Recurrent Structures in System Identification

Antônio H. Ribeiro

Universidade Federal de Minas Gerais - UFMG

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica - PPGEE

Supervisor: Luis A. Aguirre

July 19, 2017

Overview

Introduction

Problem Statement

What is System Identification?

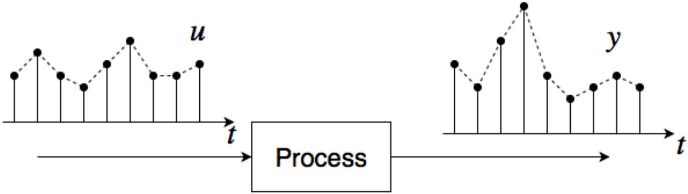


Figure: The system identification problem.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
    - Dynamic representation;
    - Approximation function;
    - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
    - Validation data.
    - One-step-ahead prediction *vs* free-run simulation.

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \mathbf{\Theta}\right).$$

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \mathbf{\Theta}\right).$$

Typical Steps
System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Output Error, Equation Error and Error-in-Variables

$$\mathbf{u}[k] = \mathbf{u}^*[k] + \mathbf{s}[k],$$
$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]\right) + \mathbf{v}[k],$$
$$\mathbf{y}[k] = \mathbf{y}^*[k] + \mathbf{w}[k].$$

Output Error, Equation Error and Error-in-Variables

$$\mathbf{u}[k] = \mathbf{u}^*[k] + \mathbf{s}[k],$$

$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]\right) + \mathbf{v}[k],$$

$$\mathbf{y}[k] = \mathbf{y}^*[k] + \mathbf{w}[k].$$

Output Error, Equation Error and Error-in-Variables

$$\mathbf{u}[k] = \mathbf{u}^*[k] + \mathbf{s}[k],$$

$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]\right) + \mathbf{v}[k],$$

$$\mathbf{y}[k] = \mathbf{y}^*[k] + \mathbf{w}[k].$$

Output Error, Equation Error and Error-in-Variables

$$\mathbf{u}[k] = \mathbf{u}^*[k] + \mathbf{s}[k],$$
$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]\right) + \mathbf{v}[k],$$
$$\mathbf{y}[k] = \mathbf{y}^*[k] + \mathbf{w}[k].$$

Noise Model

System Identification Procedure

Output Error, Equation Error and Error-in-Variables

$$\mathbf{u}[k] = \mathbf{u}^*[k] + \mathbf{s}[k],$$
$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]\right) + \mathbf{v}[k],$$
$$\mathbf{y}[k] = \mathbf{y}^*[k] + \mathbf{w}[k].$$

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

Typical Steps

System Identification Procedure

1. Test design and data collection;
2. Choice of mathematical representation;
    - Dynamic representation;
    - Approximation function;
    - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
    - Validation data.
    - One-step-ahead prediction *vs* free-run simulation.

Validation Data

System Identification Procedure

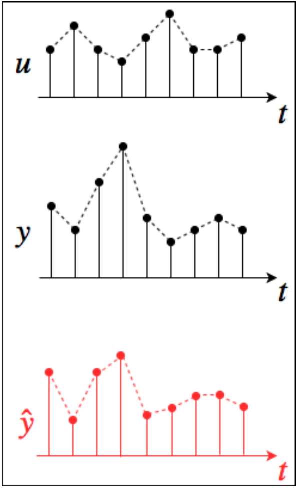


Figure: Comparison between measured data and predicted values

Typical Steps

System Identification Procedure

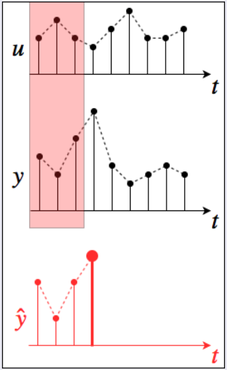
1. Test design and data collection;
2. Choice of mathematical representation;
   - Dynamic representation;
   - Approximation function;
   - Noise model.
3. Choice of model order and structure;
4. Estimation of model parameters;
5. Model validation.
   - Validation data.
   - One-step-ahead prediction *vs* free-run simulation.

One-step-ahead Prediction *vs* Free-run Simulation

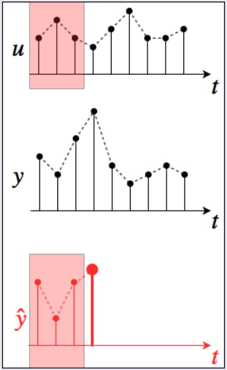
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



Free-run Simulation

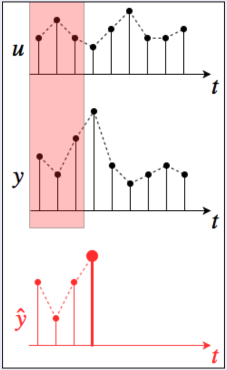


One-step-ahead Prediction *vs* Free-run Simulation

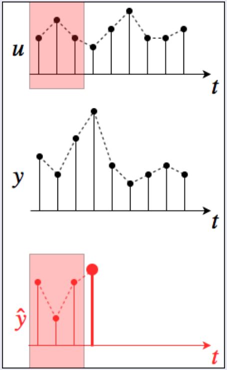
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



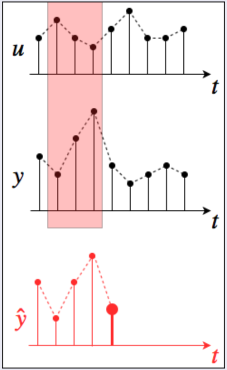
Free-run Simulation



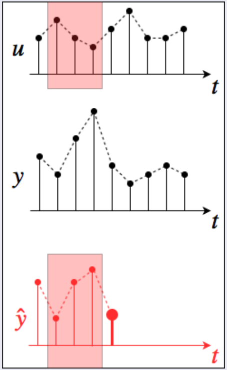
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



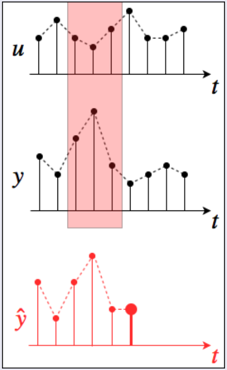
Free-run Simulation



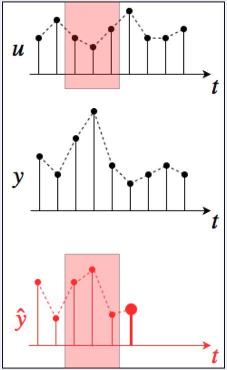
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



