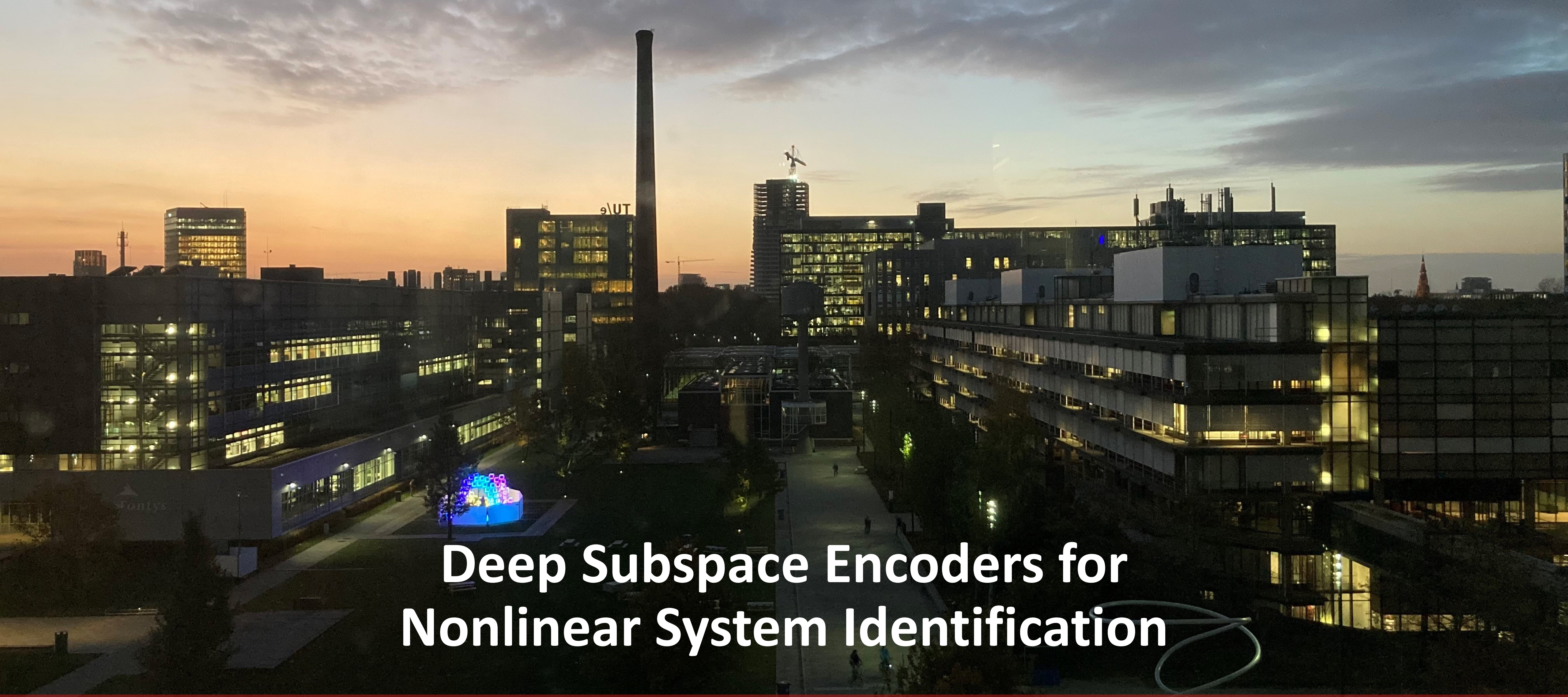
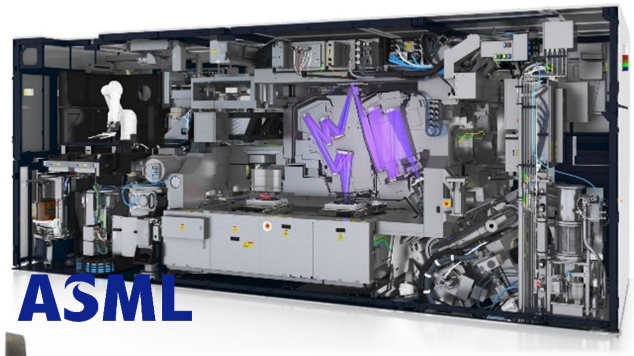


Deep Subspace Encoders for Nonlinear System Identification

Maarten Schoukens
m.schoukens@tue.nl

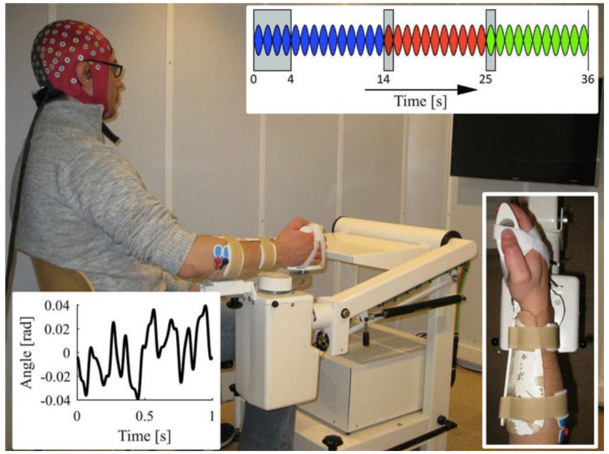
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
Electrical Engineering, Control Systems Group

Nonlinear Systems

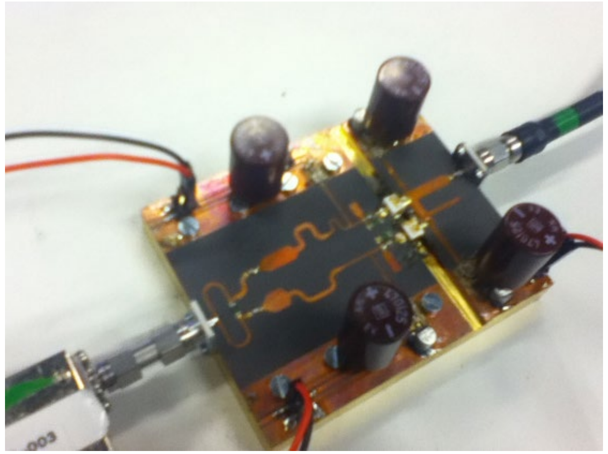


Motion Systems

Structural Dynamics



Human Body



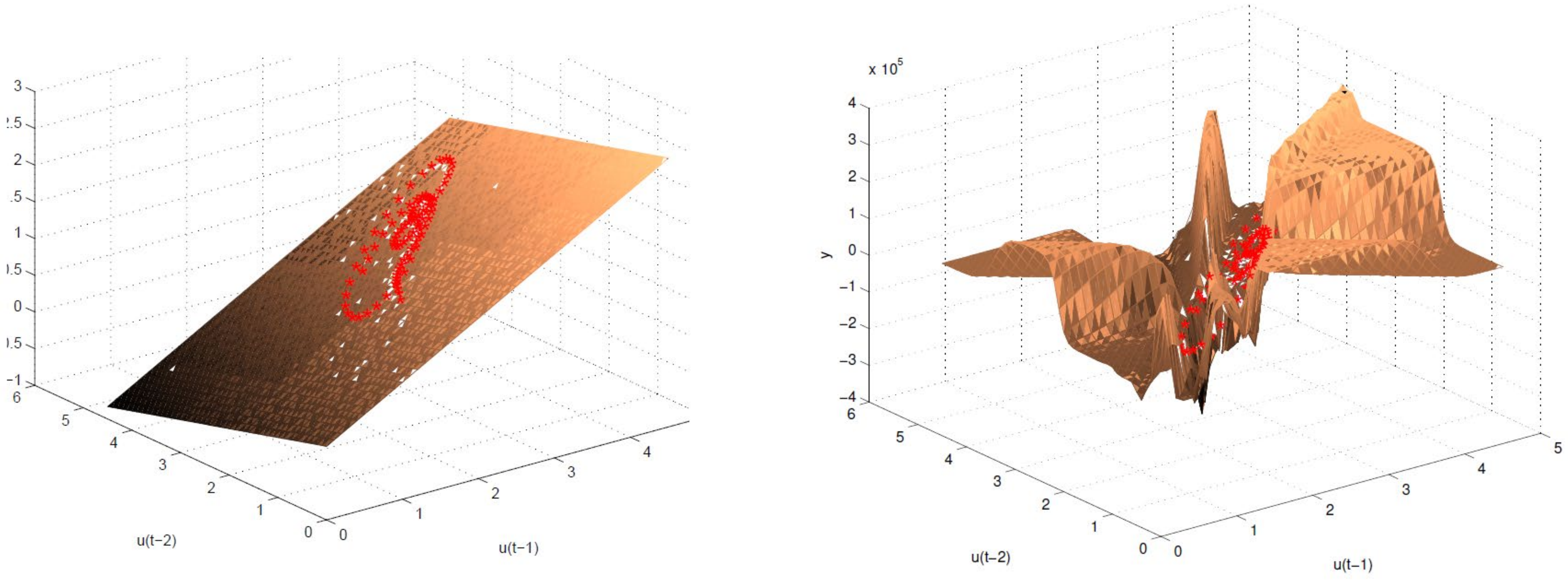
Electrical Circuits

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Nonlinear System Identification

Linear Identification: Characterize a **hyperplane**

Nonlinear Identification: Characterize a **manifold**



(a) (b)

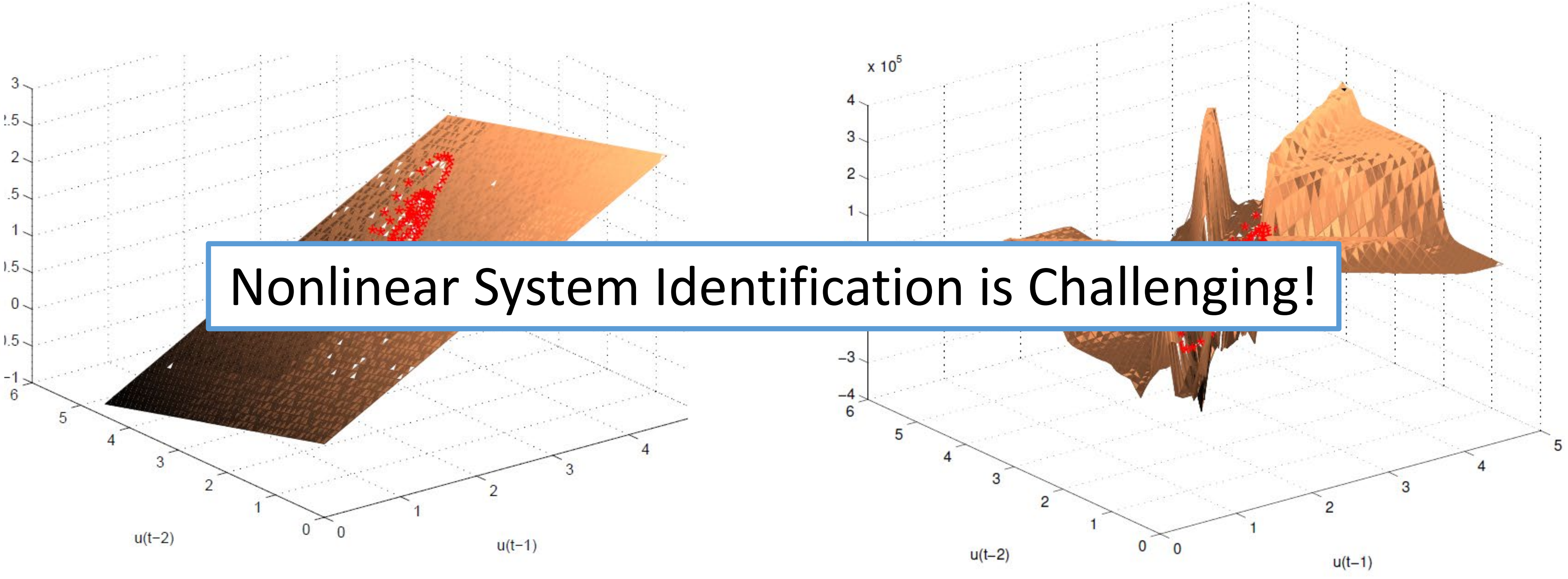
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Source: J. Schoukens and L. Ljung, Nonlinear System Identification A User-Oriented Roadmap, IEEE Control Systems Magazine, vol. 39, n. 6, Dec. 2019

Nonlinear System Identification

Linear Identification: Characterize a **hyperplane**

Nonlinear Identification: Characterize a **manifold**



(a) (b)

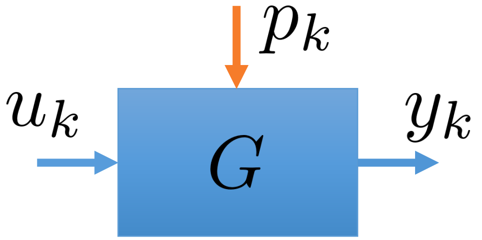
Nonlinear System Identification is Challenging!

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Source: J. Schoukens and L. Ljung, Nonlinear System Identification A User-Oriented Roadmap, IEEE Control Systems Magazine, vol. 39, n. 6, Dec. 2019

Nonlinear Models

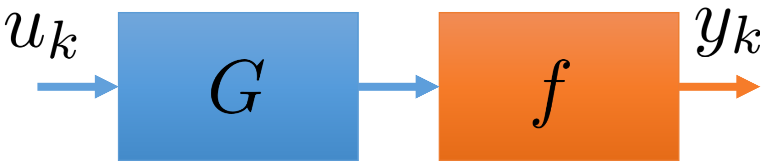
Linear Parameter-Varying



NARMAX

$$y_k = f(u_{k-1}, y_{k-1}, e_{k-1}) + e_k$$

Block-Oriented Models



Nonlinear State-Space

$$x_{k+1} = f(x_k, u_k, e_k)$$
$$y_k = g(x_k, u_k) + e_k$$

Nonlinear Models

Linear Parameter-Varying

NARMAX

Block-Oriented Models

Nonlinear State-Space

Scales well to MIMO

Compact representation of dynamics

Nonlinear State-Space Model

hidden states

$$x_{k+1} = f(x_k, u_k)$$

$$y_k = g(x_k, u_k)$$

output

inputs

Function Representation:

Neural Network

Polynomial

Local Linear Models

Probabilistic/Deterministic

Nonlinear State-Space Model

hidden states

$$x_{k+1} = f(x_k, u_k, e_k)$$
$$y_k = g(x_k, u_k) + e_k$$

output

inputs

noise

Function Representation:

- Neural Network
- Polynomial
- Local Linear Models
- Probabilistic/Deterministic

Noise Model:

