: How is it learned? (Maximum Likelihood)

- Assumptions to simplify $p(\mathbf{Z}, \mathbf{U}, \hat{\mathbf{X}}, \hat{\theta})$
  - White noise (IMU, motors)
  - Access to a prior trajectory $\rightarrow$ linearize around it and reason about "residuals" instead
- $p(\cdot)$ becomes a mixture of Gaussians $\rightarrow$ can be maximized by minimizing the negative log-likelihood

  [essentially a least square problem]

# Multirotors: How is it learned? (Maximum Likelihood)

$n := 0$

$\bar{\boldsymbol{y}} := \textsc{InitializeEstimator}\ ()$

% Solve ML problem

**while** $n < n_{max}$ **do**

$\quad \boldsymbol{b}, \boldsymbol{A} := \textsc{EvaluateResiduals}\ (\bar{\boldsymbol{y}})$

$\quad \delta\boldsymbol{y} := \textsc{SolveLeastSquaresProblem}\ (\boldsymbol{b}, \boldsymbol{A})$

$\quad \bar{\boldsymbol{y}} = \bar{\boldsymbol{y}} \boxplus \delta\boldsymbol{y}$

$\boldsymbol{\theta}^* := \textsc{ExtractParameters}\ (\bar{\boldsymbol{y}})$

$\boldsymbol{\Sigma}_\theta := \textsc{RecoverParameterCovariance}\ (\boldsymbol{A})$

**return** $\boldsymbol{\theta}^*, \boldsymbol{\Sigma}_\theta$

where $\bar{y} = (\hat{\mathbf{X}}, \hat{\theta})^\top$ from before

Michael Burri, Janosch Nikolic, Helen Oleynikova, Markus W. Achtelik, and Roland Siegwart, (2016). Maximum likelihood parameter identification for MAVs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4297–4303

Michael Burri, Michael Bloesch, Zachary Taylor, Roland Siegwart, and Juan Nieto, (2018). A framework for maximum likelihood parameter identification applied on MAVs. *Journal of Field Robotics*, 35(1):5–22

# Multirotors: How is it learned? (Supervised Deep NN)

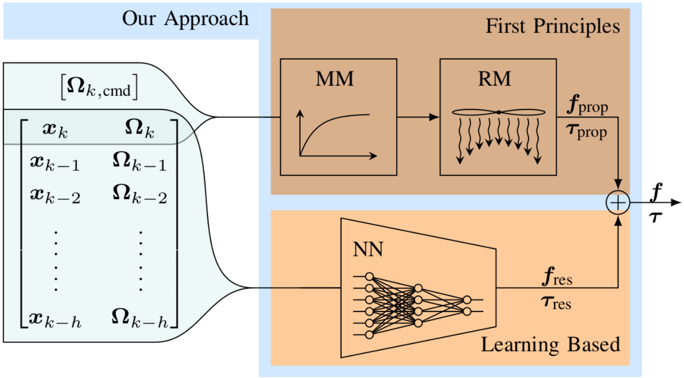
- Basic models do not capture "complicated" aerodynamic effects
- Blade Element Momentum (BEM) work for single rotors (but high computational effort)
- Can we use (more) data to use function approximation instead?
  Challenges:
  - Training/Data efficiency
  - Inference speed

# Multirotors: How is it learned? (Supervised Deep NN)

- Key idea: learn the “residual physics”, only

  [Input: past $h$ states and motor commands $\rightarrow$ not Markovian!]

  [Output: forces and torques that cannot be explained by the basic model(s) $(\mathbf{f}_a, \boldsymbol{\tau}_a)$]



# Multirotors: How is it learned? (Supervised Deep NN)

- ML method: Supervised training — Where do the labels come frome?

# Multirotors: How is it learned? (Supervised Deep NN)

- ML method: Supervised training — Where do the labels come frome?
  [Solve dynamics for $\mathbf{f}_a, \boldsymbol{\tau}_a$]

- Architecture
  - Input $h = 20$ (past 50 ms)
  - temporal convolutional (TCN) with 25k parameters (MLP and other parameters in ablation)

- Main takeaway: strong model/physics priors are better

Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza, (2021). NeuroBEM: Hybrid aerodynamic quadrotor model.
In *Robotics: Science and Systems XVII*, volume 17

Video: `https://youtu.be/Nze1wlfmzTQ`

# Multirotors: Data Collection

- Motion capture system for accurate position/orientation state estimates
  [Sampling at 500 Hz, submillimeter accuracy]
  [Very costly: EUR 20k – 100k]
- On-board data logging of IMU
  [Sampling at 1000 Hz, very noisy]

# Multirotors: Data Preprocessing

- Two data sources $\rightarrow$ Synchronization needed (incl. clock skew)
  - Online Option: Send data to one computer using a low-latency link (and account for link delay)
  - Offline Option: Solve optimization problem for clock skew and bias

- Some derivatives (e.g., $\mathbf{v}$) are not directly observable
  - Online Option: Use data from an online filter (e.g., Extended Kalman Filter)
  - Offline Option: Interpolate data (e.g., using splines), use analytical solution of fitted spline