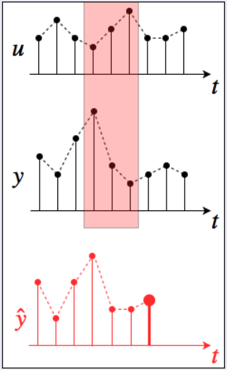
Free-run Simulation



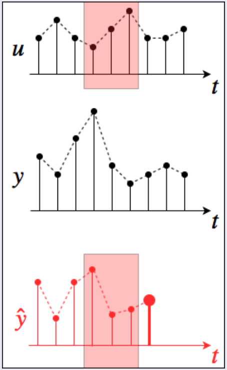
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



Free-run Simulation

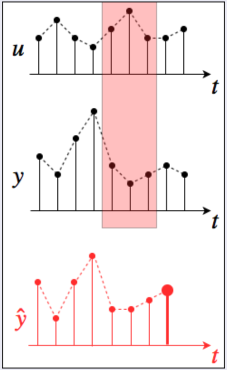


One-step-ahead Prediction *vs* Free-run Simulation

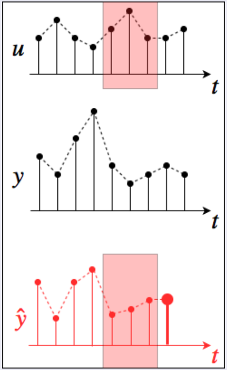
Nonlinear Difference Equation

$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}\right).$$

One-step-ahead Prediction



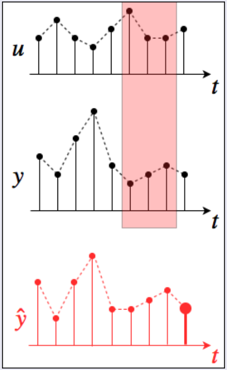
Free-run Simulation



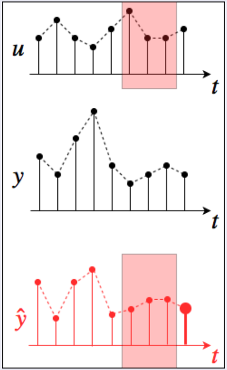
Nonlinear Difference Equation

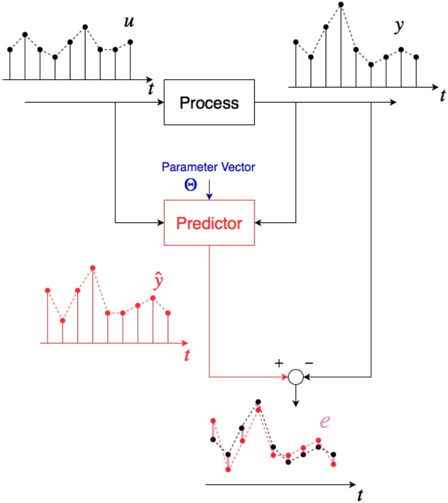
$$\mathbf{y}[k] = \mathbf{F}\left(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \mathbf{\Theta} \right).$$

One-step-ahead Prediction



Free-run Simulation





General Framework

- Noise model ⇒ Optimal Predictor:

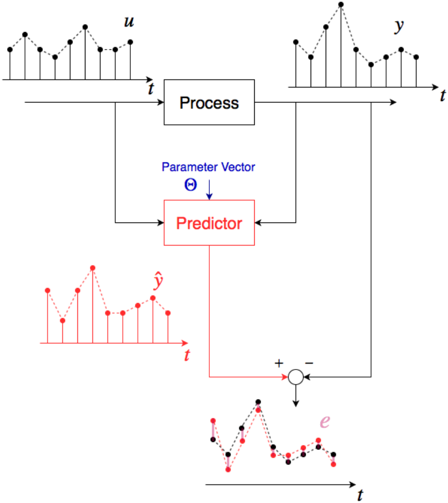
$$\hat{\mathbf{y}}[k] = E\{\mathbf{y}[k] \mid k-1\}$$

- Compute errors:

$$\mathbf{e}[k] = \hat{\mathbf{y}}[k] - \mathbf{y}[k]$$

- Find parameter Θ such the sum of square errors is minimized:

$$\min_{\Theta} \sum_k \|\mathbf{e}[k]\|^2$$



General Framework

- Noise model ⇒ Optimal Predictor:

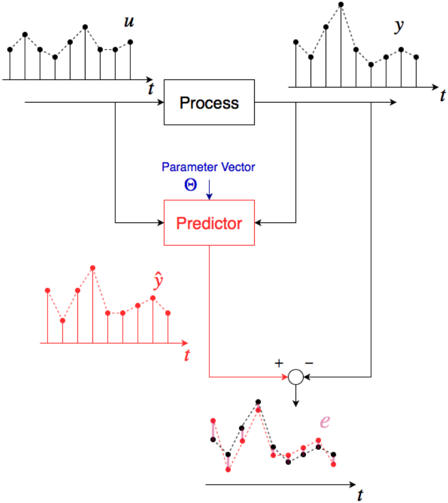
$$\hat{\mathbf{y}}[k] = E\{\mathbf{y}[k] \mid k-1\}$$

- Compute errors:

$$\mathbf{e}[k] = \hat{\mathbf{y}}[k] - \mathbf{y}[k]$$

- Find parameter Θ such the sum of square errors is minimized:

$$\min_{\Theta} \sum_k \|\mathbf{e}[k]\|^2$$



General Framework

- Noise model ⇒ Optimal Predictor:

$$\hat{\mathbf{y}}[k] = E\{\mathbf{y}[k] \mid k-1\}$$

- Compute errors:

$$\mathbf{e}[k] = \hat{\mathbf{y}}[k] - \mathbf{y}[k]$$

- Find parameter $\boldsymbol{\Theta}$ such the sum of square errors is minimized:

$$\min_{\boldsymbol{\Theta}} \sum_k \|\mathbf{e}[k]\|^2$$

NARX (Nonlinear AutoRegressive with eXogenous input) model.

True system

$$\mathbf{y}[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}) + \underbrace{\mathbf{v}[k]}_{\text{white noise}}.$$

Optimal Predictor

One-step-ahead prediction:

$$\hat{\mathbf{y}}_1[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \boldsymbol{\Theta}).$$

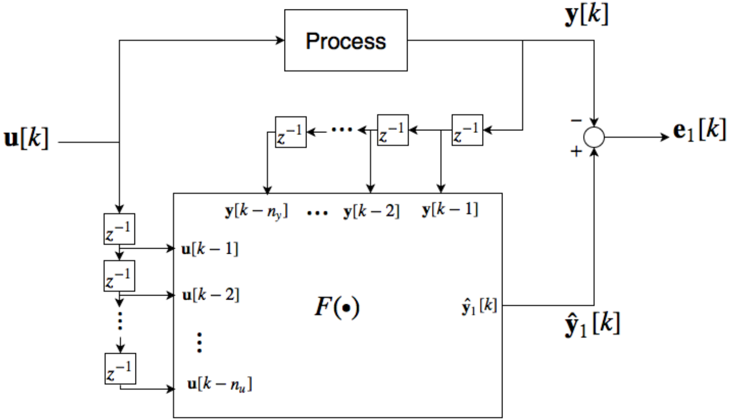


Figure: NARX model prediction error.