- Output Error
- Innovation Form

Outline

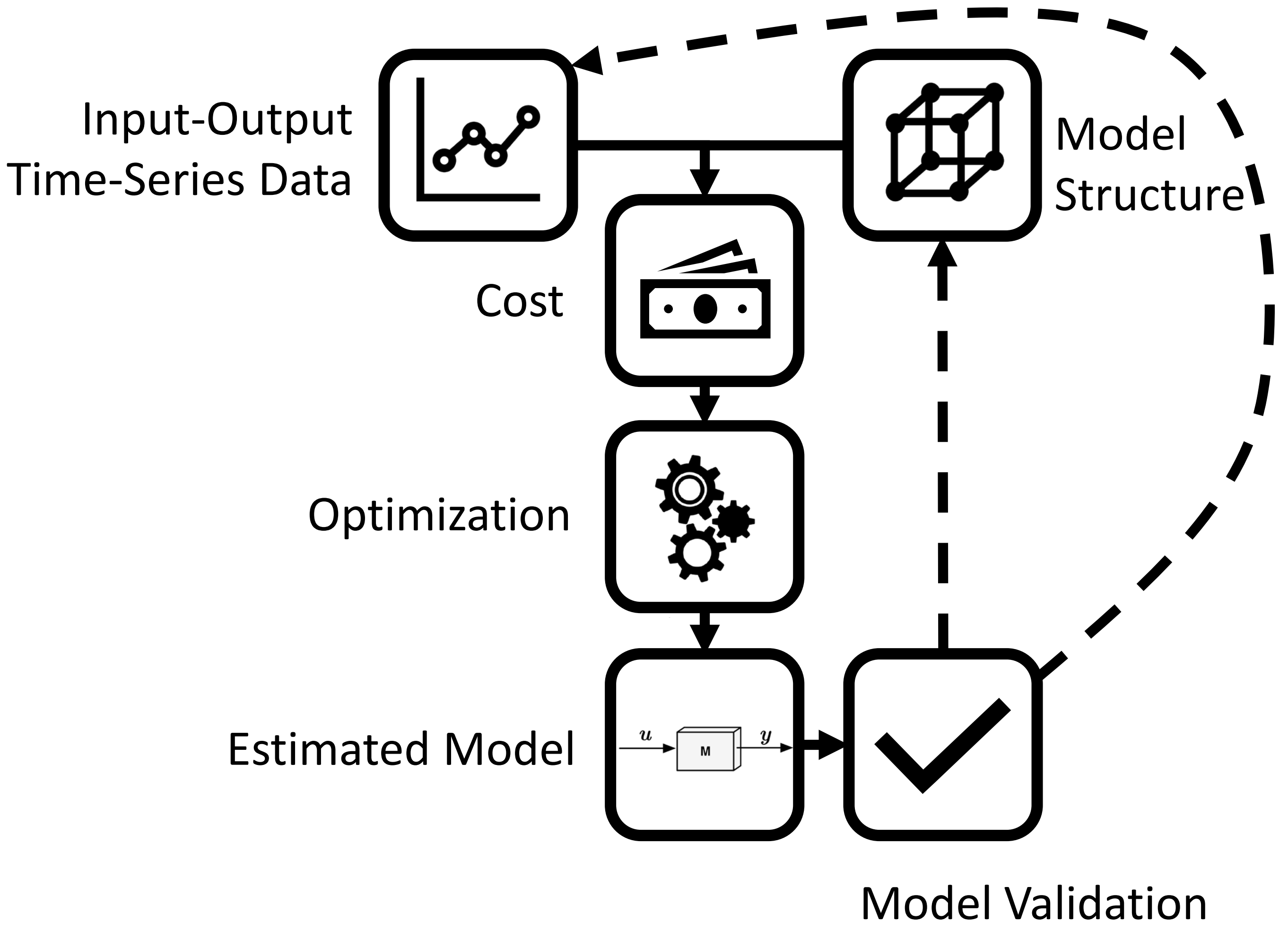
‘Classical’ Approach

Challenges

Deep Subspace Encoder

Examples

‘Classical’ Approach



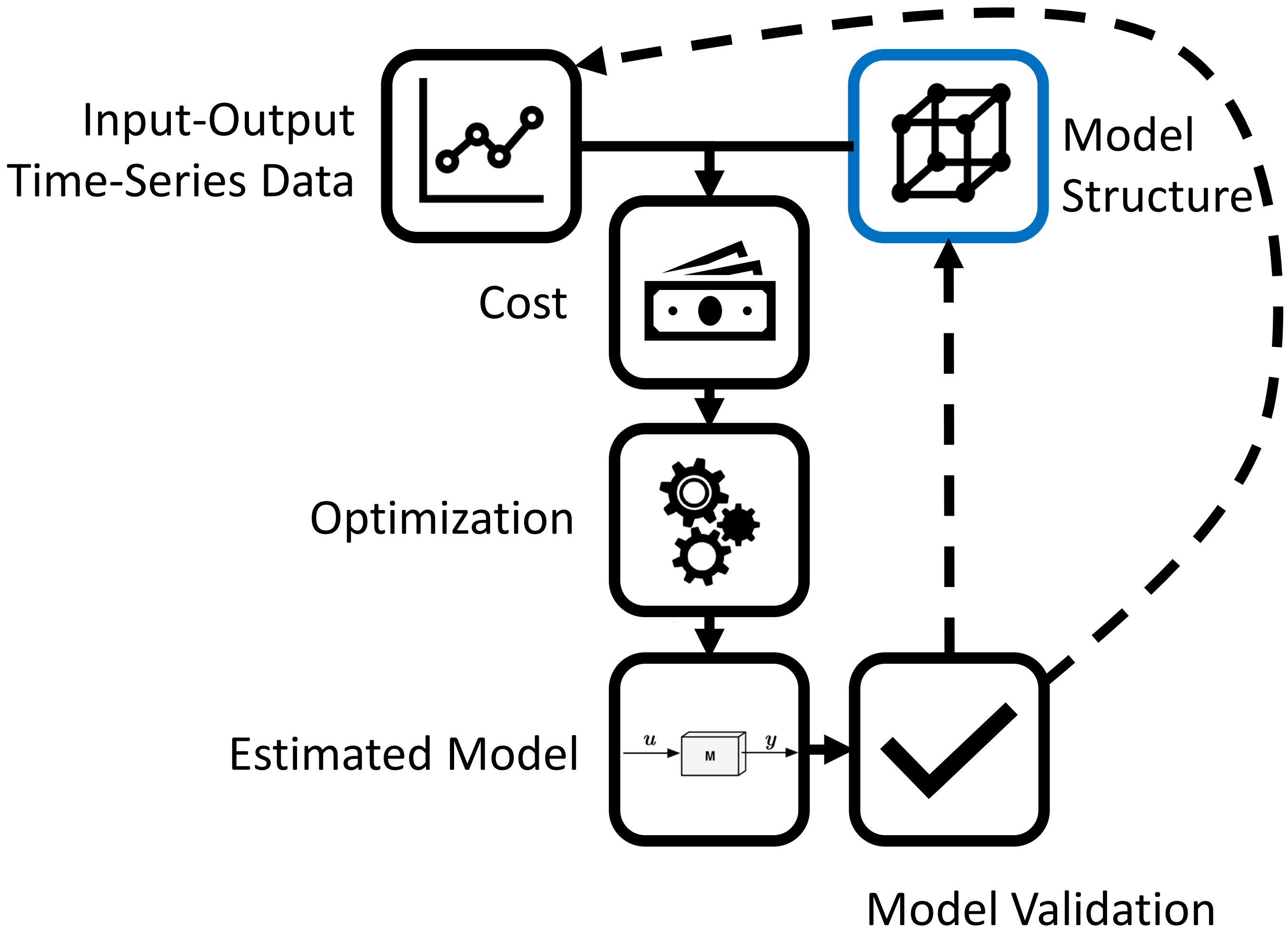
'Classical' Approach

Model Structure:

$$\hat{x}_{k+1} = f_\theta(\hat{x}_k, u_k)$$
$$\hat{y}_k = g_\theta(\hat{x}_k, u_k) + e_k$$

OE noise structure

Basis function nonlinearity representation



‘Classical’ Approach

Model Structure:

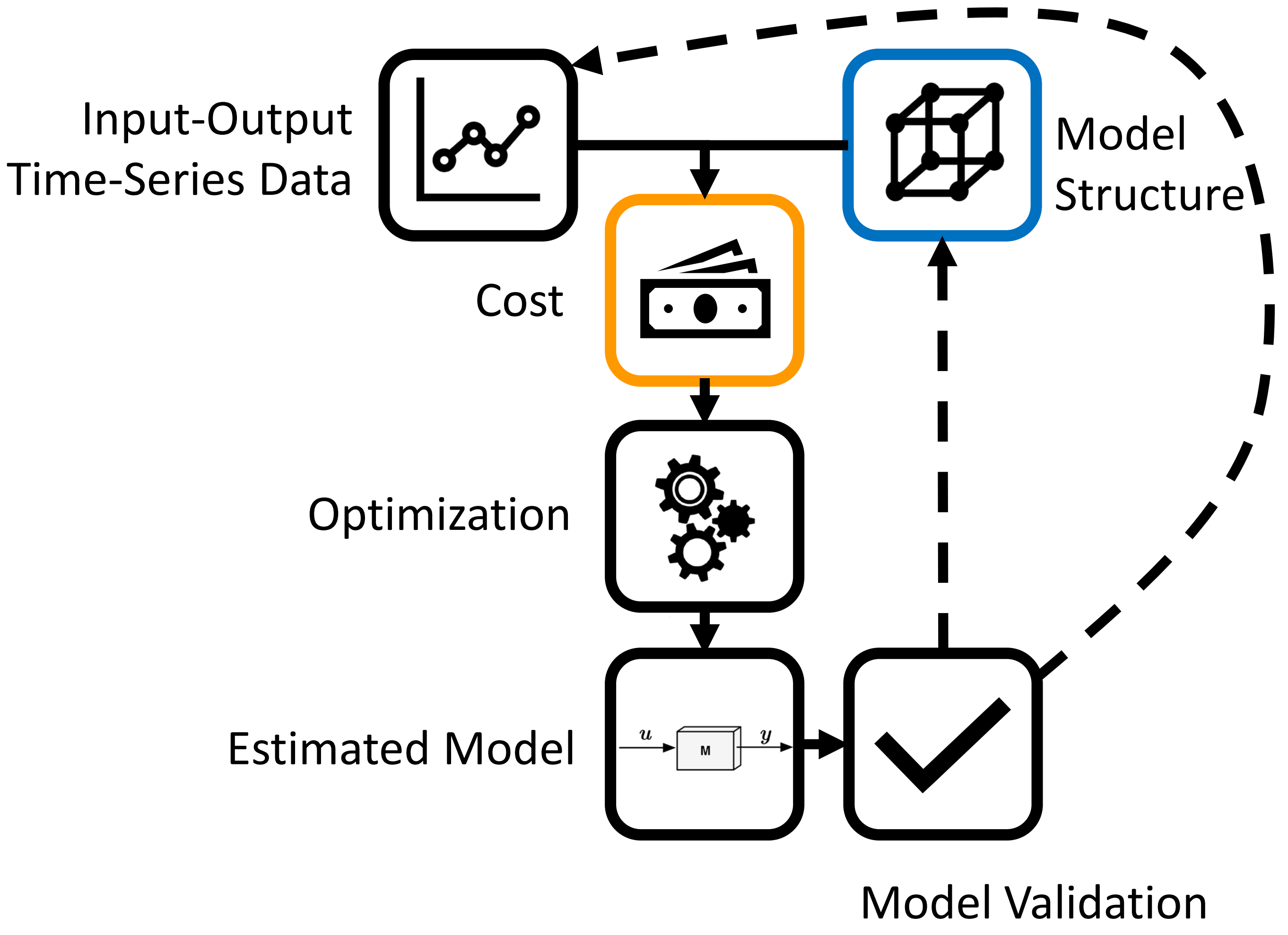
$$\hat{x}_{k+1} = f_\theta(\hat{x}_k, u_k)$$
$$\hat{y}_k = g_\theta(\hat{x}_k, u_k) + e_k$$

OE noise structure
Basis function nonlinearity representation

Cost Function:

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$$

Simulation error minimization



'Classical' Approach

Model Structure:

$$\hat{x}_{k+1} = f_\theta(\hat{x}_k, u_k)$$
$$\hat{y}_k = g_\theta(\hat{x}_k, u_k) + e_k$$

OE noise structure

Basis function nonlinearity representation

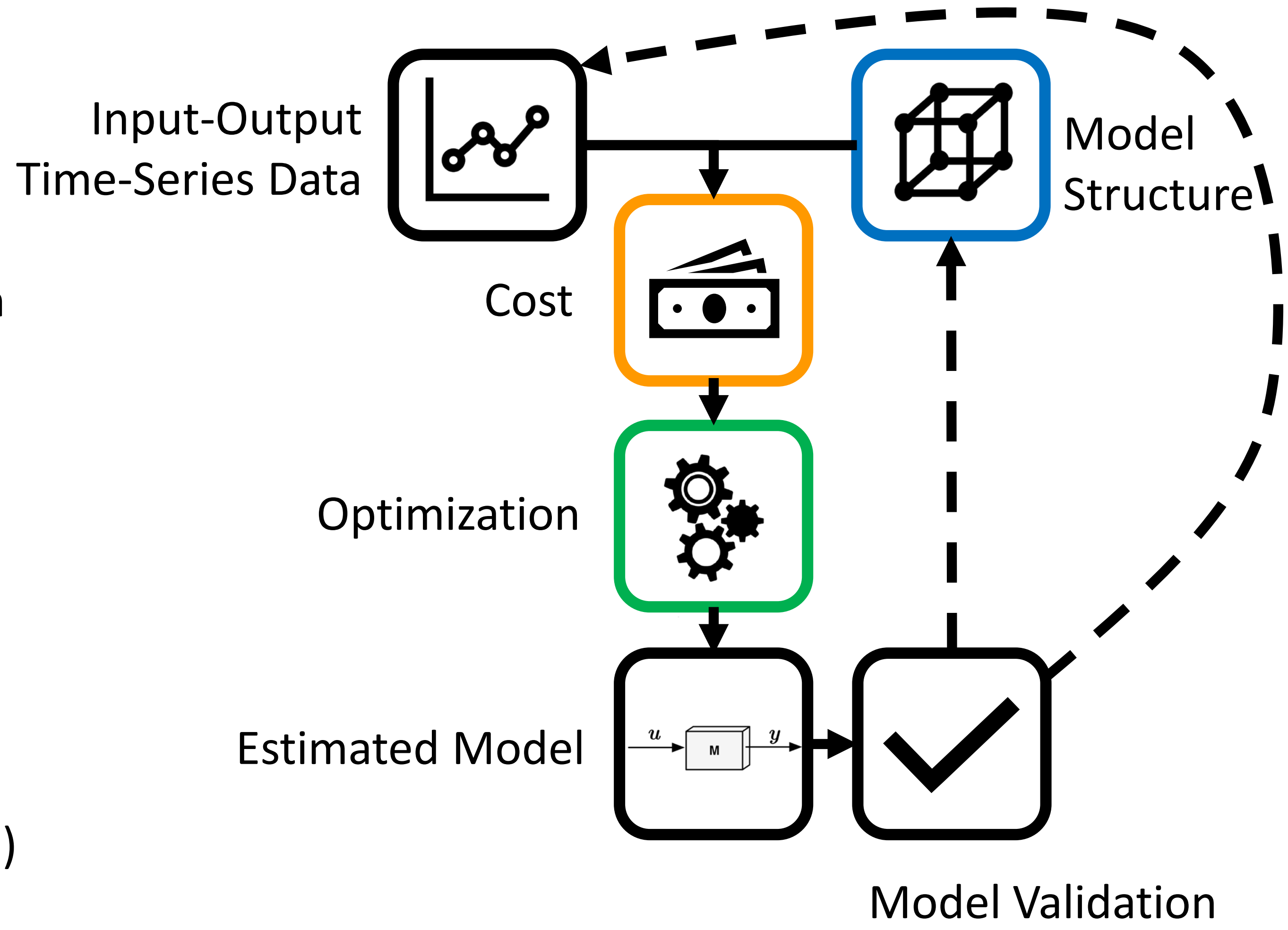
Cost Function:

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$$

Simulation error minimization

Optimization:

Gauss-Newton algorithms (e.g. Lev. Marq.)



Outline

‘Classical’ Approach

Challenges

Deep Subspace Encoder

Examples

Challenges

Large # parameters

$$x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k, u_k)$$

parameters grow with state, input and output dimension

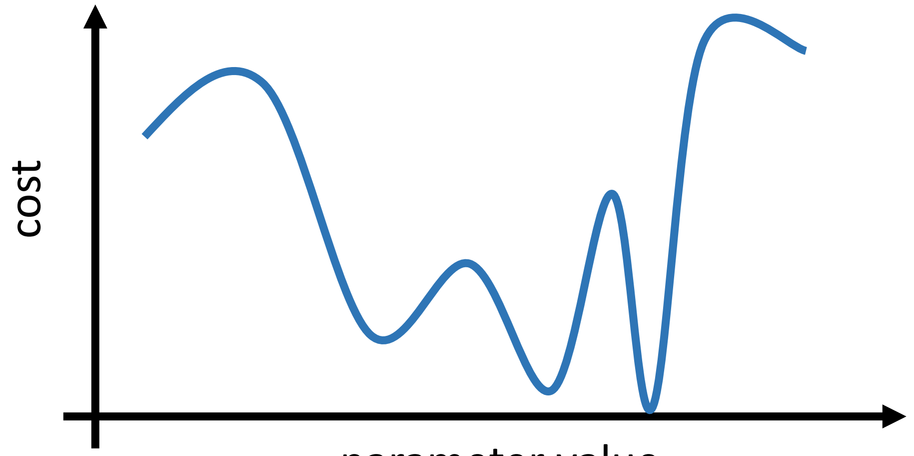
Challenges

Large # parameters

Local minima

$$x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k, u_k)$$

Problem is nonlinear in the parameters



Challenges

Large # parameters

Local minima

$$x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k, u_k)$$

Instabilities during training

Model simulation or gradient calculation can become unstable during optimization

Challenges

Large # parameters

Local minima

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$$

1. Minimize using Gauss-Newton like algorithm

→ Scales badly for growing data

Instabilities during training

Computational / memory cost

2. Simulate full data record for one optimization step

→ No room for parallelization due to initial state transients

Challenges

Large # parameters	efficient representations
Local minima	smart initialization
	cost smoothening
Instabilities during training	multiple shooting
Computational / memory cost	truncated simulation error
	stochastic optimization methods

Outline

‘Classical’ Approach

Challenges

Deep Subspace Encoder

Examples

Deep Subspace Encoder

Challenges:

Large # parameters

Local minima

Instabilities during training

Computational / memory cost

Solution:

Combining System Identification and Deep Learning!

