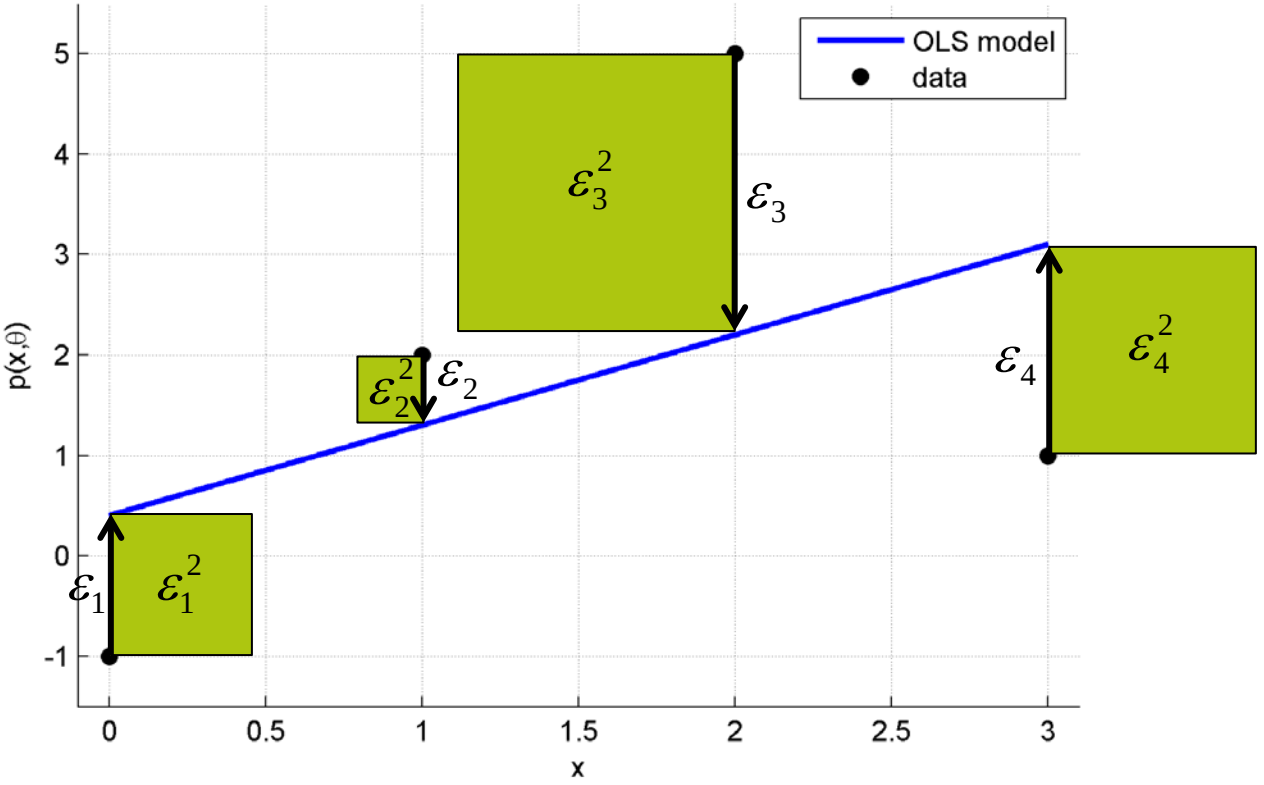
# Lecture 4: Parameter Estimation

## AE4320 System Identification of Aerospace Vehicles

Dr.ir. Coen de Visser
Department of Control & Simulation



# Course Outline

- **Lecture 1: (dr.ir. Coen de Visser)**
  - Course goals and objectives
  - Introduction to System Identification
- **Lecture 2,3: (dr.ir. Daan Pool)**
  - System Identification Experiments
- **Lecture 4,5,6: (dr.ir. Daan Pool)**
  - Kalman filters
  - State estimation & Sensor Fusion
- **Lecture 7,8: (dr.ir. Coen de Visser)**
  - Model structure selection
  - Model parameter estimation

# Course Outline

- **Lecture 9: (dr.ir. Coen de Visser)**
  - Advanced identification approach: Neural networks
- **Lecture 10,11: (dr.ir. Coen de Visser)**
  - Advanced identification approach: Multivariate B-Splines
- **Lecture 12: (dr.ir. Coen de Visser)**
  - Model validation, course conclusion