NOE Model

Prediction Error Methods

NOE (Nonlinear Output Error) model.

True system

$$\mathbf{y}^*[k] = \mathbf{F}\left(\mathbf{y}^*[k-1], \mathbf{y}^*[k-2], \mathbf{y}^*[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \mathbf{\Theta}\right),$$
$$\mathbf{y}[k] = \mathbf{y}^*[k] + \underbrace{\mathbf{w}[k]}_{\text{white noise}} .$$

Optimal Predictor

Free-run simulation:

$$\hat{\mathbf{y}}_s[k] = \mathbf{F}(\hat{\mathbf{y}}_s[k-1], \hat{\mathbf{y}}_s[k-2], \hat{\mathbf{y}}_s[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3]; \mathbf{\Theta}).$$

NOE Model

Prediction Error Methods

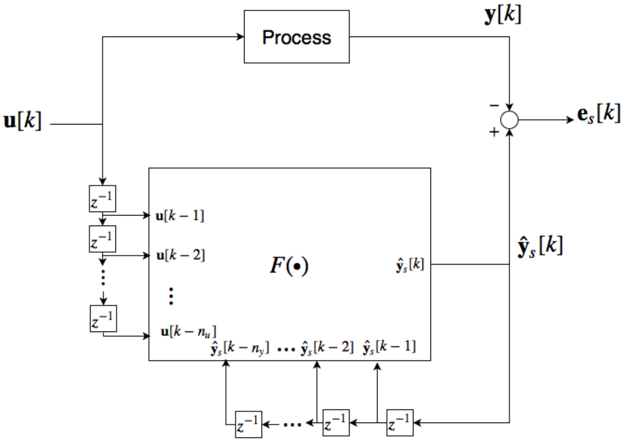


Figure: NOE model prediction error.

NARMAX Model

Prediction Error Methods

NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous input) model.

True system

$$\mathbf{y}[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{v}[k-1], \mathbf{v}[k-2], \mathbf{v}[k-3]; \boldsymbol{\Theta}) + \underbrace{\mathbf{v}[k]}_{\text{white noise}}.$$

Optimal Predictor

$$\hat{\mathbf{y}}_v[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{y}[k-1] - \hat{\mathbf{y}}[k-1], \mathbf{y}[k-2] - \hat{\mathbf{y}}[k-2], \mathbf{y}[k-3] - \hat{\mathbf{y}}[k-3]; \boldsymbol{\Theta}).$$

NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous input) model.

True system

$$\mathbf{y}[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{v}[k-1], \mathbf{v}[k-2], \mathbf{v}[k-3]; \mathbf{\Theta}) + \underbrace{\mathbf{v}[k]}_{\text{white noise}} .$$

Optimal Predictor

$$\hat{\mathbf{y}}_v[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{y}[k-1] - \hat{\mathbf{y}}[k-1], \mathbf{y}[k-2] - \hat{\mathbf{y}}[k-2], \mathbf{y}[k-3] - \hat{\mathbf{y}}[k-3]; \mathbf{\Theta}).$$

NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous input) model.

True system

$$\mathbf{y}[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{v}[k-1], \mathbf{v}[k-2], \mathbf{v}[k-3]; \boldsymbol{\Theta}) + \underbrace{\mathbf{v}[k]}_{\text{white noise}}.$$

Optimal Predictor

$$\hat{\mathbf{y}}_v[k] = \mathbf{F}(\mathbf{y}[k-1], \mathbf{y}[k-2], \mathbf{y}[k-3], \mathbf{u}[k-1], \mathbf{u}[k-2], \mathbf{u}[k-3], \mathbf{y}[k-1] - \hat{\mathbf{y}}[k-1], \mathbf{y}[k-2] - \hat{\mathbf{y}}[k-2], \mathbf{y}[k-3] - \hat{\mathbf{y}}[k-3]; \boldsymbol{\Theta}).$$

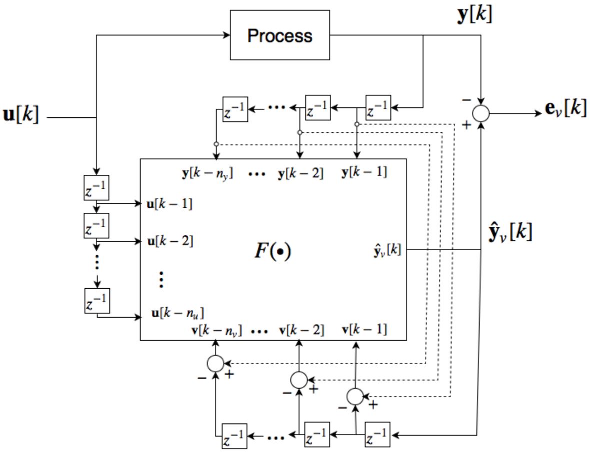


Figure: NARMAX model prediction error.

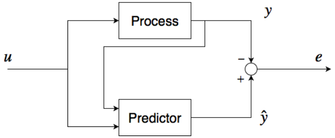


Figure: Prediction depends only on measured values.

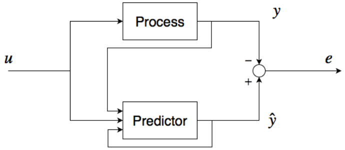


Figure: Predictor has a recurrent structure.

Chalenges

- Unboundedness;
- Multiple Minima.

Minimizing the sum of squared errors:

$$\min_{\boldsymbol{\Theta}} V(\boldsymbol{\Theta}) = \|\mathbf{e}(\boldsymbol{\Theta})\|^2$$

Derivatives:

$$\nabla V(\mathbf{\Theta}) = J(\mathbf{\Theta})^T \ \mathbf{e}(\mathbf{\Theta}),$$

$$\nabla^2 V(\mathbf{\Theta}) = J^T(\mathbf{\Theta}) J(\mathbf{\Theta}) + \sum_{i=1}^{N_e} e_i(\mathbf{\Theta}) \left(\nabla^2 e_i(\mathbf{\Theta})\right).$$

Objective Function Derivatives

Nonlinear Least Squares

Derivatives:

$$\nabla V(\mathbf{\Theta}) = J(\mathbf{\Theta})^T \; \mathbf{e}(\mathbf{\Theta}),$$

$$\nabla^2 V(\mathbf{\Theta}) = J^T(\mathbf{\Theta}) J(\mathbf{\Theta}) + \sum_{i=1}^{N_e} e_i(\mathbf{\Theta}) \left(\nabla^2 e_i(\mathbf{\Theta})\right).$$