Subspace Encoder

Large # parameters

Local minima

Instabilities during training

Computational / memory cost

efficient representations

smart initialization

cost smoothening

multiple shooting

truncated simulation error

stochastic optimization methods

State-Space Neural Networks

$$x_{k+1} = f\left(x_k, u_k\right)$$
$$y_k = g\left(x_k, u_k\right)$$

Representation *f* and *g* nonlinearity as a fully connected feedforward neural network (MLP)

Function approximation: integrated squared error [Baron, 1993]

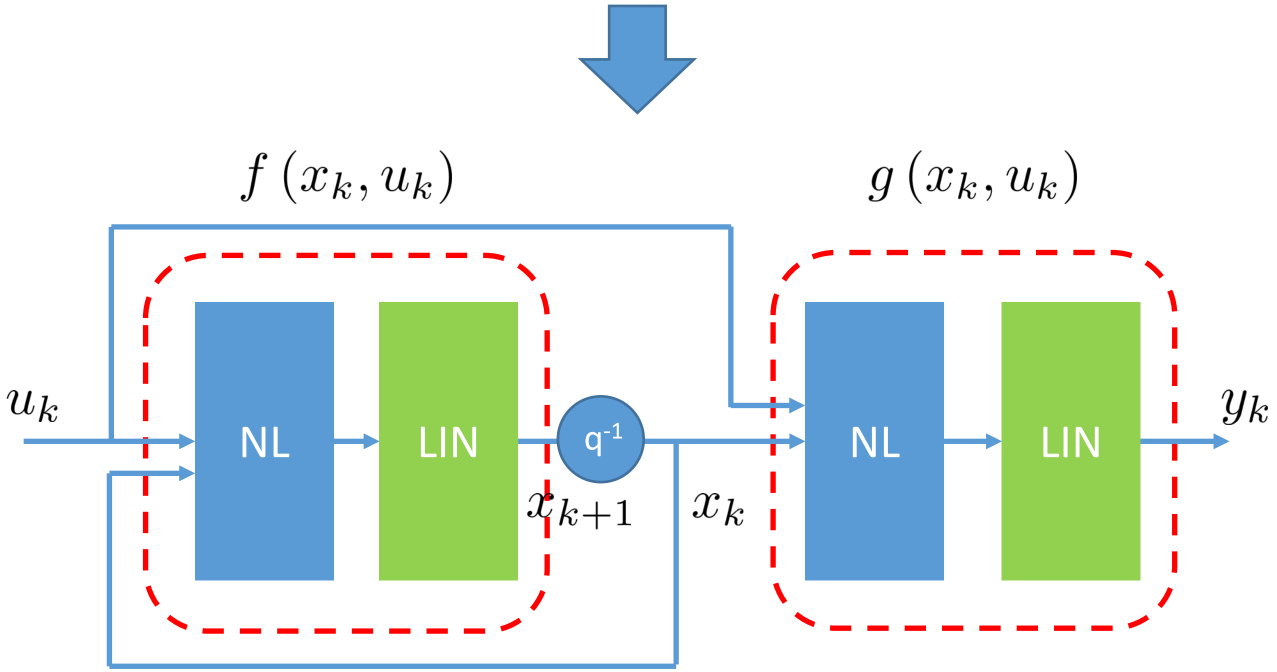
feedforward neural net $V = \mathcal{O}\left(\frac{1}{n}\right)$ → # hidden neurons

basis function expansion $V = \mathcal{O}\left(\frac{1}{n^{2/n_x}}\right)$ → # terms (n), # input variables (n_x)

State-Space Neural Networks

$$x_{k+1} = f\left(x_k, u_k\right)$$
$$y_k = g\left(x_k, u_k\right)$$

Representation f and g nonlinearity as a fully connected feedforward neural network (MLP)



J.A.K. Suykens et al., Nonlinear system identification using neural state space models, applicable to robust control design. International Journal of Control, 62(1):129–152, 1995

State-Space Neural Networks

$$x_{k+1} = f\left(x_k, u_k\right)$$
$$y_k = g\left(x_k, u_k\right)$$

Representation f and g nonlinearity as a fully connected feedforward neural network (MLP)

$$x_{k+1} = W_x \sigma \left(\begin{bmatrix} W_{fx} & W_{fu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + b_f \right) + b_x$$
$$y_k = W_y \sigma \left(\begin{bmatrix} W_{gx} & W_{gu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + b_g \right) + b_y$$

σ → activation function

W = weights, b = biases → model parameters

State-Space Neural Networks

$$x_{k+1} = Ax_k + Bu_k + f\left(x_k, u_k\right)$$
$$y_k = Cx_k + Du_k + g\left(x_k, u_k\right)$$

Representation *f* and *g* nonlinearity as a fully connected feedforward neural network (MLP)

Include an explicit linear term

Similar to residual neural networks
Also present in PNLSS

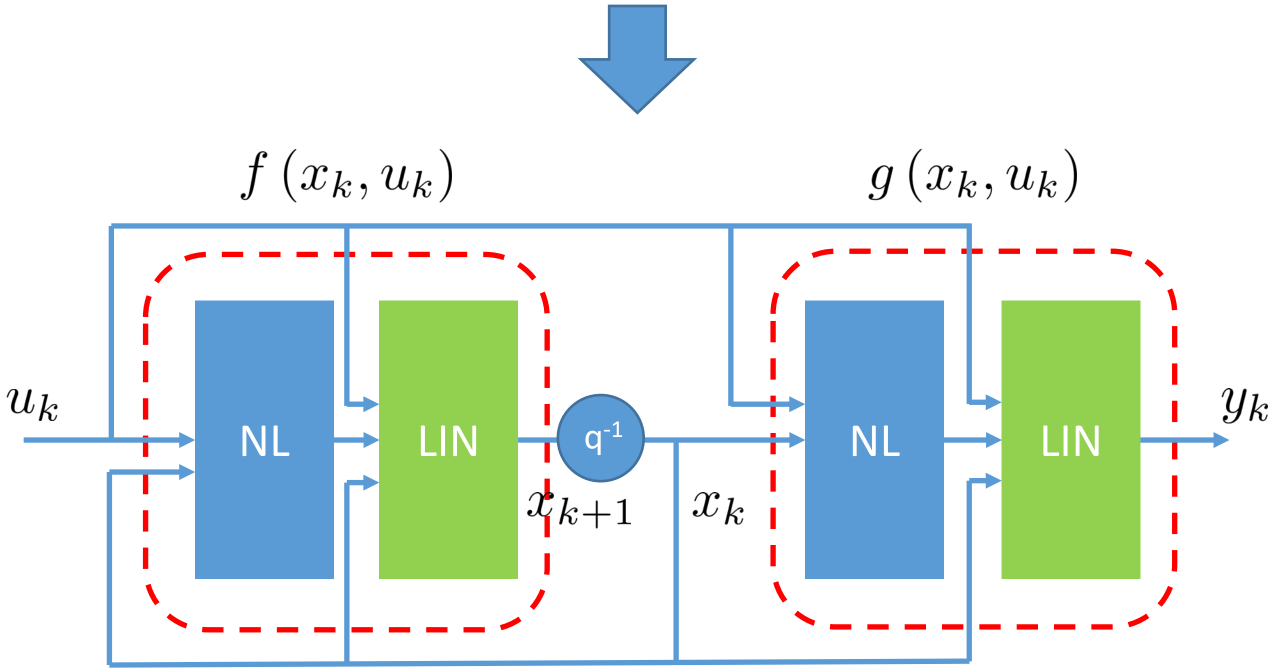
State-Space Neural Networks

$$x_{k+1} = Ax_k + Bu_k + f\left(x_k, u_k\right)$$
$$y_k = Cx_k + Du_k + g\left(x_k, u_k\right)$$

Representation f and g nonlinearity as a fully connected feedforward neural network (MLP)

Include an explicit linear term

Similar to residual neural networks
Also present in PNLSS



TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Maarten Schoukens, Improved Initialization of State-Space Artificial Neural Networks, European Control Conference, Rotterdam, 2021

Koen Tiels, PNLSS 1.0: A polynomial nonlinear state-space toolbox for Matlab, http://homepages.vub.ac.be/~jschouk, 2016

State-Space Neural Networks

$$x_{k+1} = Ax_k + Bu_k + f\left(x_k, u_k\right)$$
$$y_k = Cx_k + Du_k + g\left(x_k, u_k\right)$$

$$x_{k+1} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_x \sigma\left(\begin{bmatrix} \tilde{W}_{fx} & \tilde{W}_{fu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_f\right) + \tilde{b}_x$$
$$y_k = \begin{bmatrix} C & D \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_y \sigma\left(\begin{bmatrix} \tilde{W}_{gx} & \tilde{W}_{gu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_g\right) + \tilde{b}_y$$

Representation *f* and *g* nonlinearity as a fully connected feedforward neural network (MLP)

Include an explicit linear term

Similar to residual neural networks
Also present in PNLSS

Subspace Encoder

Large # parameters — efficient representations

Local minima — **smart initialization**

cost smoothening

Instabilities during training — multiple shooting

Computational / memory cost — truncated simulation error

stochastic optimization methods

Parameter Initialization

$$
\begin{aligned}
x_{k+1} &= Ax_k + Bu_k + f\left(x_k, u_k\right) \\
y_k &= Cx_k + Du_k + g\left(x_k, u_k\right)
\end{aligned}
$$

Parameter Initialization

$$x_{k+1} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_x \sigma\left(\begin{bmatrix} \tilde{W}_{fx} & \tilde{W}_{fu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_f \right) + \tilde{b}_x$$

$$y_k = \begin{bmatrix} C & D \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_y \sigma\left(\begin{bmatrix} \tilde{W}_{gx} & \tilde{W}_{gu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_g \right) + \tilde{b}_y$$

Parameter Initialization

$$x_{k+1} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_x \sigma \left(\begin{bmatrix} \tilde{W}_{fx} & \tilde{W}_{fu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_f \right) + \tilde{b}_x$$

$$y_k = \begin{bmatrix} C & D \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_y \sigma \left(\begin{bmatrix} \tilde{W}_{gx} & \tilde{W}_{gu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_g \right) + \tilde{b}_y$$

Random + LTI-approximation initialization

1. Use LTI sysid methods to obtain ABCD matrices (Best Linear Approximation)
2. Embed **LTI approximation** in NL state-space model
3. Initialize remainder with **Random** or **zero** values
4. Initial SSNN performance = LTI approximation

Subspace Encoder

Large # parameters

efficient representations

Local minima

smart initialization

cost smoothening

Instabilities during training

multiple shooting

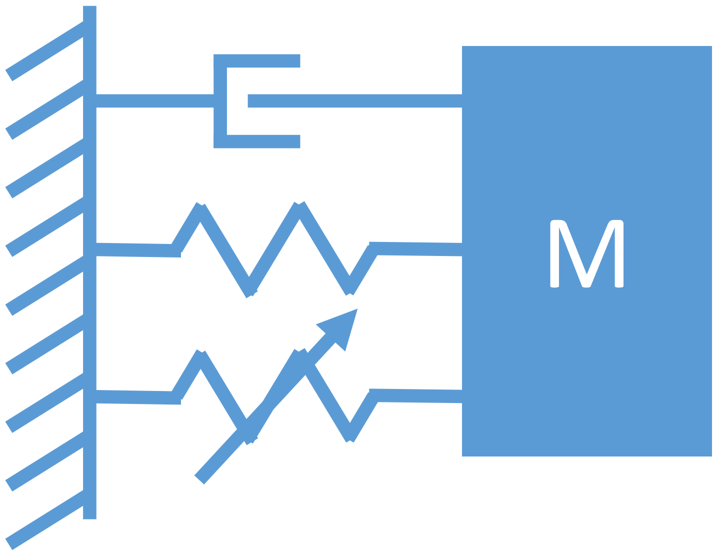
Computational / memory cost

truncated simulation error

stochastic optimization methods

Intermezzo: Bouc-Wen Benchmark

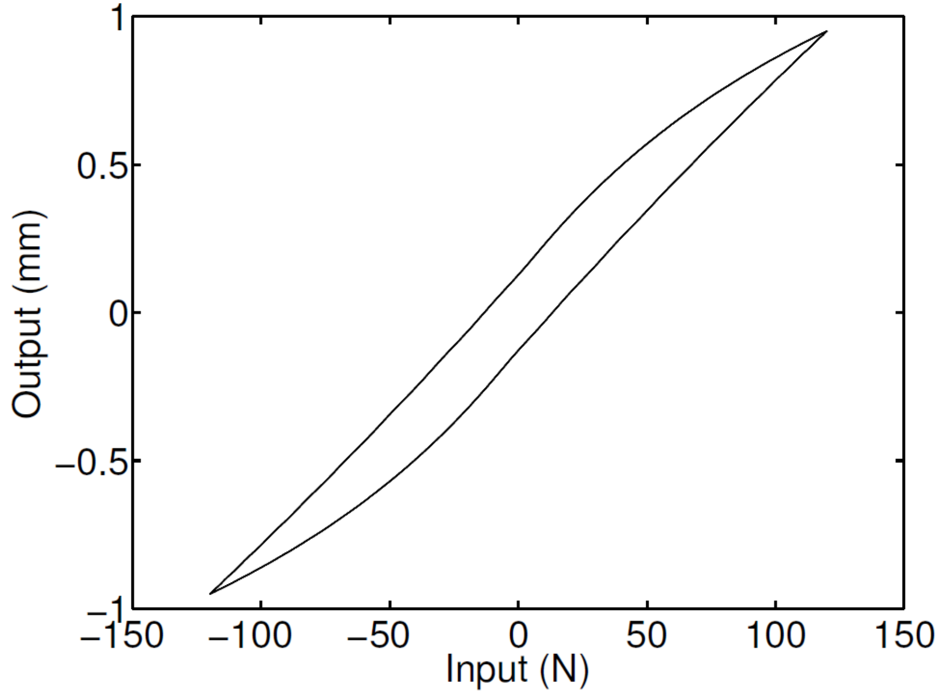
Intermezzo: Bouc-Wen Benchmark



$$m\ddot{y}_t + c\dot{y}_t + ky_t + z(y_t, \dot{y}_t) = u_t$$

$$\dot{z}_t = \alpha\dot{y}_t - \beta\left(\gamma\left|\dot{y}_t\right| z_t + \delta\dot{y}_t\left|z_t\right|\right)$$

Hysteretic Loop

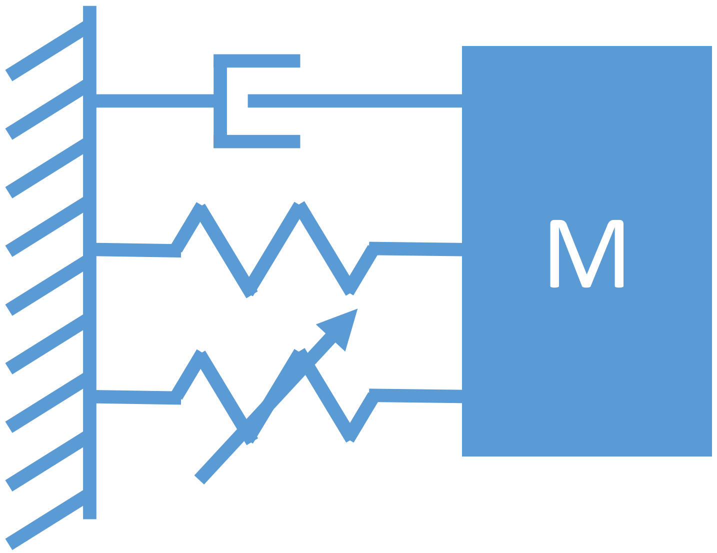


8192 samples training & test
Random phase multisine input (5-150 Hz)

Benchmark system available on:
nonlinearbenchmark.org

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Intermezzo: Bouc-Wen Benchmark



$$m\ddot{y}_t + c\dot{y}_t + ky_t + z(y_t, \dot{y}_t) = u_t$$

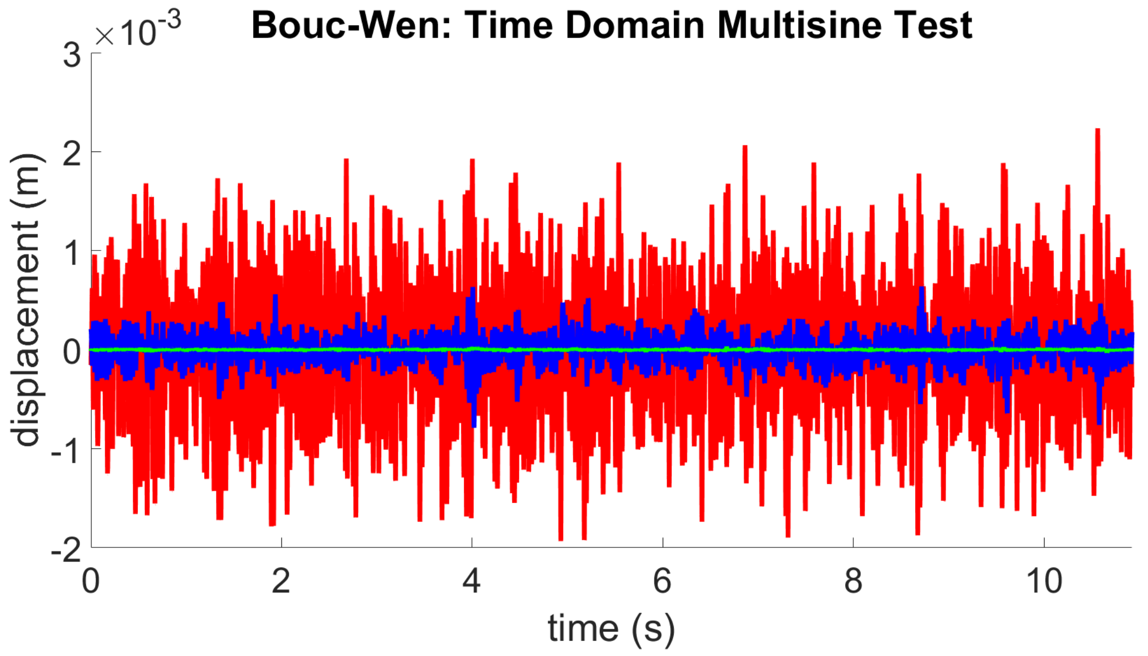
$$\dot{z}_t = \alpha\dot{y}_t - \beta\left(\gamma\left|\dot{y}_t\right| z_t + \delta\dot{y}_t\left|z_t\right|\right)$$