# Goals of this Lecture

***Questions that will be answered during this lecture:***

1. *What is parameter estimation, and why do we need it?*
2. *What types of model structures do we use?*
3. *What is an estimator?*
4. *What is linear regression?*
5. *What are the Equation Error, Filter Error, and Output Error approaches to parameter estimation?*
6. *What are the assumptions made for the OLS, WLS, GLS, TLS, RLS, and MLE parameter estimators?*
7. *How do we obtain the optimal set of parameters given a particular parameter estimator?*

# SysID High Level Overview

Where we are now in the System Identification Cycle:

**Experiment phase**

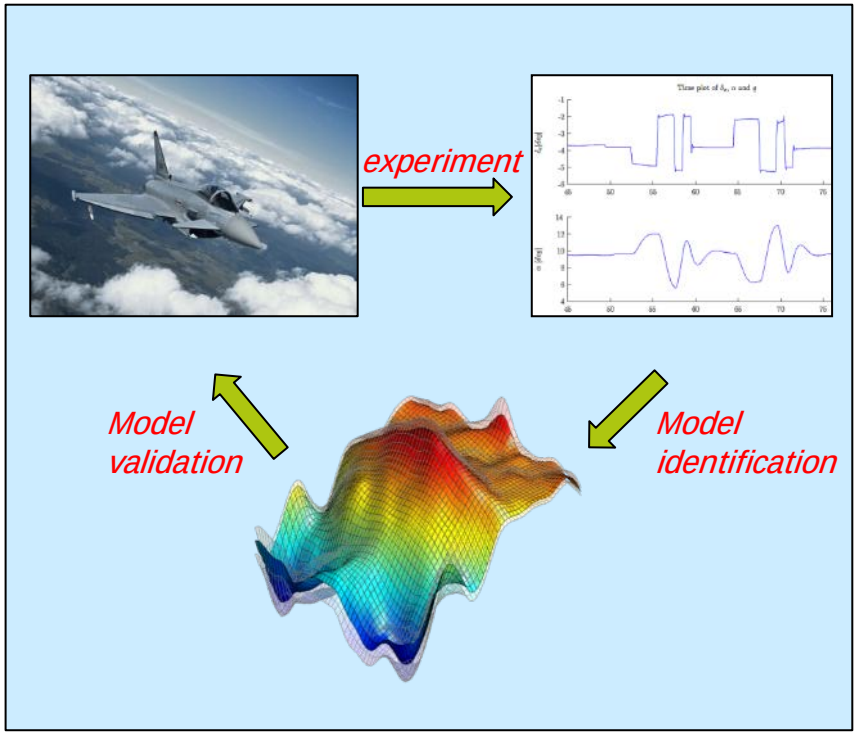
- Plant analysis
- Experiment design and execution
- Data logging and pre-processing