- Iterative Algorithms. Starting in $\mathbf{\Theta}^0$ updates the solution:

$$\mathbf{\Theta}^{n+1} = \mathbf{\Theta}^n + \Delta\mathbf{\Theta}^n$$

- Gauss-Newton:

$$\Delta\mathbf{\Theta} = -\underbrace{\mu}_{\text{step lenght}}\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}}\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\,\mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

- Levenberg-Marquardt:

$$\Delta\mathbf{\Theta} = -\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}}+\lambda D\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\,\mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

- Iterative Algorithms. Starting in $\mathbf{\Theta}^0$ updates the solution:

$$\mathbf{\Theta}^{n+1} = \mathbf{\Theta}^n + \Delta\mathbf{\Theta}^n$$

- Gauss-Newton:

$$\Delta\mathbf{\Theta} = -\underbrace{\mu}_{\text{step lenght}}\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}}\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\,\mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

- Levenberg-Marquardt:

$$\Delta\mathbf{\Theta} = -\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}}+\lambda D\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\,\mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

- Iterative Algorithms. Starting in $\mathbf{\Theta}^0$ updates the solution:

$$\mathbf{\Theta}^{n+1} = \mathbf{\Theta}^n + \Delta\mathbf{\Theta}^n$$

- Gauss-Newton:

$$\Delta\mathbf{\Theta} = -\underbrace{\mu}_{\text{step lenght}}\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}}\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\ \mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

- Levenberg-Marquardt:

$$\Delta\mathbf{\Theta} = -\Big(\underbrace{J^T(\mathbf{\Theta})J(\mathbf{\Theta})}_{\text{Hessian approx.}} + \lambda D\Big)^{-1}\underbrace{J(\mathbf{\Theta})^T\ \mathbf{e}(\mathbf{\Theta})}_{\text{grad.}}$$

“Parallel Training Considered Harmful?”

- Parallel training $\Rightarrow$ NOE model;
- Series-parallel training $\Rightarrow$ NARX model.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

References

"Parallel Training Considered Harmful?"

Narendra, K. S. and Parthasarathy, K. (1990).
Identification and control of dynamical systems using neural networks.
IEEE Transactions on Neural Networks, 1(1):4–27.

Zhang, D.-y., Sun, L.-p., and Cao, J. (2006).
Modeling of temperature-humidity for wood drying based on time-delay neural network.
Journal of Forestry Research, 17(2):141–144.

Singh, M., Singh, I., and Verma, A. (2013).
Identification on non linear series-parallel model using neural network.
MIT Int. J. Electr. Instrumen. Eng, 3(1):21–23.

Beale, M. H., Hagan, M. T., and Demuth, H. B. (2017).
Neural network toolbox for use with MATLAB.
Technical report, Mathworks.

Diaconescu, E. (2008).
The use of NARX neural networks to predict chaotic time series.
WSEAS Transactions on Computer Research, 3(3):182–191.

References

"Parallel Training Considered Harmful?"

Saad, M., Bigras, P., Dessaint, L.-A., and Al-Haddad, K. (1994).
Adaptive robot control using neural networks.
IEEE Transactions on Industrial Electronics, 41(2):173–181.

Saggar, M., Meriçli, T., Andoni, S., and Miikkulainen, R. (2007).
System identification for the Hodgkin-Huxley model using artificial neural networks.
In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 2239–2244. IEEE.

Warwick, K. and Craddock, R. (1996).
An introduction to radial basis functions for system identification. a comparison with other neural network methods.
In *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, volume 1, pages 464–469. IEEE.

Kamińnski, W., Strumitto, P., and Tomczak, E. (1996).
Genetic algorithms and artificial neural networks for description of thermal deterioration processes.
Drying Technology, 14(9):2117–2133.

Rahman, M. F., Devanathan, R., and Kuanyi, Z. (2000).
Neural network approach for linearizing control of nonlinear process plants.
IEEE Transactions on Industrial Electronics, 47(2):470–477.

Petrović, E., Ćojbašić, Ž., Ristić-Durrant, D., Nikolić, V., Ćirić, I., and Matić, S. (2013).
Kalman filter and NARX neural network for robot vision based human tracking.
Facta Universitatis, Series: Automatic Control And Robotics, 12(1):43–51.

Tijani, I. B., Akmeliawati, R., Legowo, A., and Budiyono, A. (2014).
Nonlinear identification of a small scale unmanned helicopter using optimized NARX network with multiobjective differential evolution.
Engineering Applications of Artificial Intelligence, 33:99–115.

Khan, E. A., Elgamal, M. A., and Shaarawy, S. M. (2015).
Forecasting the number of muslim pilgrims using NARX neural networks with a comparison study with other modern methods.
British Journal of Mathematics & Computer Science, 6(5):394.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

The following dynamic systems are present during the system identification procedure:

1. True System;
2. Predictor ;
3. Estimated Model.

Dynamic Systems Present During Identification

Parallel Training and Unbounded Signals

The following dynamic systems are present during the system identification procedure:

1. True System;
2. Predictor;
3. Estimated Model.

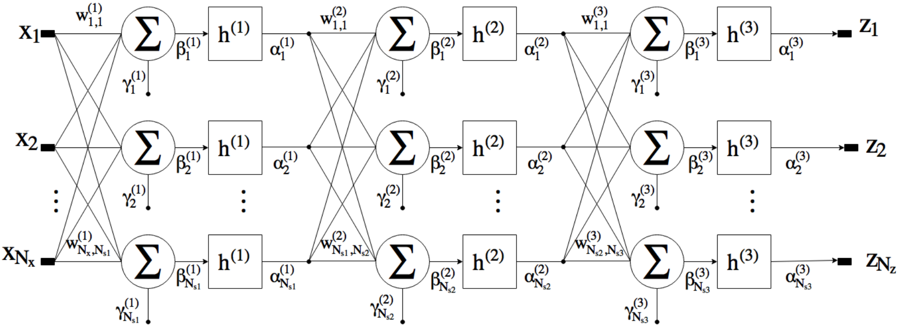


Figure: Three-layer feedforward network.