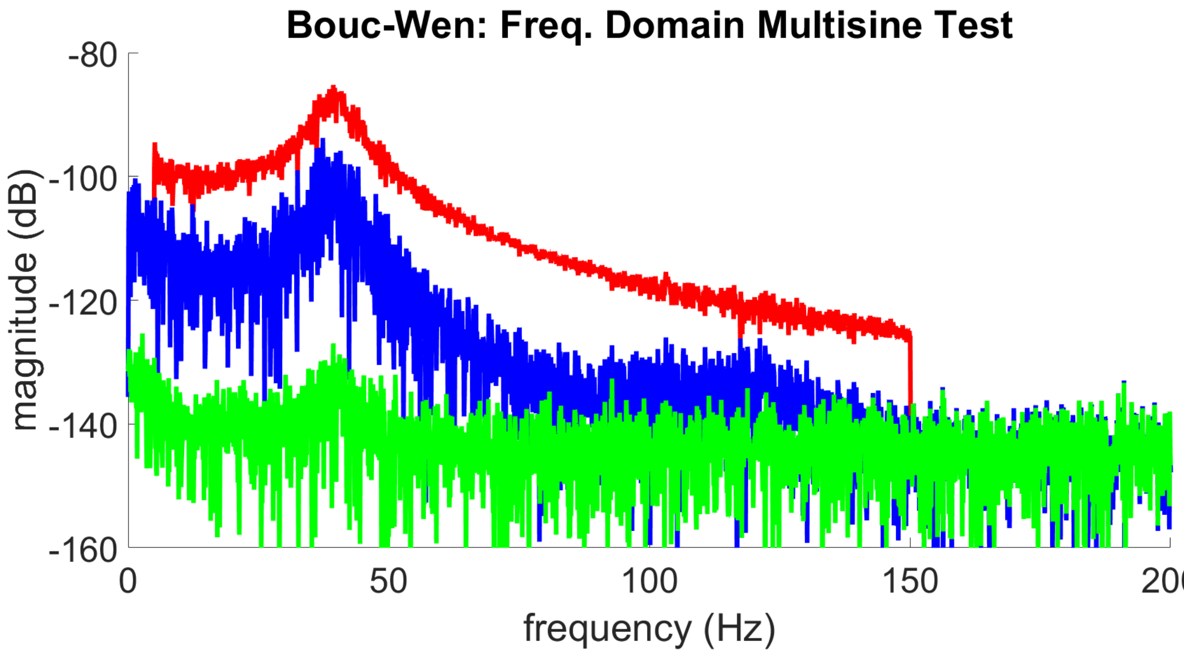
n_x = 3, 15 neurons, tanh activation
Levenberg-Marquardt optimization (Matlab)

LTI approximation using
Matlab ‘ssest’ command

Monte-Carlo simulation: 100 runs

Intermezzo: Bouc-Wen Benchmark





RMSE SS-NN (LTI init.): $7.14 \; 10^{-6}$

→ state-of-the-art result

system output

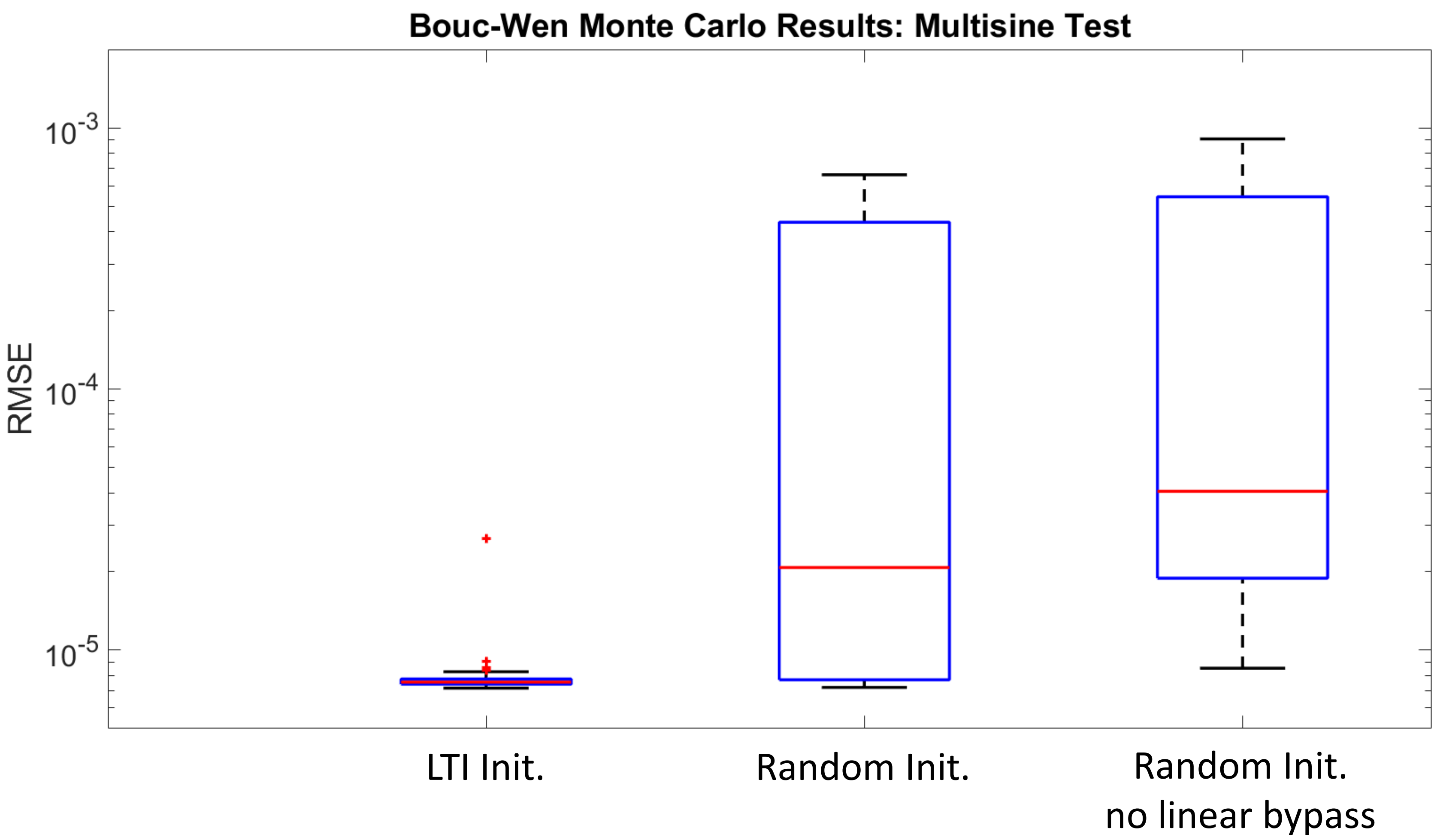
LTI error

gR-SS-NN error (LTI init.)

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Maarten Schoukens, Improved Initialization of State-Space Artificial Neural Networks, European Control Conference, Rotterdam, 2021

Intermezzo: Bouc-Wen Benchmark



Maarten Schoukens, Improved Initialization of State-Space Artificial Neural Networks, European Control Conference, Rotterdam, 2021

Subspace Encoder

Large # parameters

Local minima

Instabilities during training

Computational / memory cost

efficient representations

smart initialization

cost smoothening

multiple shooting

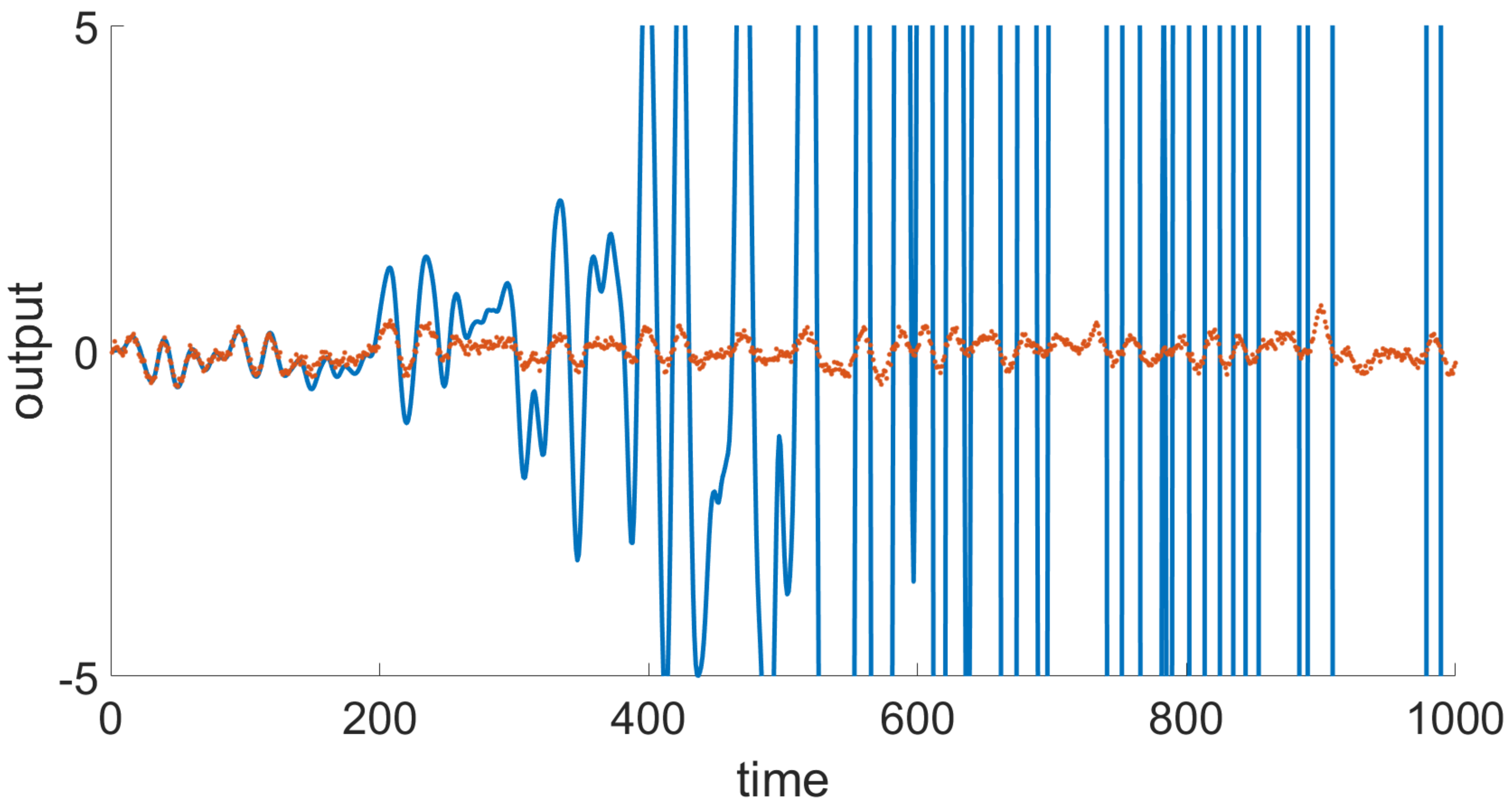
truncated simulation error

stochastic optimization methods

Multiple Shooting

Problem: training becomes unstable

Idea: Break the dataset and restart simulation (zero or estimated init. state)[1]



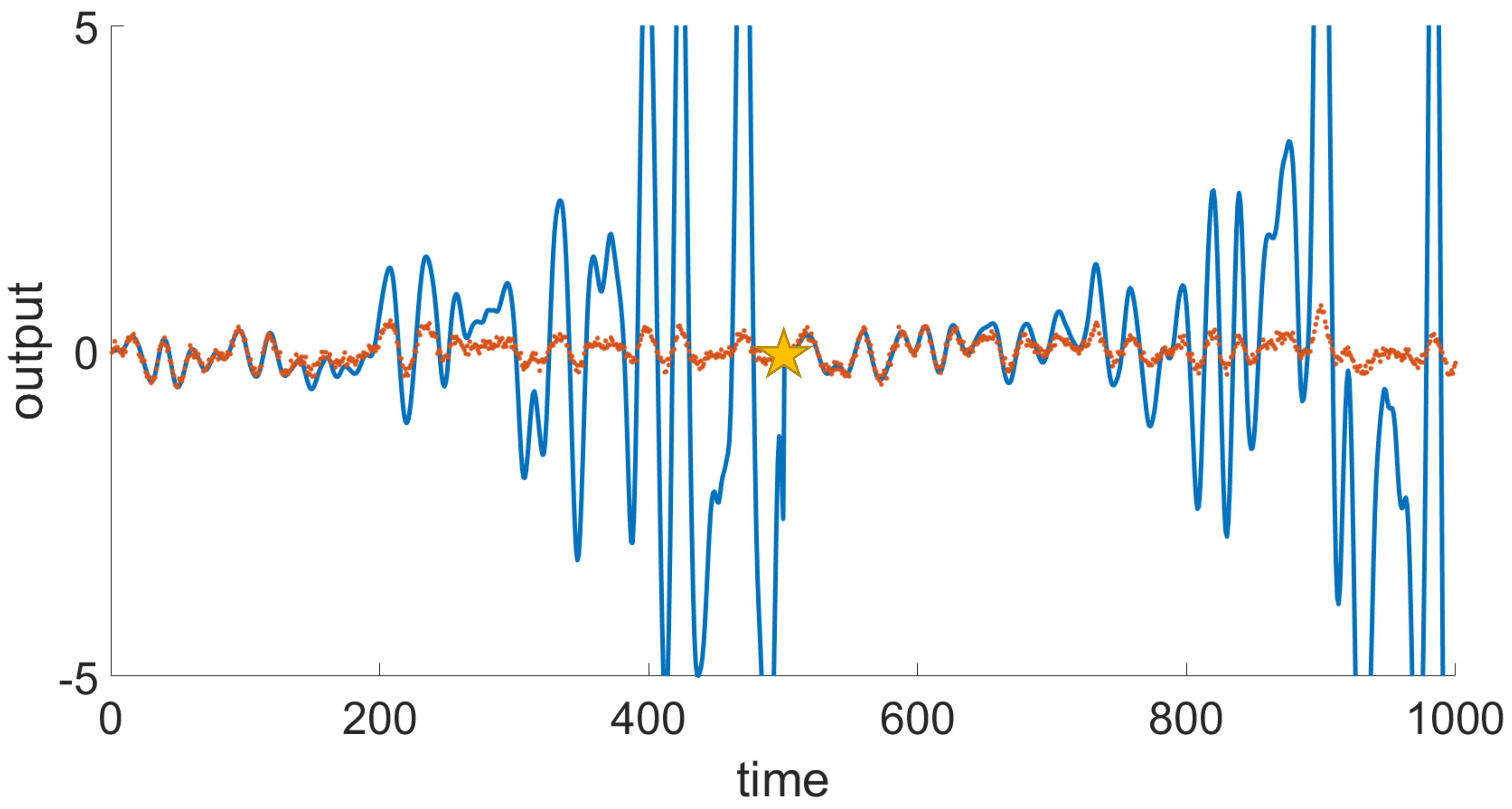
1. J. Decuyper et al., Tuning nonlinear state-space models using unconstrained multiple shooting, IFAC-PapersOnLine, vol. 53, n. 2, pp. 334-340, 2020

Multiple Shooting

Problem: training becomes unstable

Idea: Break the dataset and restart simulation (zero or estimated init. state)

Improves optimization stability and cost function smoothness[1]

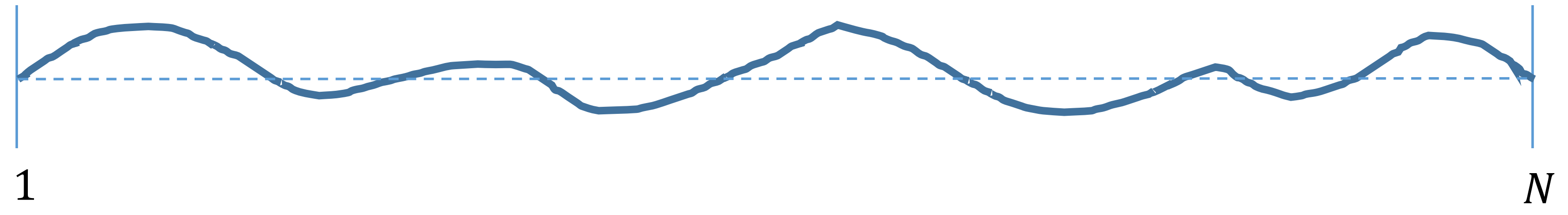


1. Antonio H. Ribeiro et al., On the smoothness of nonlinear system identification, Automatica, vol. 121, n. 109158, Nov. 2020

Subspace Encoder

Large # parameters	efficient representations
Local minima	smart initialization
	cost smoothening
Instabilities during training	multiple shooting
Computational / memory cost	**truncated simulation error**
	stochastic optimization methods

Truncated Simulation Error

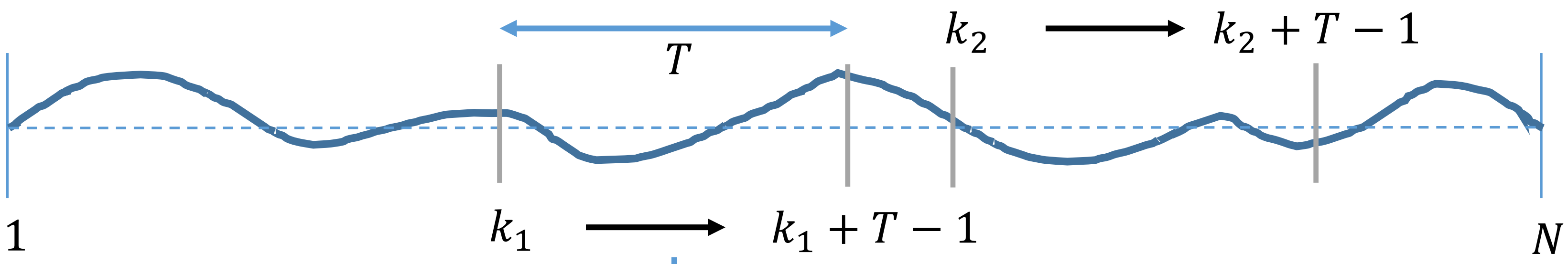


Classical Loss: $$V(\theta) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2$$

simulated model output

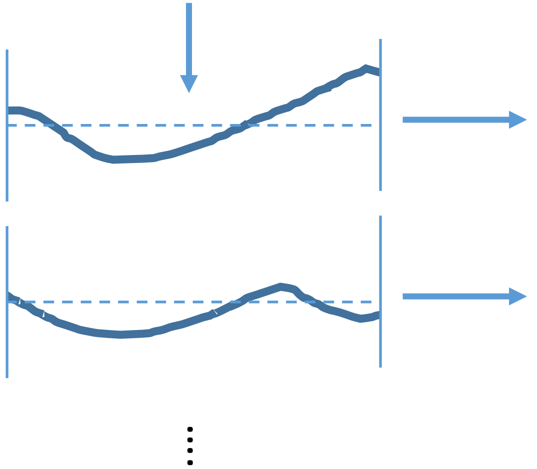
Marco Forgione et al., Continuous-time system identification with neural networks: Model structures and fitting criteria, European Journal of Control, vol. 59, pp. 69-81, May 2021

Truncated Simulation Error



Simulation Loss on section k_1

Simulation Loss on section k_2



$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\hat{y}_{k_1+t|k_1} - y_{k_1+t}\right)^2$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\hat{y}_{k_2+t|k_2} - y_{k_2+t}\right)^2$$

$\vdots$

\- - - - - - - - +

$$V(\theta) = \frac{1}{N_K T}\sum_{k\in K}\sum_{t=0}^{T-1}\left(\hat{y}_{k+t|k} - y_{k+t}\right)^2$$

Marco Forgione et al., Continuous-time system identification with neural networks: Model structures and fitting criteria, European Journal of Control, vol. 59, pp. 69-81, May 2021

Truncated Simulation Error

$$V(\theta) = \frac{1}{N_K T} \sum_{k \in K} \sum_{t=0}^{T-1} \left(\hat{y}_{k+t|k} - y_{k+t}\right)^2$$

Computational cost is $O(T < N)$ with parallelization

Gradient explosion controlled with cutoff

Smoothened cost function

Sections overlap: higher data efficiency

Batch / Stochastic gradient descent possible: efficient memory use

Subspace Encoder

How to bring all these elements together?