**Model identification phase**

- State estimation
- Model structure definition
- Parameter estimation

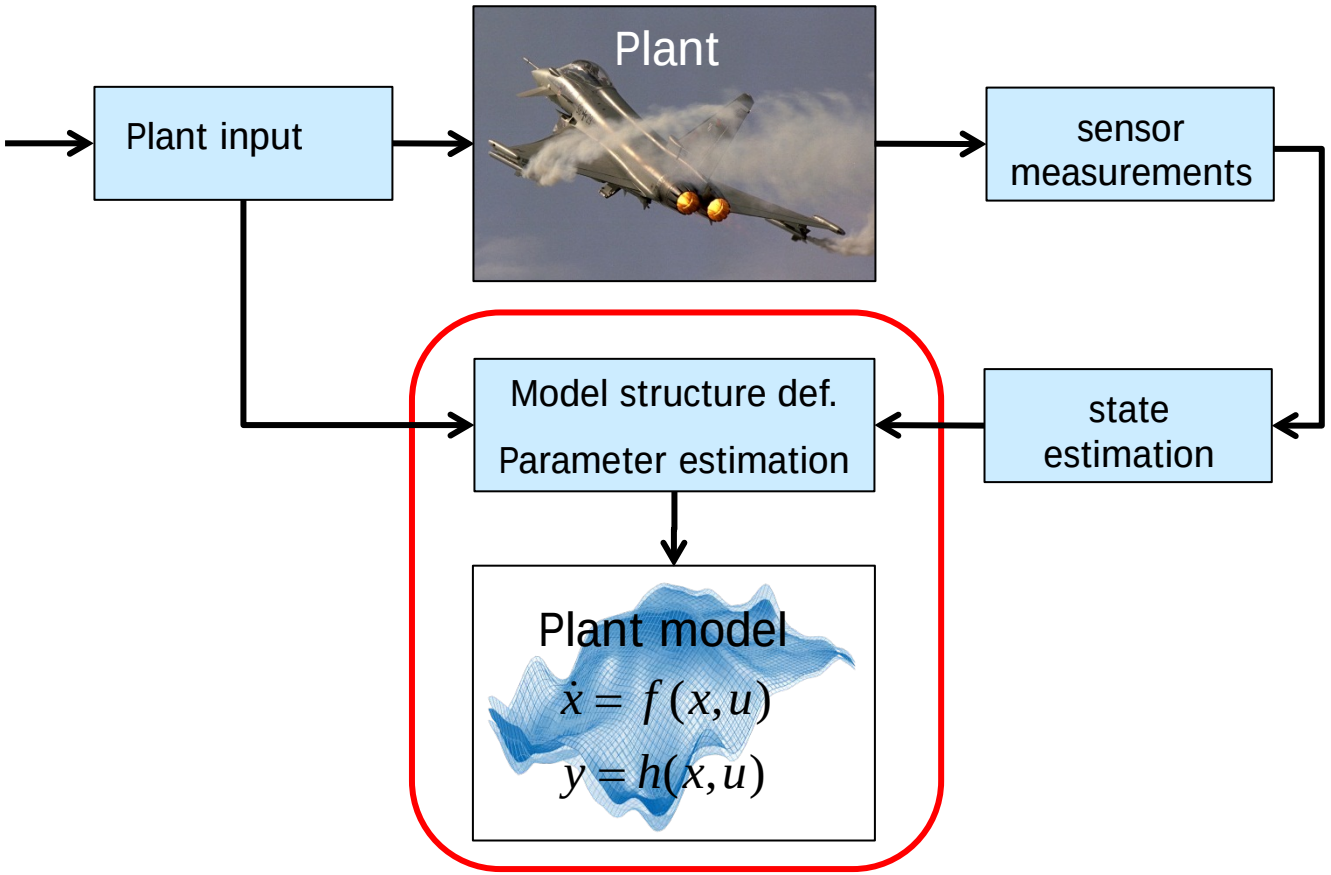
**Model validation phase**

- Model validation



# Parameter Estimation: Introduction

The focus is now on the modelling part of the SysID procedure...