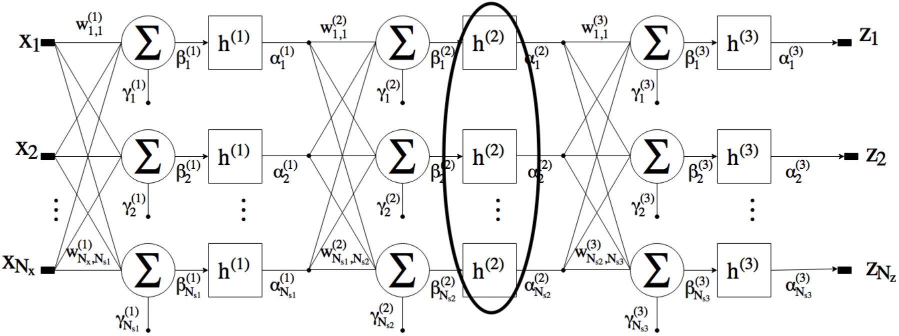


Figure: Three-layer feedforward network.

Feedforward Network

Neural Network Training

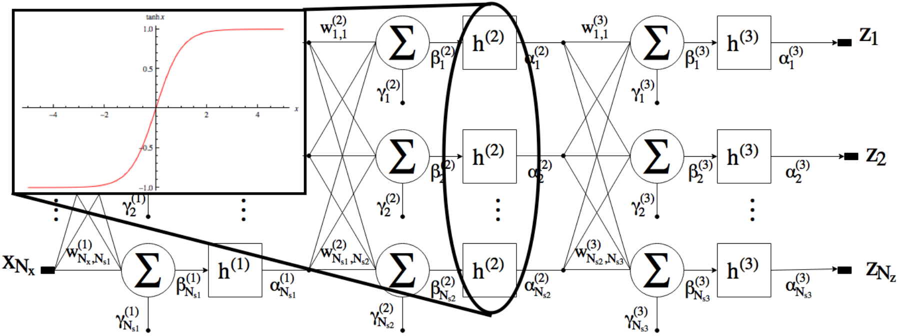


Figure: Three-layer feedforward network.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Computational Cost per Stage

Complexity Analysis

Stage - Levenberg-Marquardt	**Series-parallel**	**Parallel**
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\boldsymbol{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}.$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage

Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\boldsymbol{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}$.	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage
Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\mathbf{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}.$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage

Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\boldsymbol{\Delta\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T\ \mathbf{e}$.	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage

Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector $\mathbf{e}$	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\boldsymbol{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}$.	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage

Complexity Analysis

Stage - Levenberg-Marquardt	**Series-parallel**	**Parallel**
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\boldsymbol{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \; \mathbf{e}.$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Feedforward Network

Neural Network Training

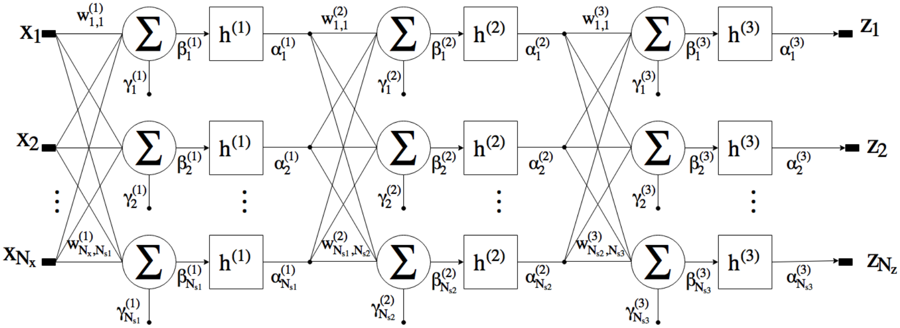


Figure: Three-layer feedforward network.

Computational Cost per Stage

Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector $\mathbf{e}$	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\boldsymbol{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}$.	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Computational Cost per Stage

Complexity Analysis

Stage	Series-parallel	Parallel
Compute error vector **e**	$\mathcal{O}(N \cdot N_w)$	$\mathcal{O}(N \cdot N_w)$
Compute Jacobian matrix J	$\mathcal{O}(N \cdot N_w \cdot N_y)$	$\mathcal{O}(N \cdot N_\Theta \cdot N_y^2)$
Parameter update $\Delta\mathbf{\Theta} = -\left(J^T J + \lambda D\right)^{-1} J^T \mathbf{e}$.	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$	$\mathcal{O}(N \cdot N_\Theta^2 + N_\Theta^3)$

Table: Complexity Analysis

Relation

$$N_y < N_y^2 < N_w \approx N_\Theta$$

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Series-parallel training alleged advantages

Series-parallel to be preferred [Narendra and Parthasarathy, 1990]:

1. Bounded signals;
2. Smaller computational cost;
3. Simulated output should tend to the real one, therefore the results should not be significantly different;
4. More accurate inputs to the neural network during training. *

Ribeiro, A. H., and Aguirre, L. A. (2017)
"Parallel Training Considered Harmful?": Comparing Series-Parallel and Parallel Feedforward Network Training.
arXiv preprint arXiv:1706.07119.

Computer Generated Example

Comparing Parallel and Series-parallel Models

Problem Statement

- Generate data using the following system: [Chen et al., 1990]

$$
\begin{aligned}
y^*[k] &= (0.8 - 0.5\exp(-y^*[k-1]^2)y^*[k-1] - \\
&\quad (0.3 + 0.9\exp(-y^*[k-1]^2)y^*[k-2] + u[k-1] + \\
&\quad 0.2u[k-2] + 0.1u[k-1]u[k-2] + v[k] \\
y[k] &= y^*[k] + w[k].
\end{aligned}
$$

- 10 nodes in the hidden layer;
- 800 samples for identification and 200 samples for validation;
- Compare error in validation window.

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

Computer Generated Example

Comparing Parallel and Series-parallel Models

Problem Statement

- Generate data using the following system: [Chen et al., 1990]

$$\begin{aligned} y^*[k] &= (0.8 - 0.5\exp(-y^*[k-1]^2)y^*[k-1] - \\ &\quad (0.3 + 0.9\exp(-y^*[k-1]^2)y^*[k-2] + u[k-1] + \\ &\quad 0.2u[k-2] + 0.1u[k-1]u[k-2] + v[k] \\ y[k] &= y^*[k] + w[k]. \end{aligned}$$

- 10 nodes in the hidden layer;
- 800 samples for identification and 200 samples for validation;
- Compare error in validation window.

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

Computer Generated Example

Comparing Parallel and Series-parallel Models

Problem Statement

- Generate data using the following system: [Chen et al., 1990]

$$\begin{aligned} y^*[k] &= (0.8 - 0.5\exp(-y^*[k-1]^2)y^*[k-1] - \\ &\quad (0.3 + 0.9\exp(-y^*[k-1]^2)y^*[k-2] + u[k-1] + \\ &\quad 0.2u[k-2] + 0.1u[k-1]u[k-2] + v[k] \\ y[k] &= y^*[k] + w[k]. \end{aligned}$$

- 10 nodes in the hidden layer;
- 800 samples for identification and 200 samples for validation;
- Compare error in validation window.