Efficient Representation

Stochastic Optimization Methods

Smart Initialization

Multiple Shooting

Truncated Simulation Error

Cost Smoothening

1. State-space neural network model
2. Unroll the state-space equation
3. Estimate the initial state using an encoder function
4. Truncated simulation error cost
5. Mini-batch optimization

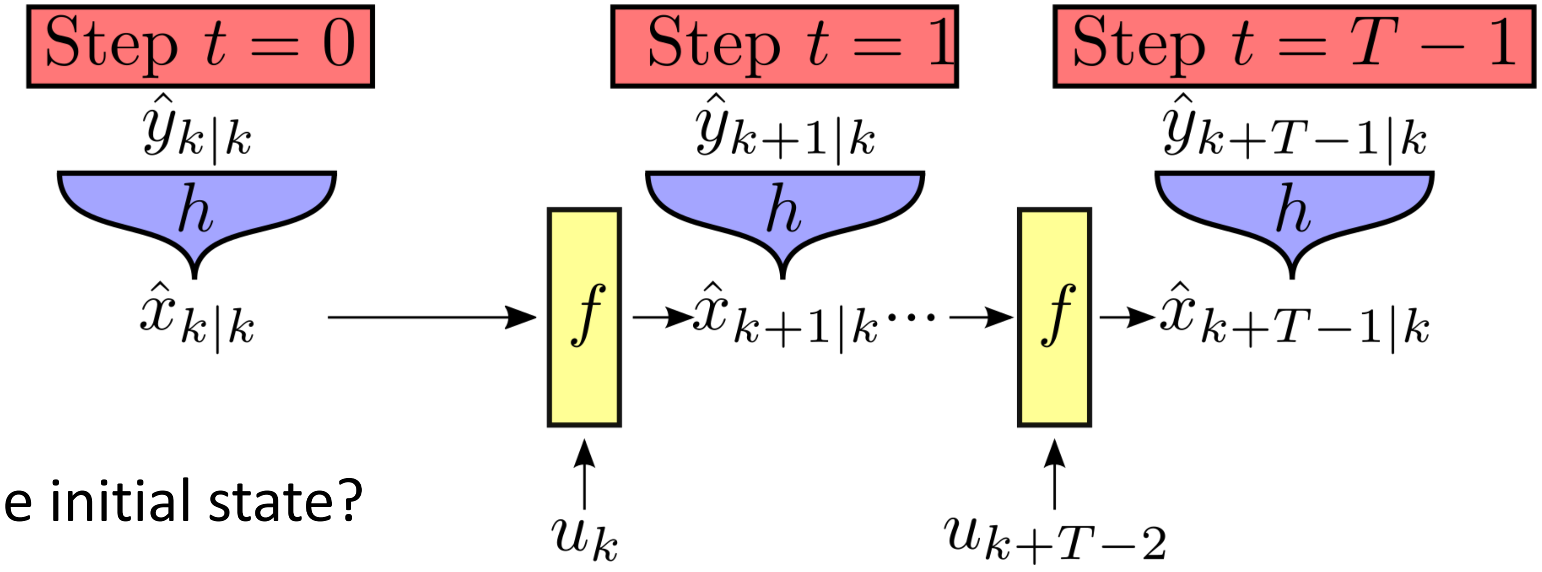
State-Space Neural Network

$$x_{k+1} = f\left(x_k, u_k\right)$$
$$y_k = g\left(x_k, u_k\right)$$

$$x_{k+1} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_x \sigma \left(\begin{bmatrix} \tilde{W}_{fx} & \tilde{W}_{fu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_f \right) + \tilde{b}_x$$
$$y_k = \begin{bmatrix} C & D \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{W}_y \sigma \left(\begin{bmatrix} \tilde{W}_{gx} & \tilde{W}_{gu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \tilde{b}_g \right) + \tilde{b}_y$$

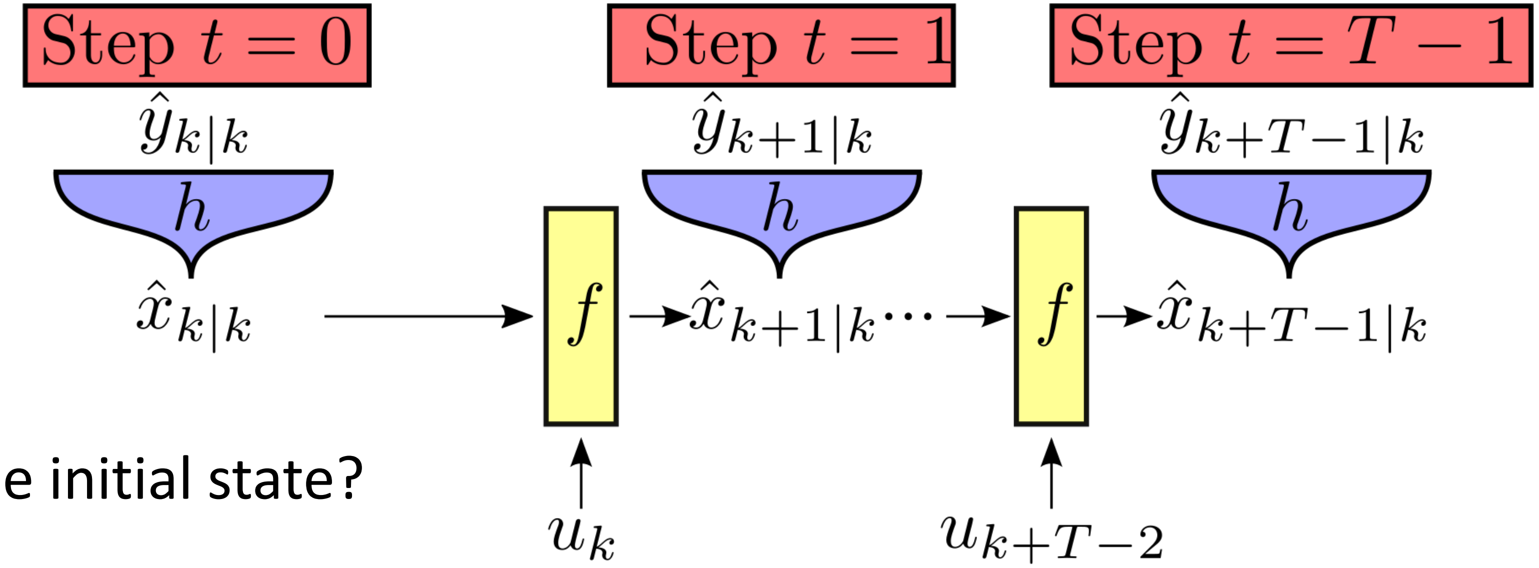
Unrolling the State-Space Equation

$$x_{k+1} = f\left(x_k, u_k\right)$$
$$y_k = g\left(x_k, u_k\right)$$



How to retrieve the initial state?

Initial State



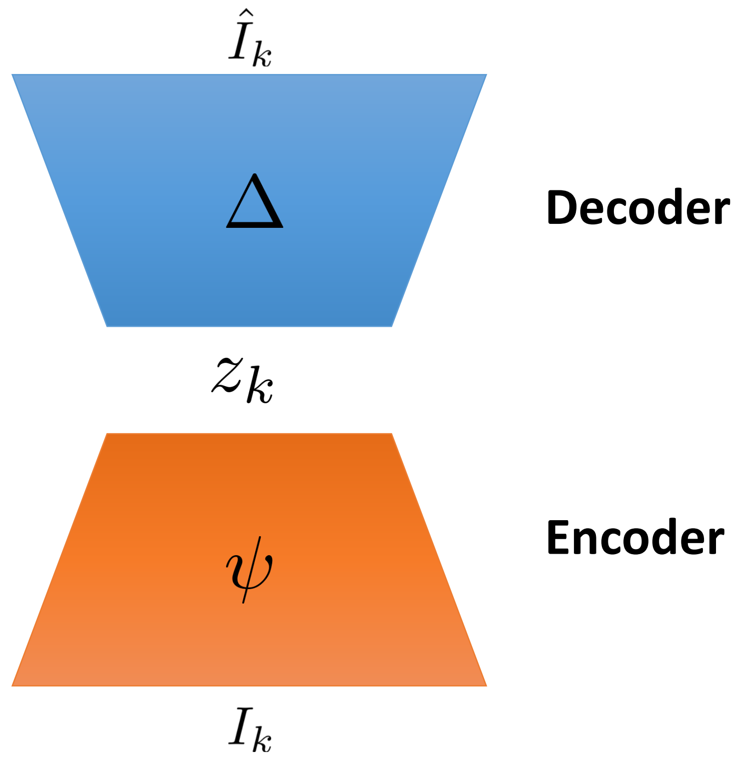
How to retrieve the initial state?

Zero initial state → transient errors

Estimate initial state directly → growing parameter vector

Auto-encoder initial state estimator

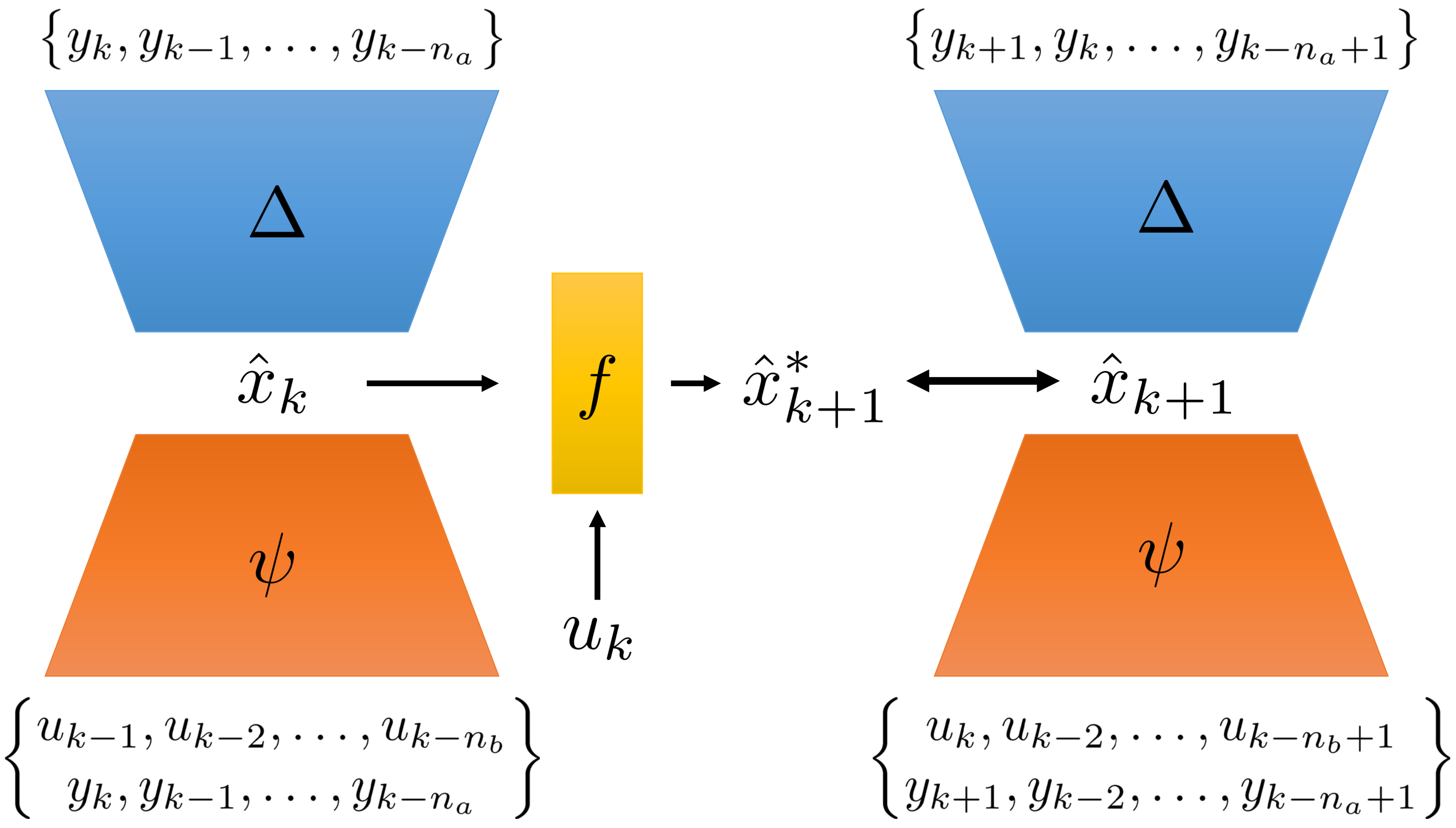
Initial State: Auto-Encoder



Objective:

Learn a low-dimensional representation of high-dimensional data

Initial State: Auto-Encoder



Challenges:

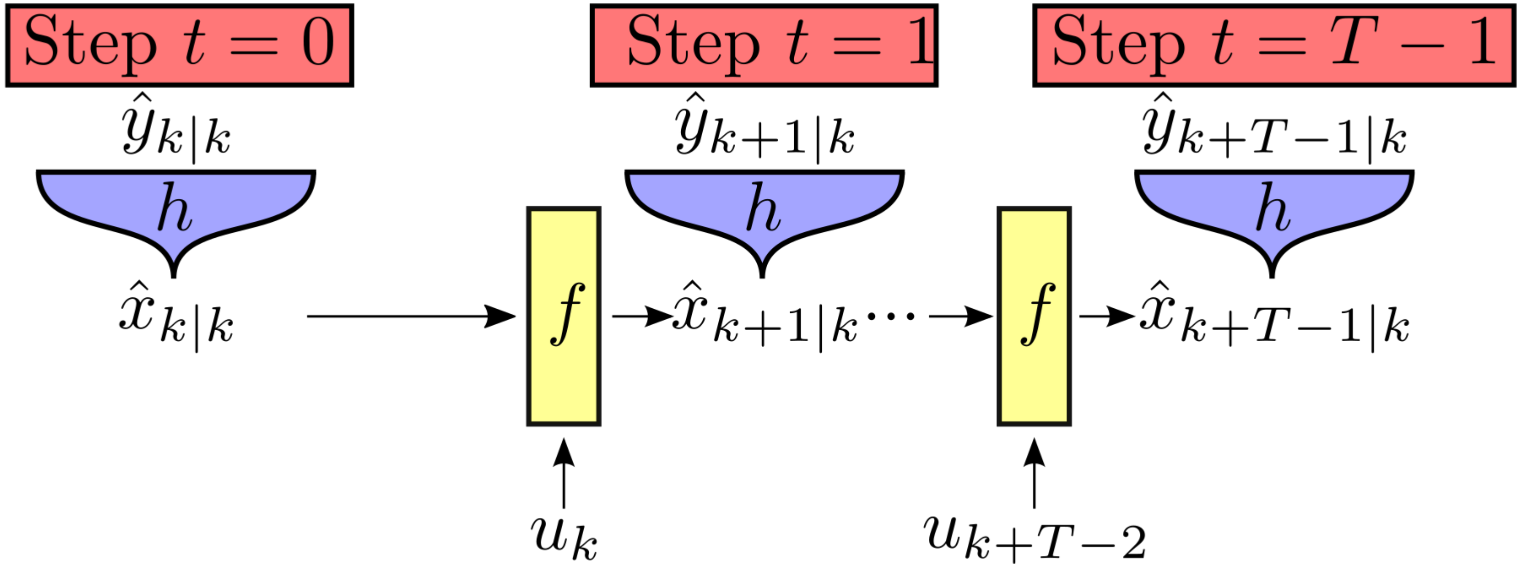
Requires multiple cost functions

States are not estimated for their predictive value

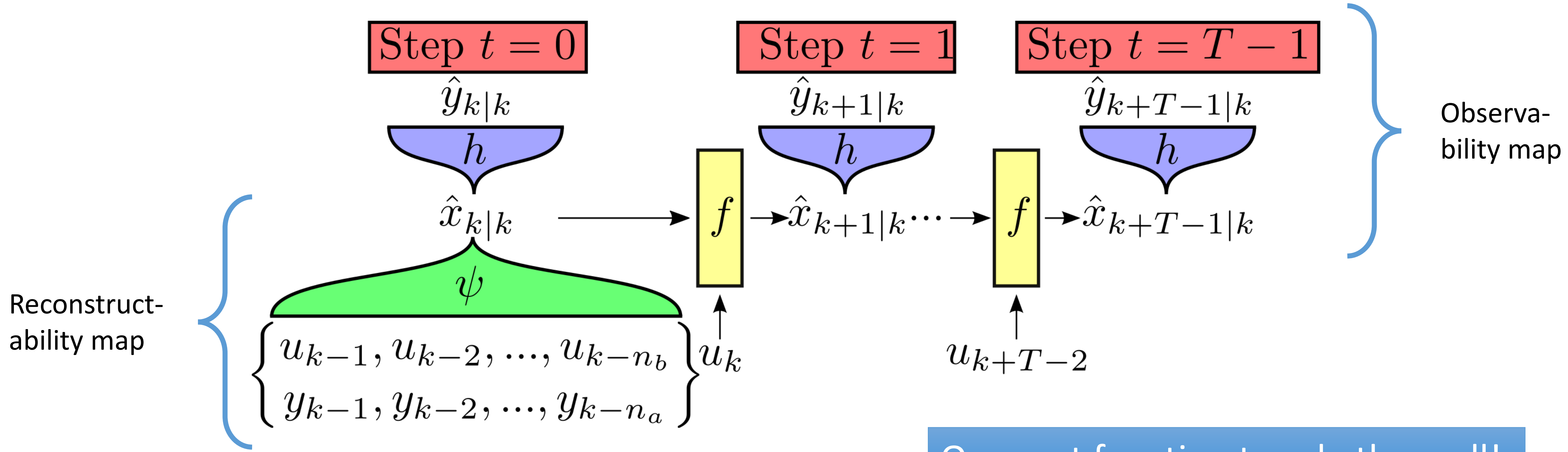
Integrate encoder in the main estimation problem!