- Motor delays ("easy" to measure)
  - Option 1: Include it in model explicitly
  - Option 2: Shift/filter data accordingly

# Multirotors: Data Quantity

- Maximum Likelihood: 45 sec flight data “The pilot was careful to excite all axes, especially in yaw direction.”
- NeuroBEM: 96 flights, 75 min flight data (1.8M data points) (up to 18 $m/s$ and 47 $m/s^2$)

# Literature

- State Dynamics – Parameter Estimation:


Julian Förster, (2015). System identification of the crazyflie 2.0 nano quadrocopter

Jonas Eschmann, Dario Albani, and Giuseppe Loianno, (2024). Data-driven system identification of quadrotors subject to motor delays

Michael Burri, Michael Bloesch, Zachary Taylor, Roland Siegwart, and Juan Nieto, (2018). A framework for maximum likelihood parameter identification applied on MAVs.
*Journal of Field Robotics*, 35(1):5–22

Claudio Gaz, Marco Cognetti, Alexander Oliva, Paolo Robuffo Giordano, and Alessandro De Luca, (2019). Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization.
*IEEE Robotics and Automation Letters*, 4(4):4147–4154


- State Dynamics – Full Regression:


Stefan Schaal, Christopher G. Atkeson, and Sethu Vijayakumar, (2002). Scalable techniques from nonparametric statistics for real time robot learning.
*Applied Intelligence*, 17(1):49–60

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen, (2015). Gaussian processes for data-efficient learning in robotics and control.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423

# Literature

- Observation-based Dynamics – Autoregression (NARX):


S. Chen, S. A. Billings, and P. M. Grant, (1990). Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214

Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, (1997). Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215


- Observation-based Dynamics – Recurrent Model (also visual!):


Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza, (2021). NeuroBEM: Hybrid aerodynamic quadrotor model. In *Robotics: Science and Systems XVII*, volume 17

Chelsea Finn and Sergey Levine, (2017). Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793

Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint, (2023). Learning multi-object dynamics with compositional neural radiance fields. In *Conference on robot learning*, pages 1755–1768

Ingmar Schubert, Jingwei Zhang, Jake Bruce, Sarah Bechtle, Emilio Parisotto, Martin Riedmiller, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, and Nicolas Heess, (2023). A generalist dynamics model for control

# Literature

- State-Space Models (learning a *state* dynamics based on only observations):

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian, (2018). Probabilistic recurrent state-space models. In *International conference on machine learning*, pages 1280–1289

## not mentioned...

- Constrained ML models (Geist)
- Embed to Control
- Koopman embedding
- Dual control
- Safe Exploration

[1] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza, (2021).
NeuroBEM: Hybrid aerodynamic quadrotor model.
In *Robotics: Science and Systems XVII*, volume 17.

[2] Michael Burri, Michael Bloesch, Zachary Taylor, Roland Siegwart, and Juan Nieto, (2018).
A framework for maximum likelihood parameter identification applied on MAVs.
*Journal of Field Robotics*, 35(1):5–22.

[3] Michael Burri, Janosch Nikolic, Helen Oleynikova, Markus W. Achtelik, and Roland Siegwart, (2016).
Maximum likelihood parameter identification for MAVs.
In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4297–4303.

[4] S. Chen, S. A. Billings, and P. M. Grant, (1990).
Non-linear system identification using neural networks.
*International Journal of Control*, 51(6):1191–1214.

[5] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen, (2015).
Gaussian processes for data-efficient learning in robotics and control.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423.

[6] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian, (2018).
Probabilistic recurrent state-space models.
In *International conference on machine learning*, pages 1280–1289.

[7] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint, (2023).
Learning multi-object dynamics with compositional neural radiance fields.
In *Conference on robot learning*, pages 1755–1768.

[8] Jonas Eschmann, Dario Albani, and Giuseppe Loianno, (2024).
Data-driven system identification of quadrotors subject to motor delays.

[9] Chelsea Finn, and Sergey Levine, (2017).

Deep visual foresight for planning robot motion.
In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793.

[10] Julian Förster, (2015).
System identification of the crazyflie 2.0 nano quadrocopter.

[11] Claudio Gaz, Marco Cognetti, Alexander Oliva, Paolo Robuffo Giordano, and Alessandro De Luca, (2019).
Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization.
*IEEE Robotics and Automation Letters*, 4(4):4147–4154.

[12] Stefan Schaal, Christopher G. Atkeson, and Sethu Vijayakumar, (2002).
Scalable techniques from nonparametric statistics for real time robot learning.
*Applied Intelligence*, 17(1):49–60.

[13] Ingmar Schubert, Jingwei Zhang, Jake Bruce, Sarah Bechtle, Emilio Parisotto, Martin Riedmiller, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, and Nicolas Heess, (2023).
A generalist dynamics model for control.

[14] Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles, (1997).
Computational capabilities of recurrent NARX neural networks.
*IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215.