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

Computer Generated Example

Comparing Parallel and Series-parallel Models

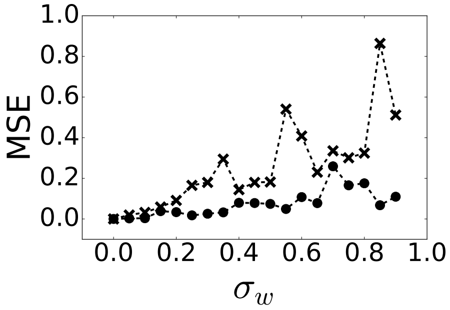
Problem Statement

- Generate data using the following system: [Chen et al., 1990]

$$
\begin{aligned}
y^*[k] &= (0.8 - 0.5\exp(-y^*[k-1]^2)y^*[k-1] - \\
&\quad (0.3 + 0.9\exp(-y^*[k-1]^2)y^*[k-2] + u[k-1] + \\
&\quad 0.2u[k-2] + 0.1u[k-1]u[k-2] + v[k] \\
y[k] &= y^*[k] + w[k].
\end{aligned}
$$

- 10 nodes in the hidden layer;
- 800 samples for identification and 200 samples for validation;
- Compare error in validation window.

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



(a) $\sigma_v = 0$; (b) $\sigma_w = 0$;

Figure: MSE (mean square error) *vs* noise levels on the validation window for parallel training ($\bullet$) and series-parallel training ($\times$).

Table: Running time.

Experiment Conditions		Execution time	
N_{hidden}	N	Parallel Training	Series-parallel Training
10	1000 samples	3.7 s	3.1 s
30	1000 samples	6.4 s	5.7 s
10	5000 samples	14.6 s	11.0 s
30	5000 samples	18.5 s	17.5 s

Computer Generated Example

Comparing Parallel and Series-parallel Models

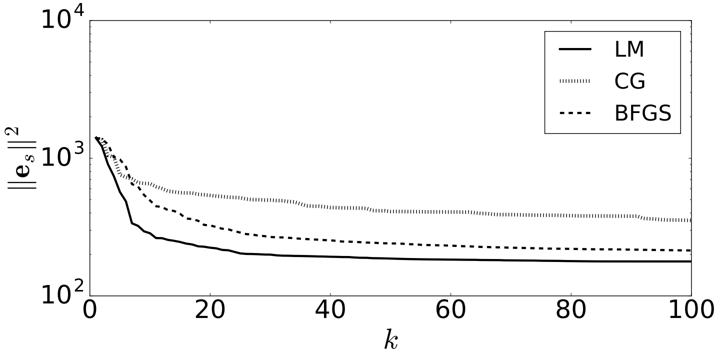


Figure: Sum of squared simulation errors per epoch for:
Levenberg-Marquardt (LM); Conjugate-gradient (CG); and, BFGS

Optimization Methods and Unboundedness

First-Order Linear System

$$\hat{y}[k] = \theta_1 \hat{y}[k-1] + \theta_2 u[k-1]$$

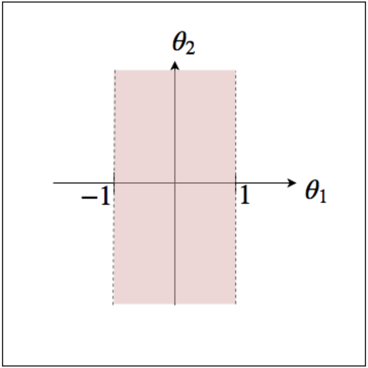


Figure: Set of parameters (θ_1, θ_2) that yield a bounded solution $\hat{y}[k]$.

First-Order Linear System

$$\hat{y}[k] = \theta_1 \hat{y}[k-1] + \theta_2 u[k-1]$$

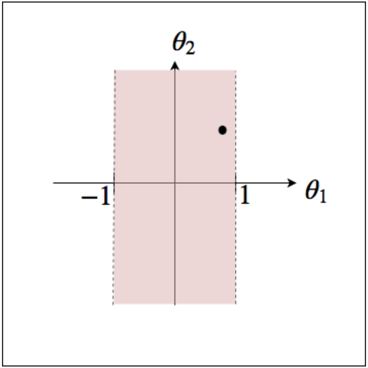


Figure: Set of parameters (θ_1, θ_2) that yield a bounded solution $\hat{y}[k]$.

First-Order Linear System

$$\hat{y}[k] = \theta_1 \hat{y}[k-1] + \theta_2 u[k-1]$$

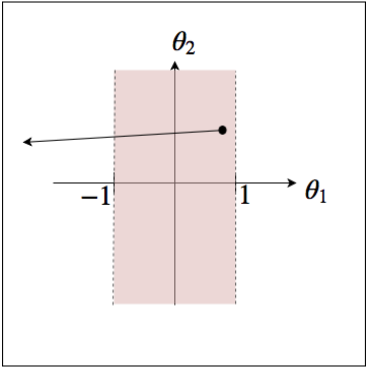


Figure: Set of parameters (θ_1, θ_2) that yield a bounded solution $\hat{y}[k]$.

First-Order Linear System

$$\hat{y}[k] = \theta_1 \hat{y}[k-1] + \theta_2 u[k-1]$$

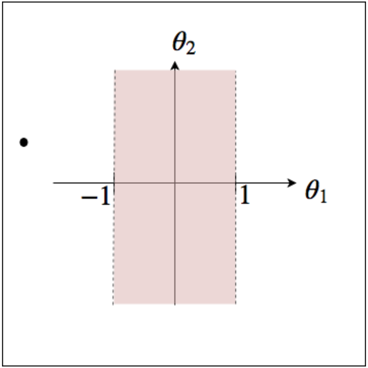


Figure: Set of parameters (θ_1, θ_2) that yield a bounded solution $\hat{y}[k]$.

- Trust-region methods;
- Levenberg-Marquardt;
- Backtrack line search;
- Pattern-Search;

Multiple Shooting

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
    1. Boundary values problems;
    2. ODE parameter estimation;
    3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

Multiple Shooting

- Applications:
  1. Boundary values problems;
  2. ODE parameter estimation;
  3. Optimal control;
- Escape local minima;
- Better numerical stability;
- Can be implemented in parallel.

Ribeiro, A. H., and Aguirre, L. A. (2017)
Shooting methods for Parameter Estimation of Output Error Models.
IFAC world congress (Toulouse, France 2017).

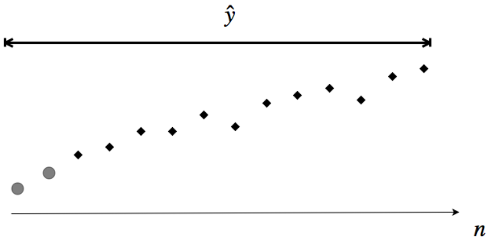


Figure: The initial conditions are represented with circles ○ and subsequent simulated values with diamonds ◇.