Daniele Masti et al., Learning nonlinear state–space models using autoencoders, Automatica, vol. 129, n. 109666, May. 2021

Subspace Encoder



Subspace Encoder

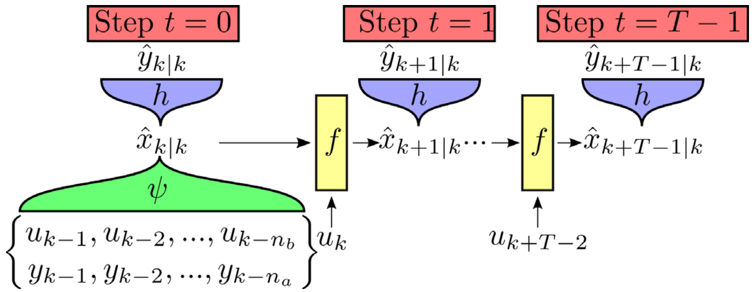


One cost function to rule them all!

$$Loss(\theta) = \sum_{k \in K} \sum_{t=0}^{T-1} \left(\hat{y}_{k+t|k} - y_{k+t}\right)^2$$

Vanilla Subspace Encoder: Overview

Model Structure:



OE Noise structure

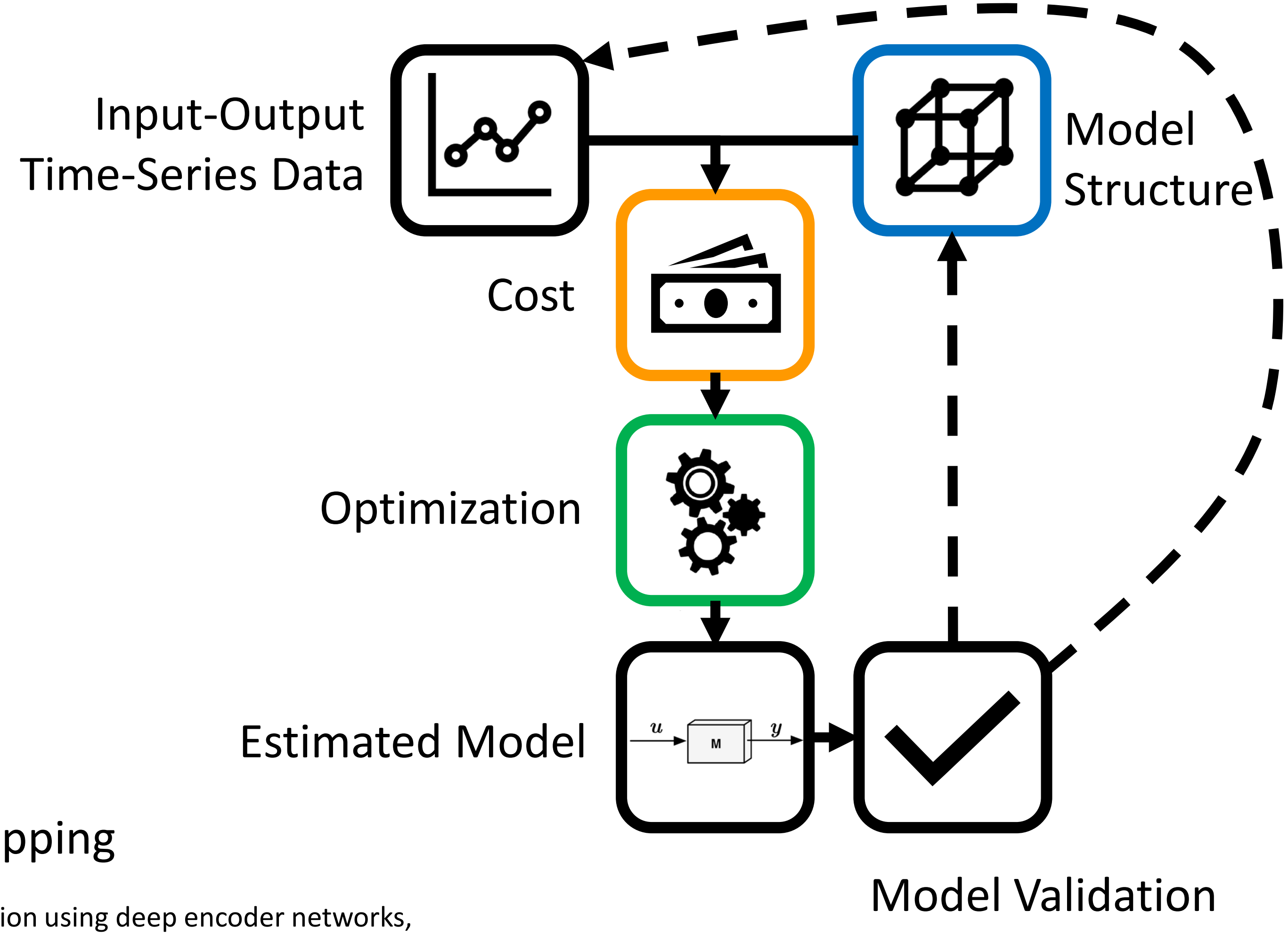
Cost Function:

$$Loss(\theta) = \sum_{k \in K} \sum_{t=0}^{T-1} \left(\hat{y}_{k+t|k} - y_{k+t}\right)^2$$

Truncated simulation error minimization

Optimization:

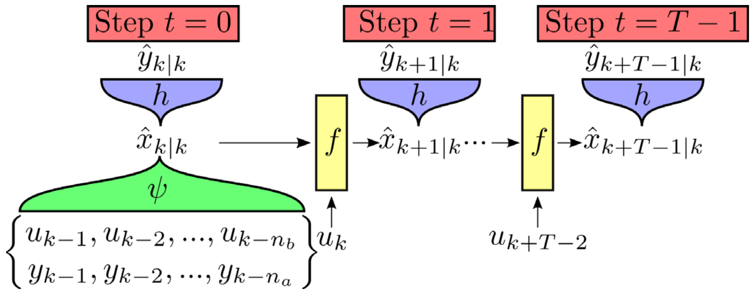
Minibatch gradient descent with early stopping



G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

Vanilla Subspace Encoder: Overview

Model Structure:



OE Noise structure

Cost Function:

$$Loss(\theta) = \sum_{k \in K} \sum_{t=0}^{T-1} \left(\hat{y}_{k+t|k} - y_{k+t}\right)^2$$

Truncated simulation error minimization

Optimization:

Minibatch gradient descent with early stopping

Advantages:

Efficient nonlinearity representation

Good scaling towards MIMO

Cost function smoothness

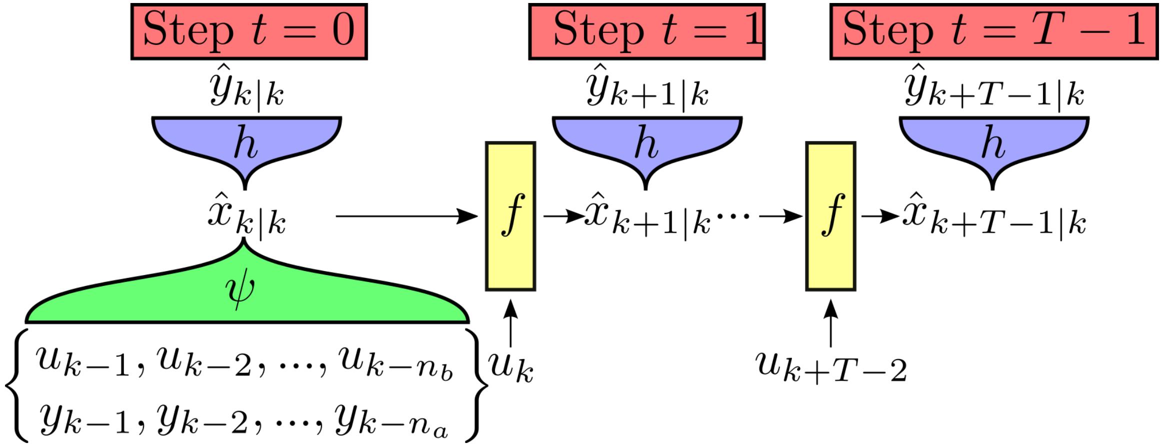
Good results starting from random init.

Good scaling towards large datasets

Consistency

G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

Subspace Encoder: Extensions



Implemented extensions:

From OE to Innovation noise

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k, u_k) + e_k \end{aligned} \quad \Rightarrow \quad \begin{aligned} x_{k+1} &= f(x_k, u_k, e_k) \\ y_k &= g(x_k, u_k) + e_k \end{aligned}$$

From DT to CT

From time-series to video sequences / spatiotemporal data

From nonlinear to Koopman

G.I. Beintema et al., Non-linear state-space model identification from video data using deep encoders, IFAC-PapersOnLine, vol. 54, nr. 7, pp. 697-701, 2021.

L.C. Iacob et al., Deep Identification of Nonlinear Systems in Koopman Form, IEEE Conference on Decision and Control, 2021 (accepted).

Outline

‘Classical’ Approach

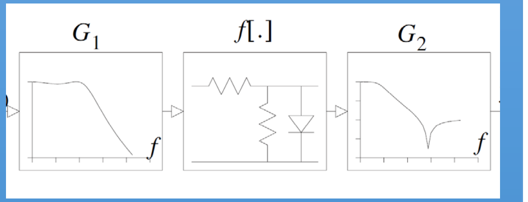
Challenges

Deep Subspace Encoder

Examples

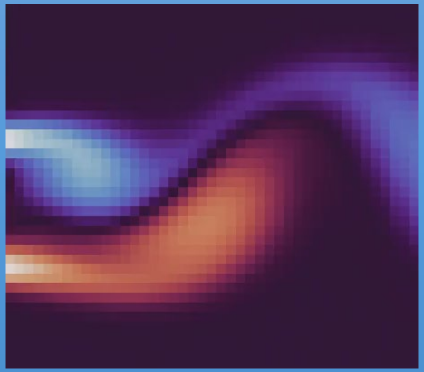
Examples

Wiener-Hammerstein Benchmark



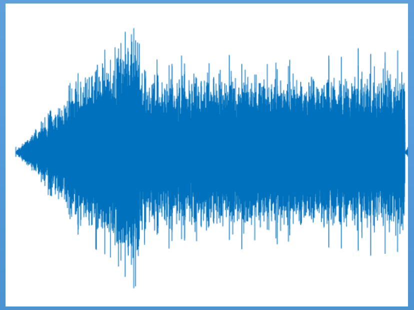
Vanilla Subspace Encoder

Von Karman Vortices



Spatiotemporal Encoder

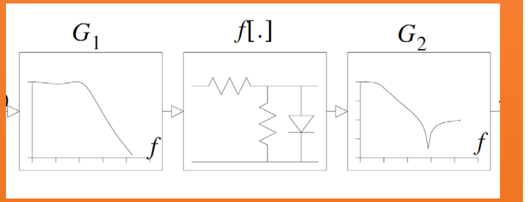
Silverbox Benchmark



Koopman Encoder

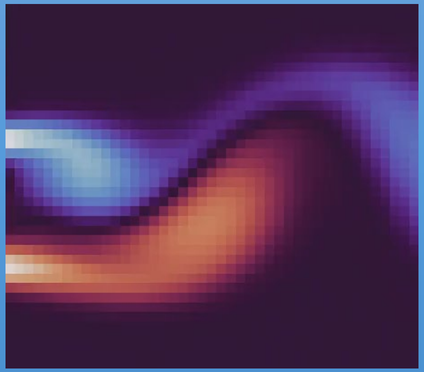
Examples

Wiener-Hammerstein Benchmark



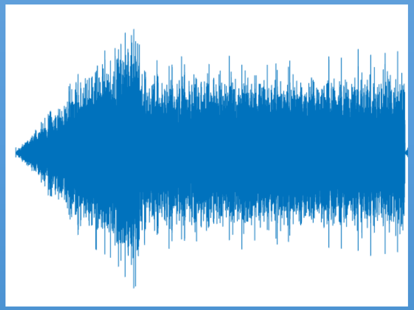
Vanilla Subspace Encoder

Von Karman Vortices



Spatiotemporal Encoder

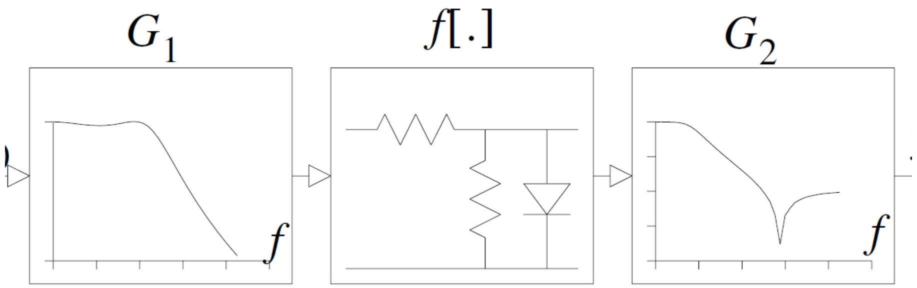
Silverbox Benchmark



Koopman Encoder

Wiener-Hammerstein Benchmark

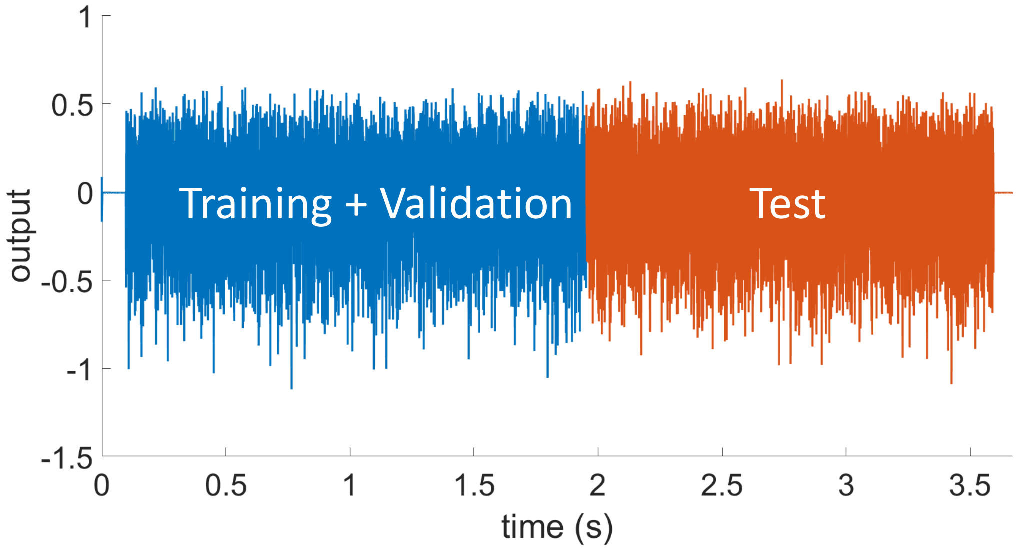
System:



G_1, G_2: 3rd order low-pass filters

$f()$: one sided soft saturation nonlinearity

Data:



80 10^3 samples for training

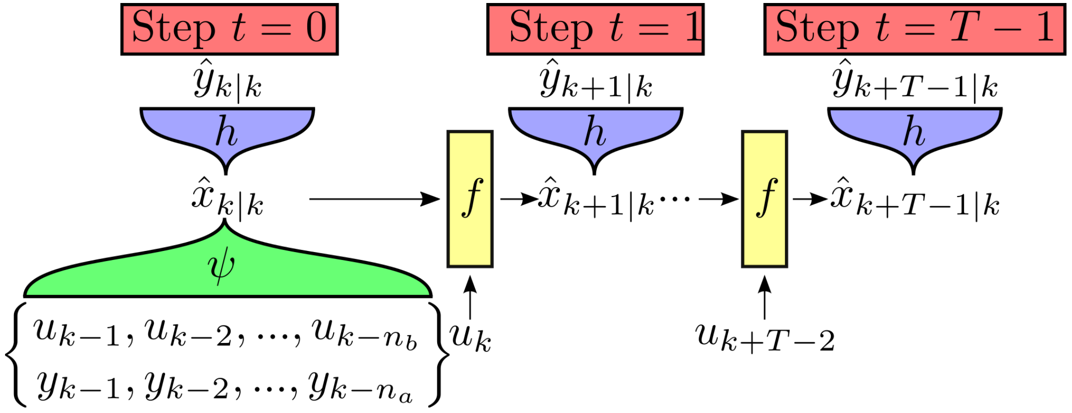
20 10^3 samples for validation

88 10^3 samples for test

G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

Wiener-Hammerstein Benchmark

Model:



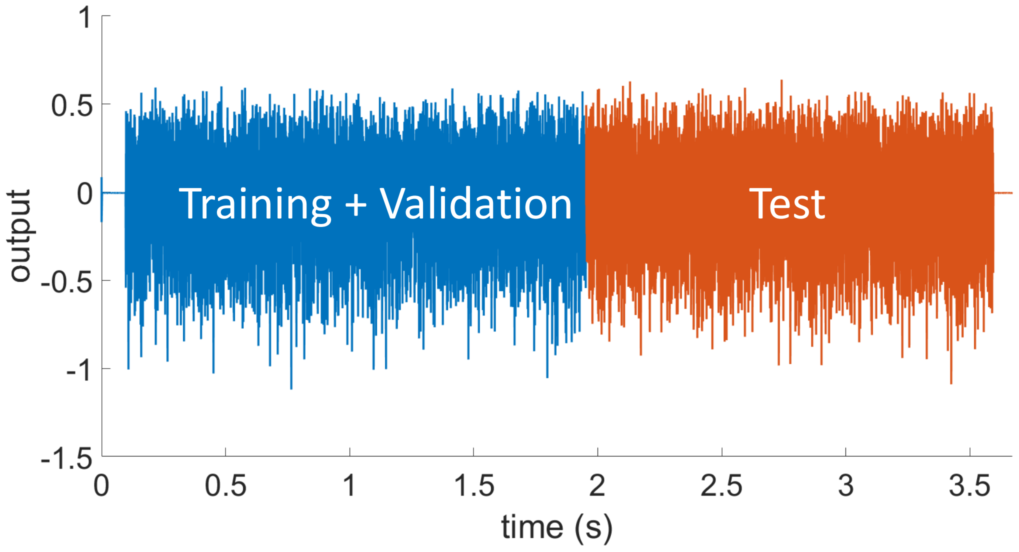
1-hidden layer, 15 neurons, tanh activation

$n_b = n_a = 50$, $n_x = 6$, $T = 80$

Adam optimizer, batch size: 1024, learning rate: 10^{-3}

Random parameter initialization

Data:



$80\ 10^3$ samples for training

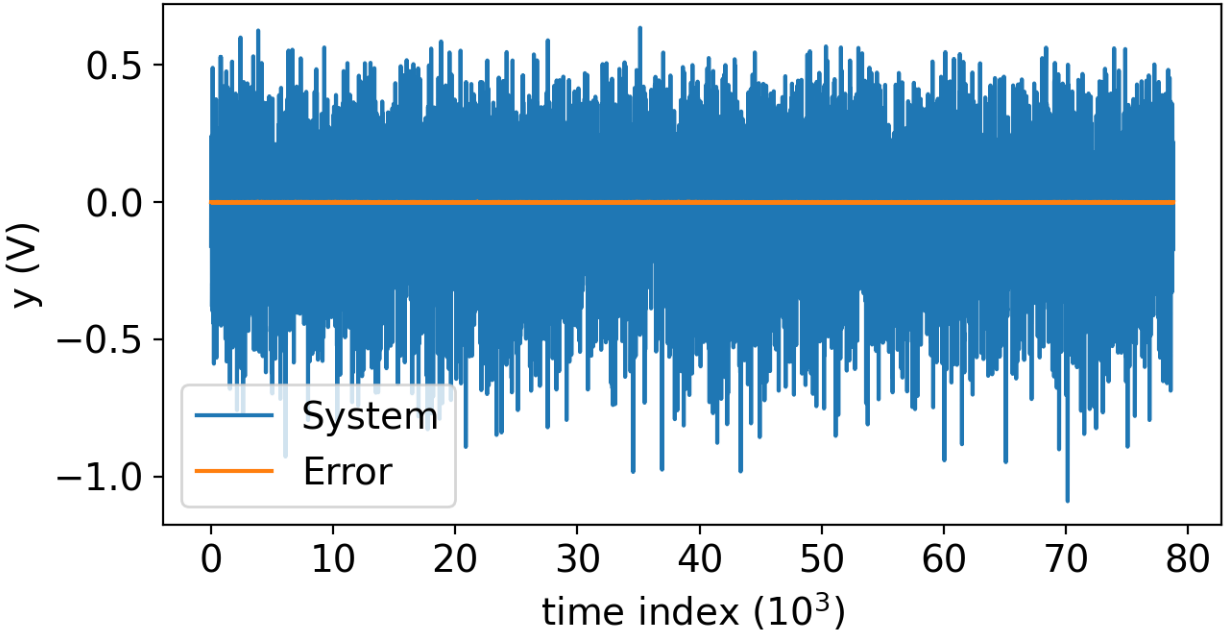
$20\ 10^3$ samples for validation

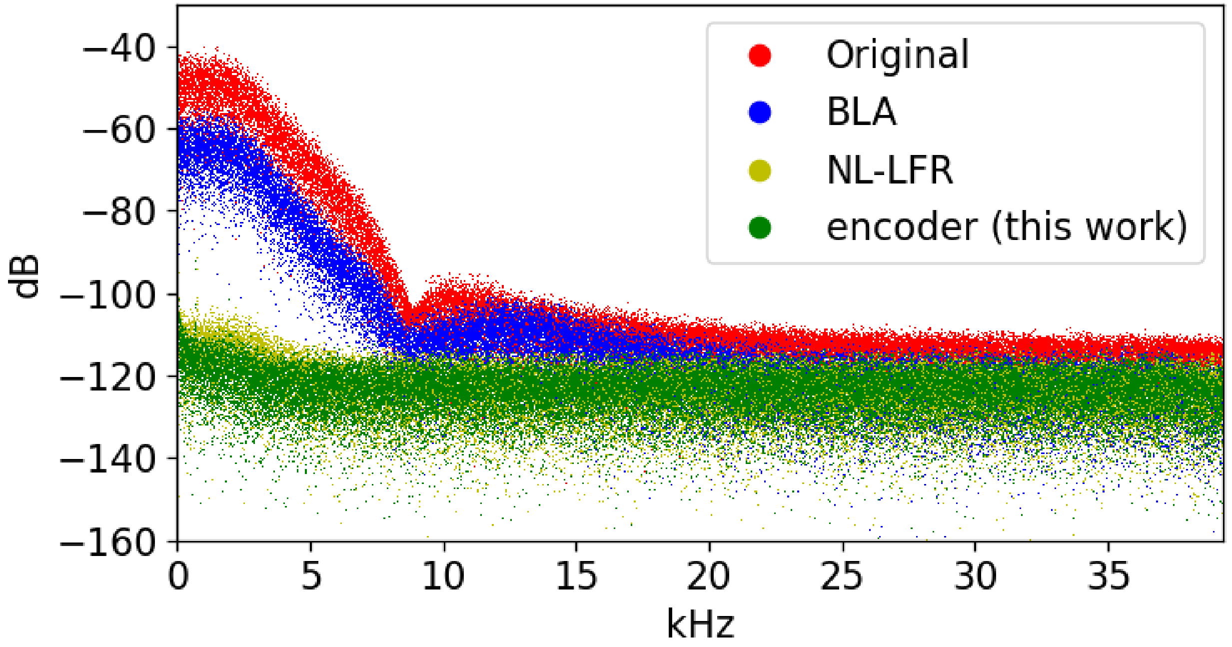
$88\ 10^3$ samples for test

EINDHOVEN UNIVERSITY OF TECHNOLOGY

G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

Wiener-Hammerstein Benchmark





G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

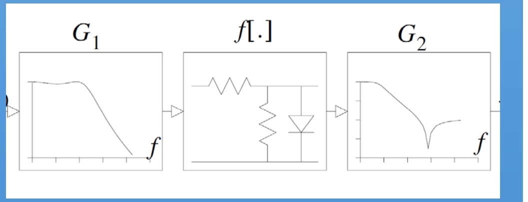
Wiener-Hammerstein Benchmark

Identification Method	Test RMS Simulation (mV)	Test NRMS Simulation
State-space Encoder (this work)	**0.241**	**0.0987%**
QBLA (Schoukens et al., 2014)	0.279	0.113%
Pole-zero splitting (Sjöberg et al., 2012)	0.30	0.123%
NL-LFR (Schoukens and Toth, 2020)	0.30	0.123%
PNLSS (Paduart et al., 2012)	0.42	0.172%
Generalized WH (Wills and Ninness, 2009)	0.49	0.200%
LS-SVM (Falck et al., 2009)	4.07	1.663%
Bio-social evolution (Naitali and Giri, 2016)	8.55	3.494%
Auto-encoder (reproduction) (Masti and Bemporad, 2018)	12.01	4.907%
Genetic Programming (Khandelwal, 2020)	23.50	9.605%
SVM (Marconato and Schoukens, 2009)	47.40	19.373%
BLA (Lauwers et al., 2009)	56.20	22.969%

G.I. Beintema et al., Nonlinear state-space identification using deep encoder networks, Learning for Dynamics and Control, PMLR vol. 144, pp. 241-250, May. 2020

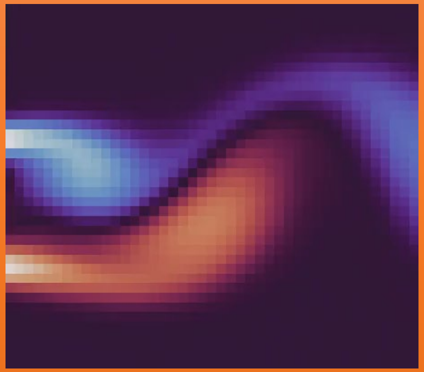
Examples

Wiener-Hammerstein Benchmark



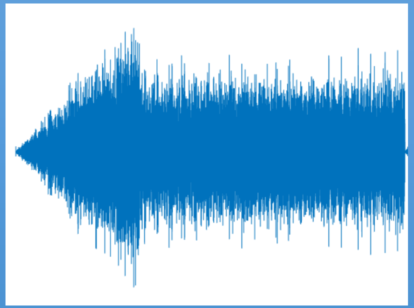
Vanilla Subspace Encoder

Von Karman Vortices



Spatiotemporal Encoder

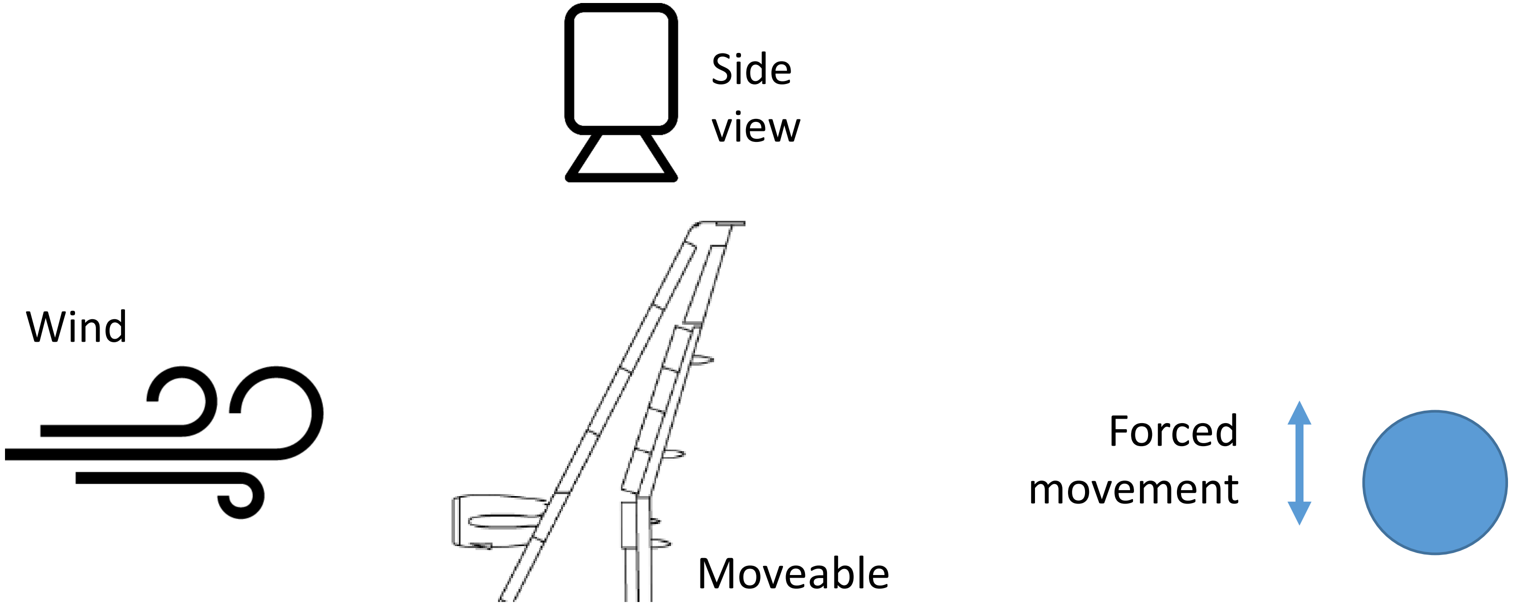
Silverbox Benchmark

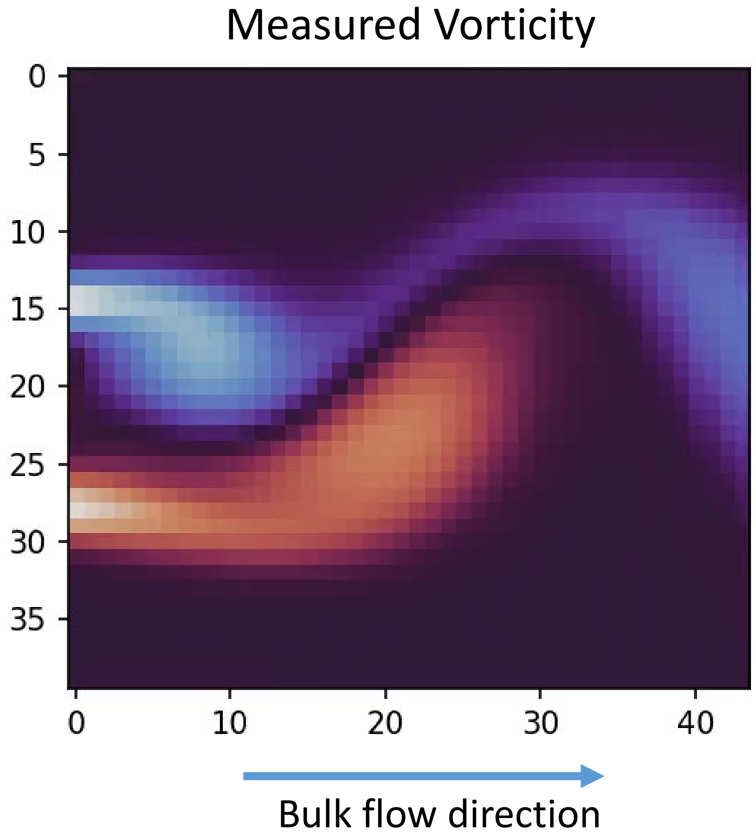


Koopman Encoder

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Von Karman Vortices





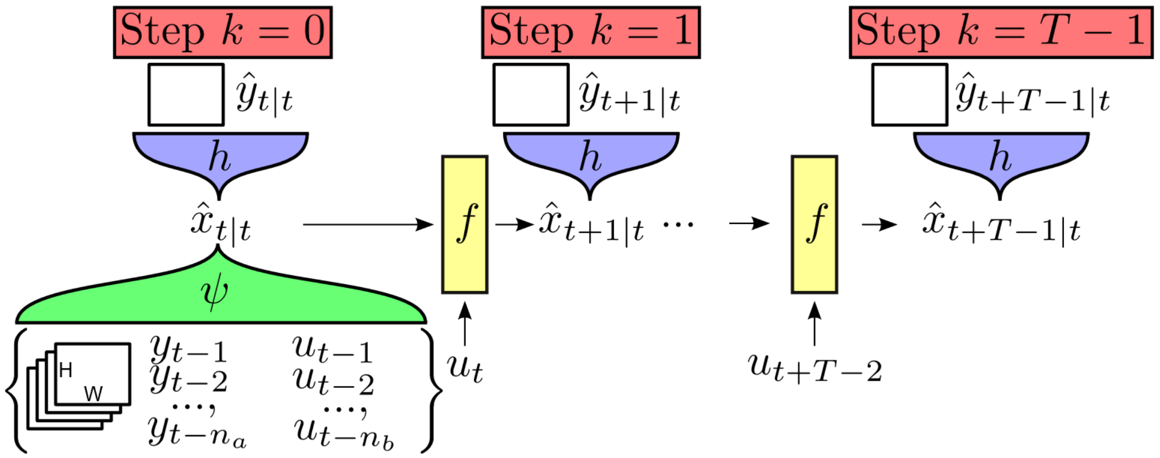
Input: Sinesweep cylinder position

Output: Pressure, x-,y- flow velocity
41 x 45 grid (1845 points)

Collaboration with Jan Decuyper, INDI, VUB

Von Karman Vortices

Model:



What is the form of h_θ and ψ_θ?

Fully Connected NN (MLP):
Flatten images into vectors
Disregards spatial information

Convolutional neural networks:
Applied directly on images
Exploits spatial information

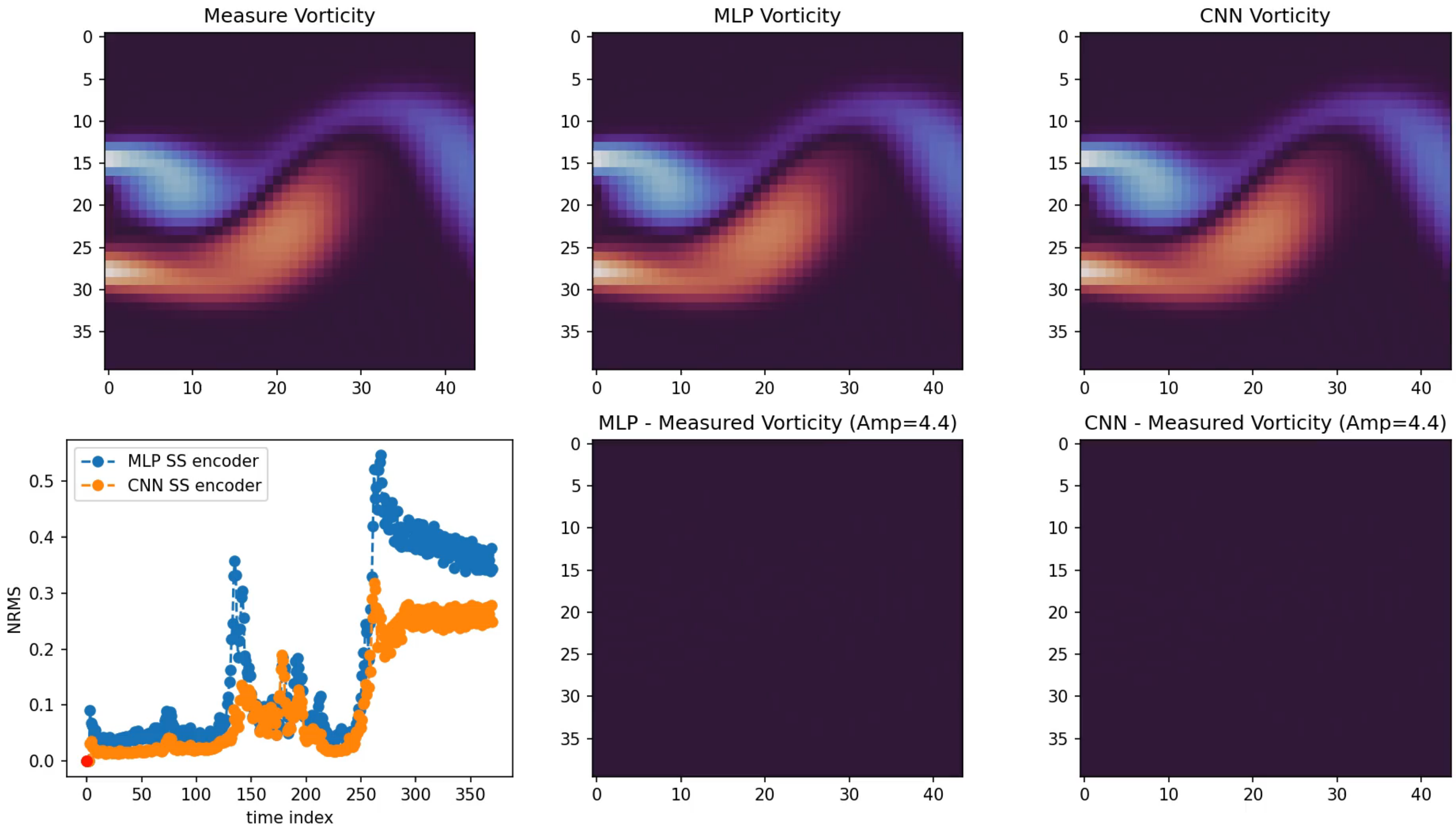
2-hidden layer, 64 neurons, tanh activation

$n_b = n_a = 3$, $n_x = 10$, $T = 30$

Adam optimizer, batch size: 256, learning rate: 10^{-3}

Random parameter initialization

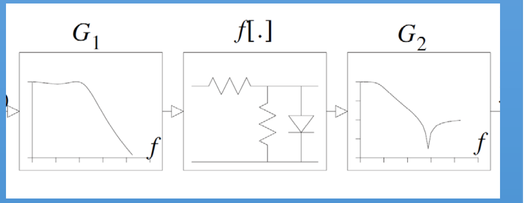
Von Karman Vortices



Collaboration with Jan Decuyper, INDI, VUB

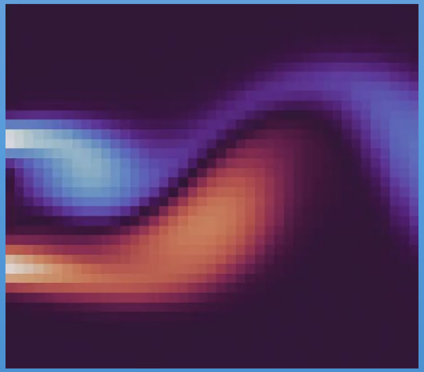
Examples

Wiener-Hammerstein Benchmark



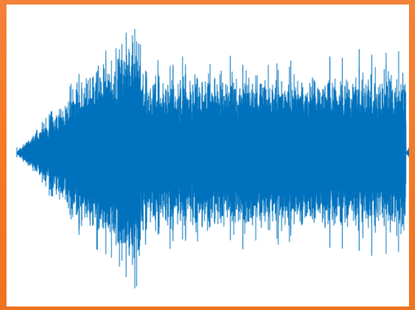
Vanilla Subspace Encoder

Von Karman Vortices



Spatiotemporal Encoder

Silverbox Benchmark



Koopman Encoder

Silverbox Benchmark

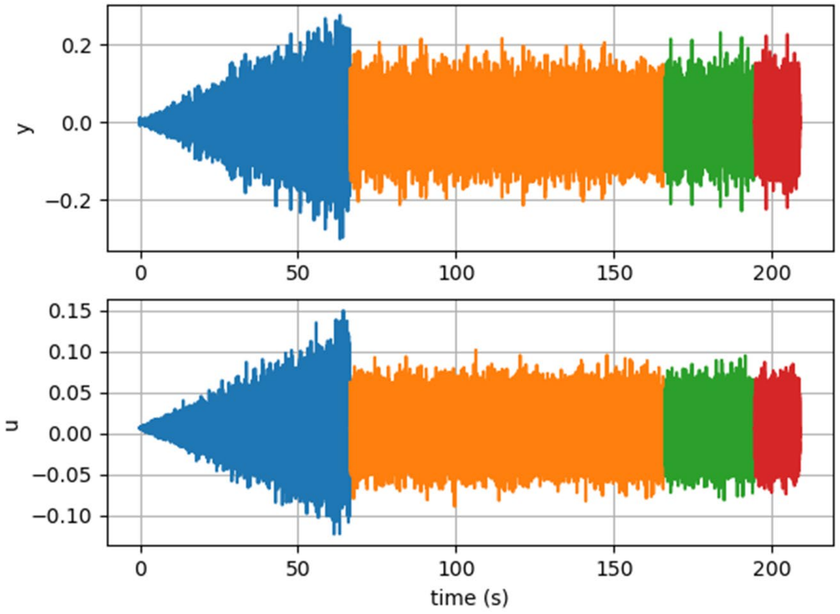
System: Forced Duffing Oscillator

$$m\ddot{y}(t) + d\dot{y}(t) + k_1 y(t) + k_3 y^3(t) = u(t)$$

Electrical implementation of mass-spring-damper

Cubic spring nonlinearity

Data:



Multisine Training, Validation, Test

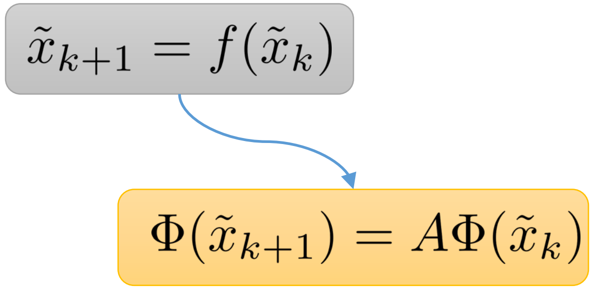
Arrowhead Test

L.C. Iacob et al., Deep Identification of Nonlinear Systems in Koopman Form, IEEE Conference on Decision and Control, 2021 (accepted).

Silverbox Benchmark: Koopman

Koopman model with input:

Idea: embed nonlinear dynamics in a linear model by lifting the states to a high (infinite) dimensional space.

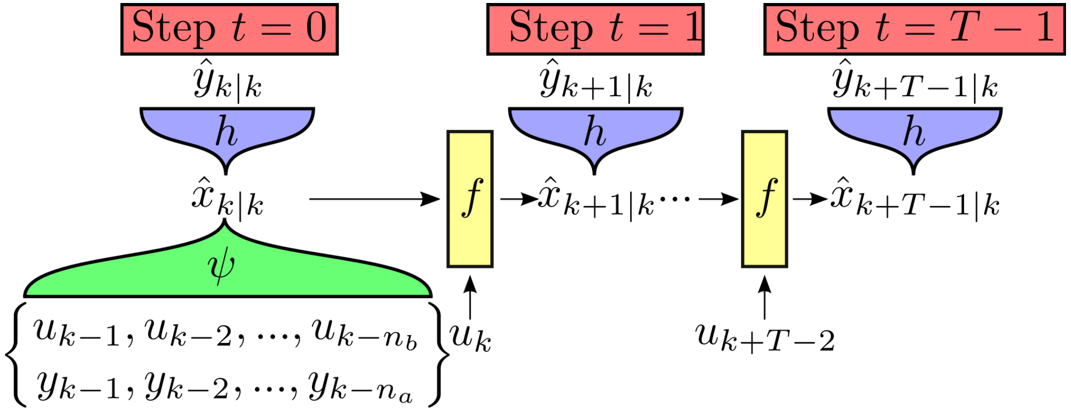


Koopman subspace Encoder:

Encoder simultaneously learns reconstructability map and lifting function

Model Dynamics:

$$f(x_k, u_k) = Ax_k + B(x_k)u_k$$
$$h(x_k) = Cx_k$$



n_a=n_b=10, n_x = 20, T = 49, batch size = 256, ADAM
2-hidden layer ANN for encoder and *B*, 40 neurons

L.C. Iacob et al., Deep Identification of Nonlinear Systems in Koopman Form, IEEE Conference on Decision and Control, 2021 (accepted).

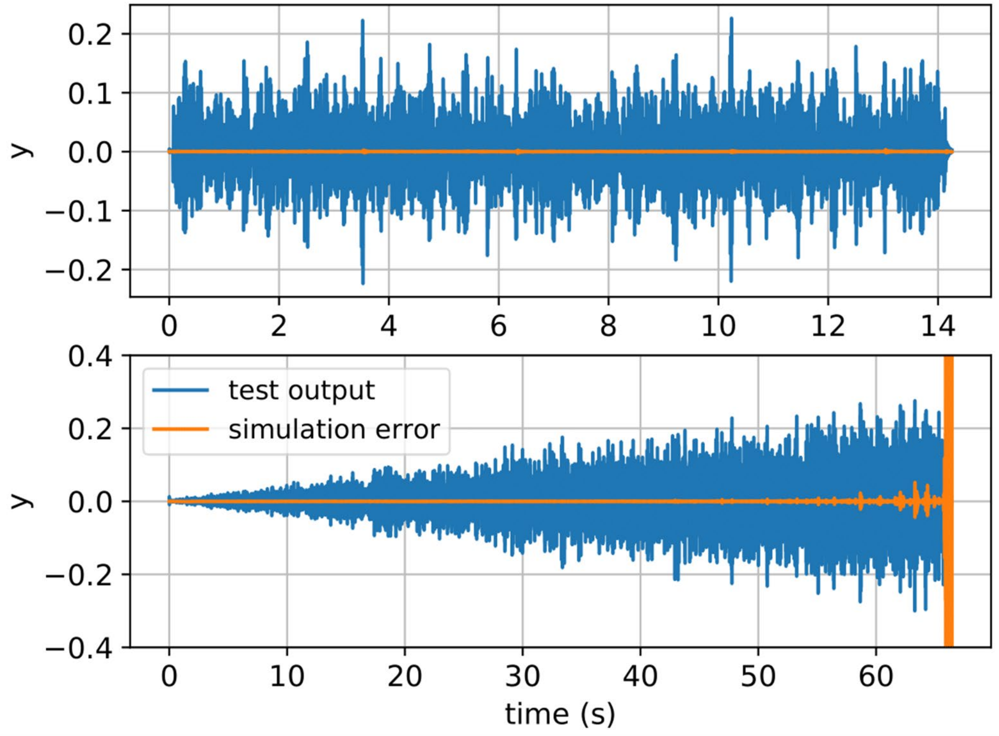
Silverbox Benchmark

	NRMS	RMS (V)
Test	0.00552	0.00029
Arrowhead	229.411	12.2502
Arrowhead - no extrapol.	0.00811	0.00033

Problem in the extrapolation region

Methods that use poly basis perform better

Results close to state of the art



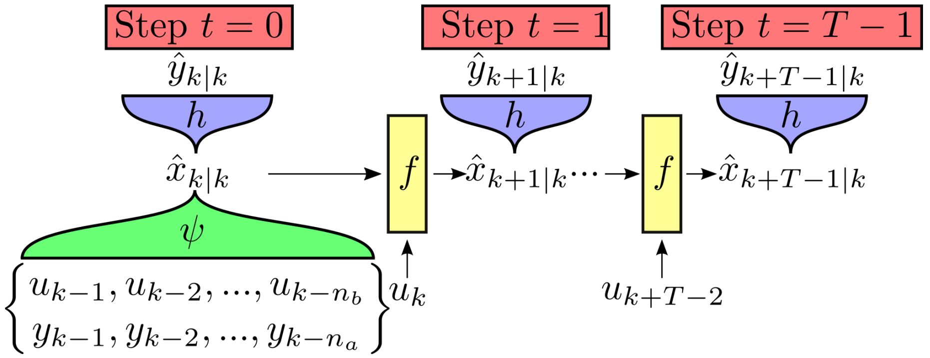
Multisine test (top), arrowhead test (bottom)

L.C. Iacob et al., Deep Identification of Nonlinear Systems in Koopman Form, IEEE Conference on Decision and Control, 2021 (accepted).

Conclusions

Deep Subspace Encoder combines:

- ANN state-space representations
- Multiple shooting / truncated simulation error
- State encoder / reconstructability map
- Batch optimization



Resulting in:

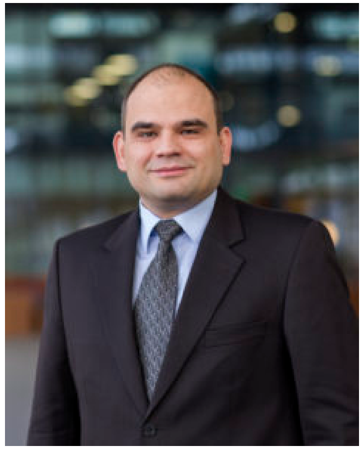
- Cost function smoothness
- Good scaling with data size / dimension
- State-of-the-art benchmarking results
- Flexible to include other model representations (thanks to automatic differentiation)

Implementation available in
Python DeepSI toolbox:
https://github.com/GerbenBeintema/deepSI

Team

Gerben I. Beintema
PhD Candidate

L. Cristi Iacob
PhD Candidate

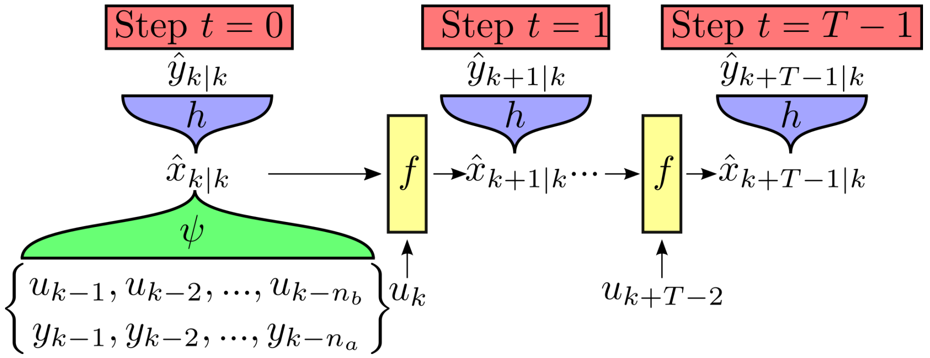


Roland Toth
Associate Professor

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Conclusions

Deep Subspace Encoder combines:
- ANN state-space representations
- Multiple shooting / truncated simulation error
- State encoder / reconstructability map
- Batch optimization



Resulting in:
- Cost function smoothness
- Good scaling with data size / dimension
- State-of-the-art benchmarking results
- Flexible to include other model representations (thanks to automatic differentiation)

Implementation available in
Python DeepSI toolbox:
https://github.com/GerbenBeintema/deepSI