Single Shooting

Estimate NOE model solving:

$$\min_{\mathbf{\Theta}} \|\mathbf{e}_s\|^2$$

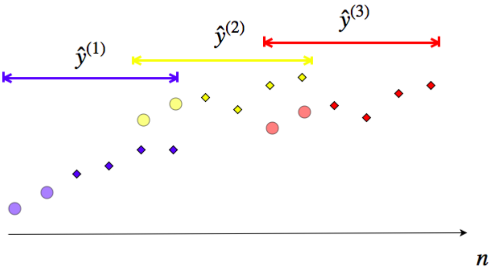


Figure: Three consecutive simulations $\hat{y}^{(i)}$, $i = 1, 2, 3$ are indicated with different colors. The initial conditions are represented with circles $\bigcirc$ and subsequent simulated values with diamonds $\diamond$.

Multiple Shooting

- m_s subdivisions.
- $\hat{y}^{(i)} \Rightarrow$ i-th simulation.
- $\mathbf{e}_s^{(i)} \Rightarrow$ i-th error.
- $\mathbf{e}_{\text{ms}} = \begin{bmatrix} \mathbf{e}_s^{(1)} \\ \vdots \\ \mathbf{e}_s^{(m_s)} \end{bmatrix}$

Multiple Shooting

Shooting Methods for Parameter Estimation of Output Error Models

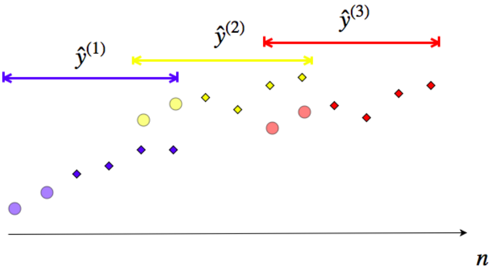


Figure: Three consecutive simulations $\hat{y}^{(i)},\ i = 1, 2, 3$ are indicated with different colors.The initial conditions are represented with circles $\bigcirc$ and subsequent simulated values with diamonds $\diamond$.

Multiple Shooting

- m_s subdivisions.
- $\hat{y}^{(i)} \Rightarrow i$-th simulation.
- $\mathbf{e}_s^{(i)} \Rightarrow i$-th error.
- $\mathbf{e}_{\text{ms}} = \begin{bmatrix} \mathbf{e}_s^{(1)} \\ \vdots \\ \mathbf{e}_s^{(m_s)} \end{bmatrix}$

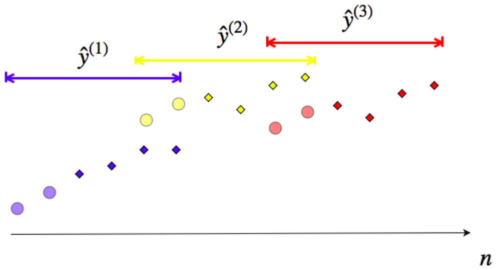


Figure: Three consecutive simulations $\hat{y}^{(i)}, \; i = 1, 2, 3$ are indicated with different colors. The initial conditions are represented with circles $\bigcirc$ and subsequent simulated values with diamonds $\diamond$.

Multiple Shooting

- m_s subdivisions.
- $\hat{y}^{(i)} \Rightarrow$ i-th simulation.
- $\mathbf{e}_s^{(i)} \Rightarrow$ i-th error.
- $\mathbf{e}_{\text{ms}} = \begin{bmatrix} \mathbf{e}_s^{(1)} \\ \vdots \\ \mathbf{e}_s^{(m_s)} \end{bmatrix}$

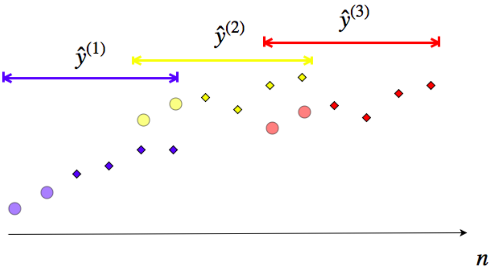


Figure: Three consecutive simulations $\hat{y}^{(i)}$, $i = 1, 2, 3$ are indicated with different colors.The initial conditions are represented with circles ○ and subsequent simulated values with diamonds ◇.

Multiple Shooting

- m_s subdivisions.
- $\hat{y}^{(i)} \Rightarrow i$-th simulation.
- $\mathbf{e}_s^{(i)} \Rightarrow i$-th error.
- $\mathbf{e}_{\mathrm{ms}} = \begin{bmatrix} \mathbf{e}_s^{(1)} \\ \vdots \\ \mathbf{e}_s^{(m_s)} \end{bmatrix}$

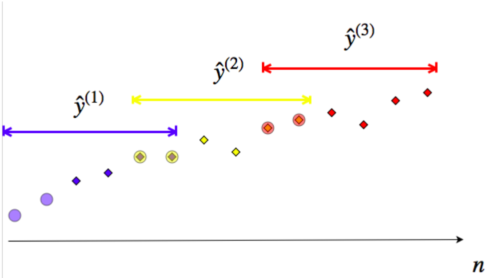
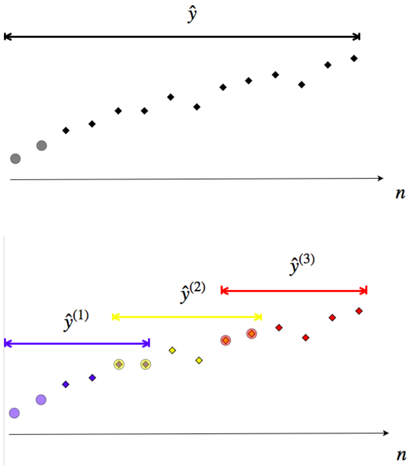


Figure: Three consecutive simulations $\hat{y}^{(i)}, \ i = 1, 2, 3$ are indicated with different colors.The initial conditions are represented with circles $\bigcirc$ and subsequent simulated values with diamonds $\diamond$.

Multiple Shooting

- $\mathbf{e}_{\text{ms}} = \mathbf{e}_s$ if initial conditions matches the previous ones.



Single Shooting

Estimate NOE model solving:

$$\min_{\boldsymbol{\Theta}} \|\mathbf{e}_s\|^2$$

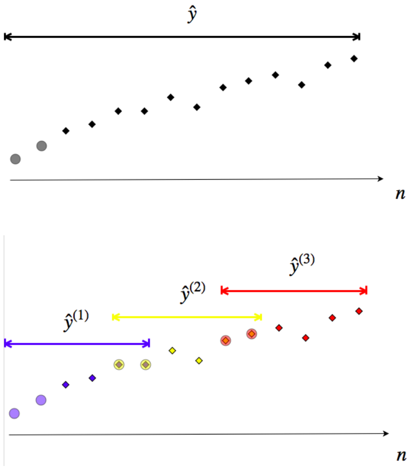
Multiple Shooting

Estimate NOE model solving:

$$\min_{\boldsymbol{\Phi}} \|\mathbf{e}_{\mathrm{ms}}\|^2$$

subject to: $\underline{\hat{\mathbf{y}}}^{(i)}[\mathbf{end}] = \underline{\mathbf{y}}_0^{(i+1)}$

$$i = 1, \cdots, m_s$$



Single Shooting

Parameter $\mathbf{\Theta}$.

Multiple Shooting

Extended parameter $\mathbf{\Phi}$:

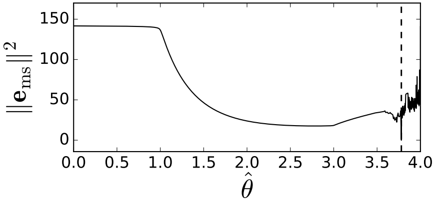
$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Theta} \\ \underline{\mathbf{y}}_0^{(1)} \\ \vdots \\ \underline{\mathbf{y}}_0^{(m_s)} \end{bmatrix}$$

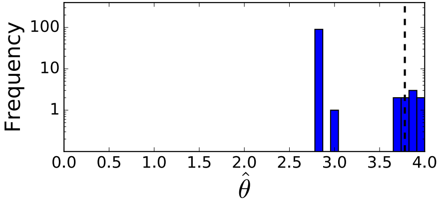
Logistic Map

A dataset with 300 samples were generated using the logistic map:

$$y[k] = \theta y[k-1](1 - y[k-1]),$$

for $\theta = 3.78$.





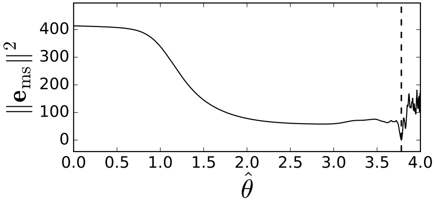
$$m_s = 1$$

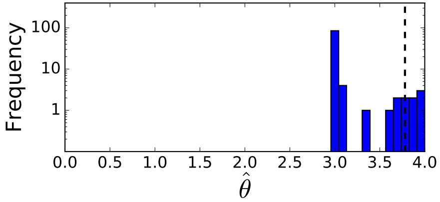
Logistic Map

A dataset with 300 samples were generated using the logistic map:

$$y[k] = \theta y[k-1](1 - y[k-1]),$$

for $\theta = 3.78$.





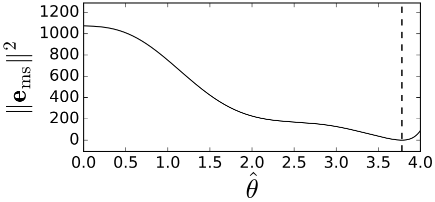
$$m_s = 30$$

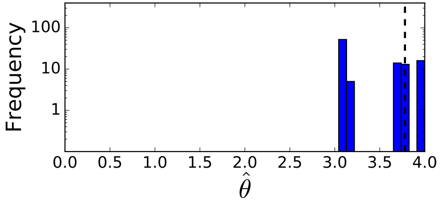
Logistic Map

A dataset with 300 samples were generated using the logistic map:

$$y[k] = \theta y[k-1](1 - y[k-1]),$$

for $\theta = 3.78$.





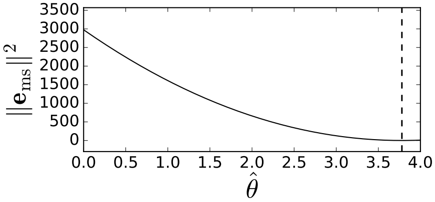
$m_s = 100$

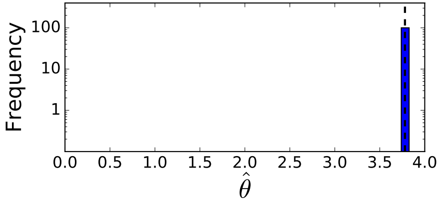
Logistic Map

A dataset with 300 samples were generated using the logistic map:

$$y[k] = \theta y[k-1](1 - y[k-1]),$$

for $\theta = 3.78$.





$m_s = 300$

Conclusion

Future Work

- Penalty method $\Rightarrow$ Byrd-Omojokun SQP method;
- Structure selection procedure using l^1 regularization:

$$\min_{\mathbf{\Theta}} \|\mathbf{e}\|_2^2 + \mu\|\mathbf{\Theta}\|_1$$

- And its application to multiple shooting:

$$\begin{aligned} \min_{\mathbf{\Phi}} \quad & \tfrac{1}{2}\|\mathbf{e}_{\text{ms}}\|^2 + \mu\|\mathbf{\Theta}\|_1 \\ \text{subject to:} \quad & \hat{\underline{\mathbf{y}}}^{(i)}[\text{end}] = \underline{\mathbf{y}}_0^{(i+1)}, \; i = 1, \cdots, m_s - 1. \end{aligned}$$

Future Work

- Penalty method $\Rightarrow$ Byrd-Omojokun SQP method;
- Structure selection procedure using l^1 regularization:

$$\min_{\mathbf{\Theta}} \|\mathbf{e}\|_2^2 + \mu \|\mathbf{\Theta}\|_1$$

- And its application to multiple shooting:

$$\begin{aligned} \min_{\mathbf{\Phi}} \quad & \frac{1}{2}\|\mathbf{e}_{\text{ms}}\|^2 + \mu \|\mathbf{\Theta}\|_1 \\ \text{subject to:} \quad & \underline{\hat{\mathbf{y}}}^{(i)}[\text{end}] = \underline{\mathbf{y}}_0^{(i+1)}, \;\; i = 1, \cdots, m_s - 1. \end{aligned}$$

Future Work

- Penalty method $\Rightarrow$ Byrd-Omojokun SQP method;
- Structure selection procedure using l^1 regularization:

$$\min_{\mathbf{\Theta}} \|\mathbf{e}\|_2^2 + \mu\|\mathbf{\Theta}\|_1$$

- And its application to multiple shooting:

$$\begin{aligned} \min_{\mathbf{\Phi}} \quad & \frac{1}{2}\|\mathbf{e}_{\text{ms}}\|^2 + \mu\|\mathbf{\Theta}\|_1 \\ \text{subject to:} \quad & \underline{\hat{\mathbf{y}}}^{(i)}[\text{end}] = \underline{\mathbf{y}}_0^{(i+1)}, \;\; i = 1, \cdots, m_s - 1. \end{aligned}$$

The End