# Parameter Estimation: Introduction

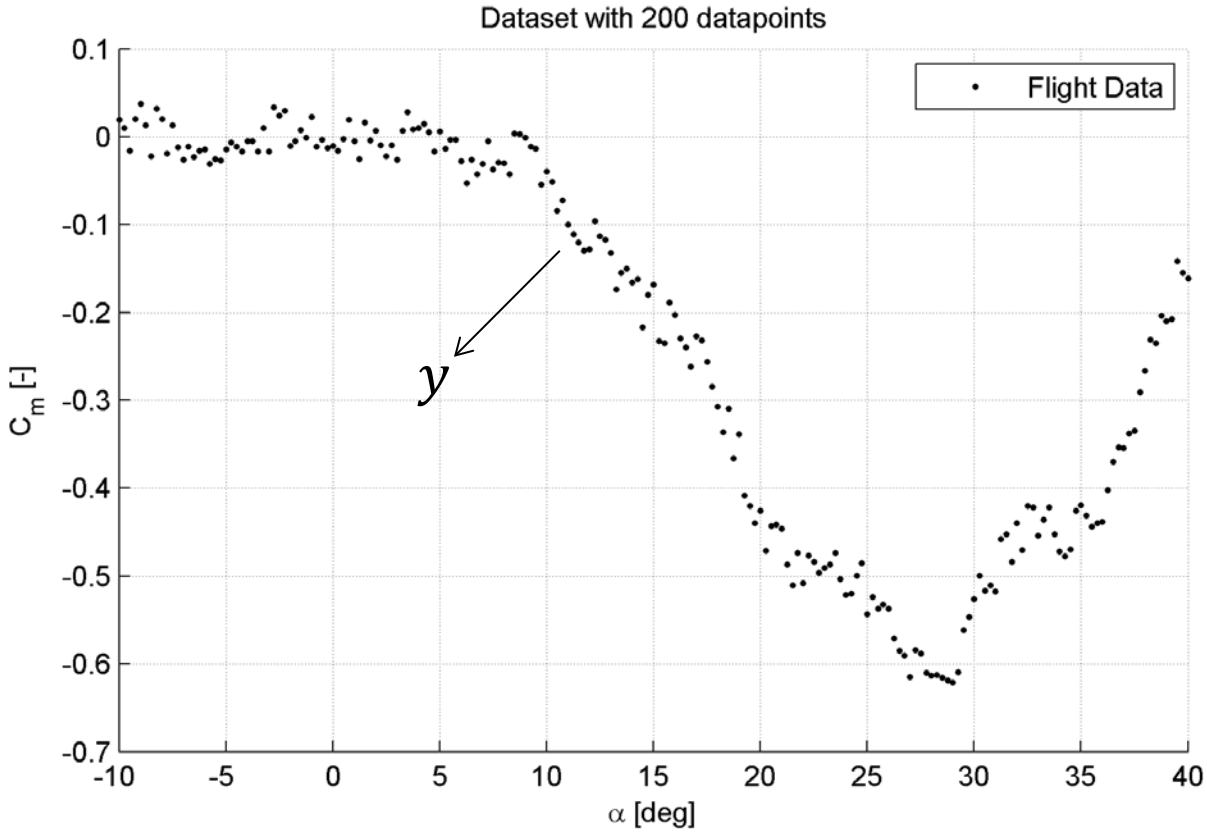
## Some facts on Parameter Estimation

- Parameter estimation is the act of determining an optimal set of model parameters given a set of measurements.
- **Parameter estimation is absolutely essential in any modeling application!**
- Parameter estimation is used in virtually **all fields** of science and engineering, from econometrics to astrophysics, and from the social sciences to aerospace engineering.
- There are a very large number of different PE techniques; each technique has its advantages and disadvantages.
- Confusingly, PE is sometimes called 'learning'.
- The first rigorous PE technique, ordinary least squares, was invented by Carl Friedrich Gauss in 1795.

# Parameter Estimation: Introduction

## The definition of parameter estimation

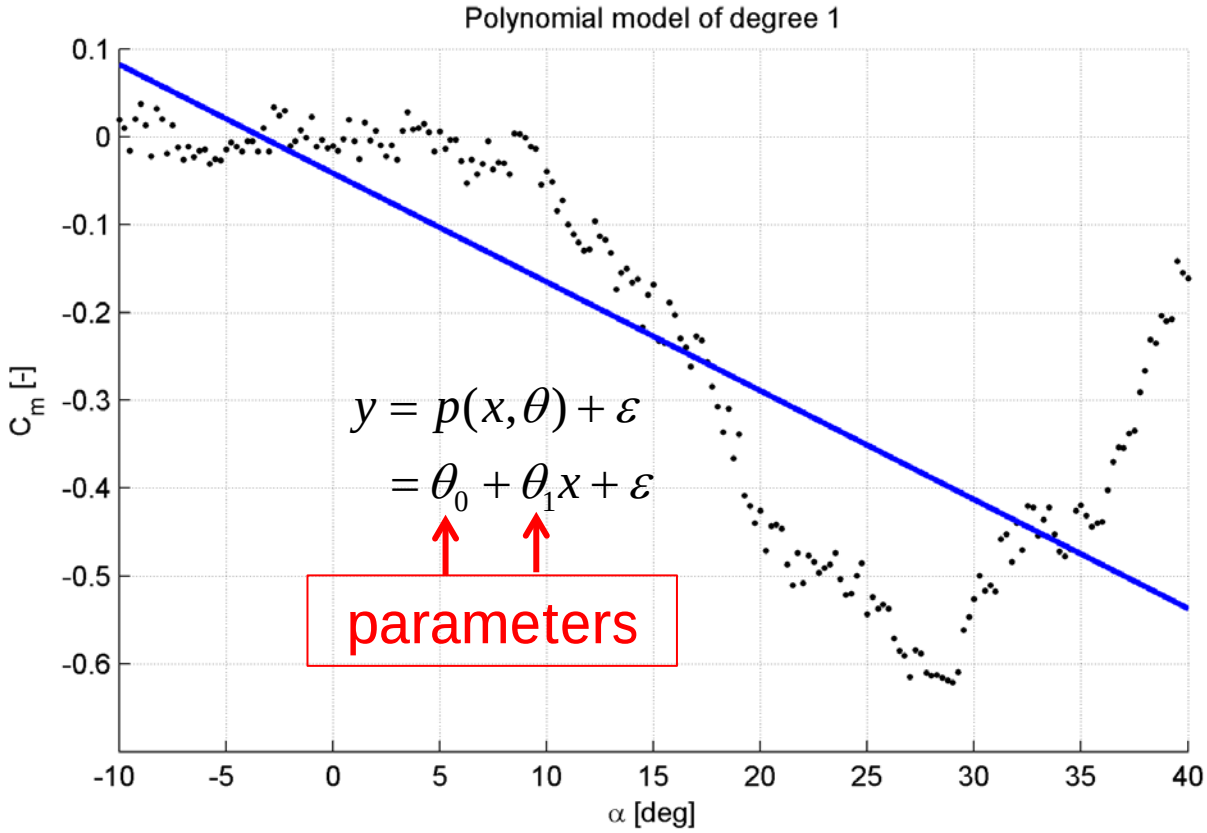
Let $y$ be a vector of observations made on some system:



# Parameter Estimation: Introduction

## The definition of parameter estimation

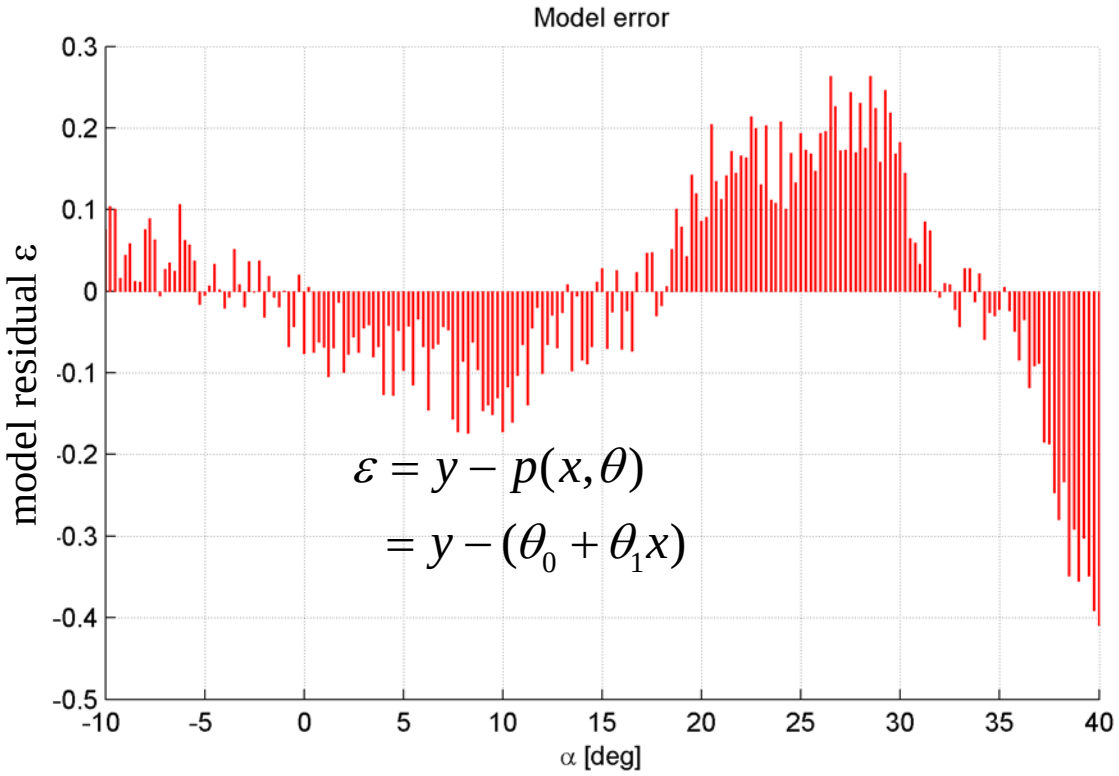
Let $p(x,\theta)$ be a model for $y$. Let's assume a linear **model structure**:



# Parameter Estimation: Introduction

## The definition of parameter estimation

The goal of parameter estimation is to find a set of parameters $\theta$ that *in some way* minimizes the model residuals $\varepsilon = y - p(x,\theta)$ :



# Parameter Estimation: Introduction

## The definition of parameter estimation

Let $y$ be the observations on a system. Let $p(x,\theta)$ be a model for $y$ in terms of the **known** states $x$, and the **unknown** parameters $\theta$.

Let the error (residual) $\varepsilon$ between the model $p(x,\theta)$ and the measurement $y$ be defined as follows:

$$\varepsilon = y - p(x,\theta)$$

The **parameter estimation problem** is then to determine the values for $\theta$ that in some way minimizes the modeling error:

$$\hat{\theta} = \arg\min J\left(\varepsilon\right) = \arg\min J\left(y - p(x,\theta)\right)$$

with $J(\varepsilon)$ a **cost function**, and with $\hat{\theta}$ the **estimator** for $\theta$.

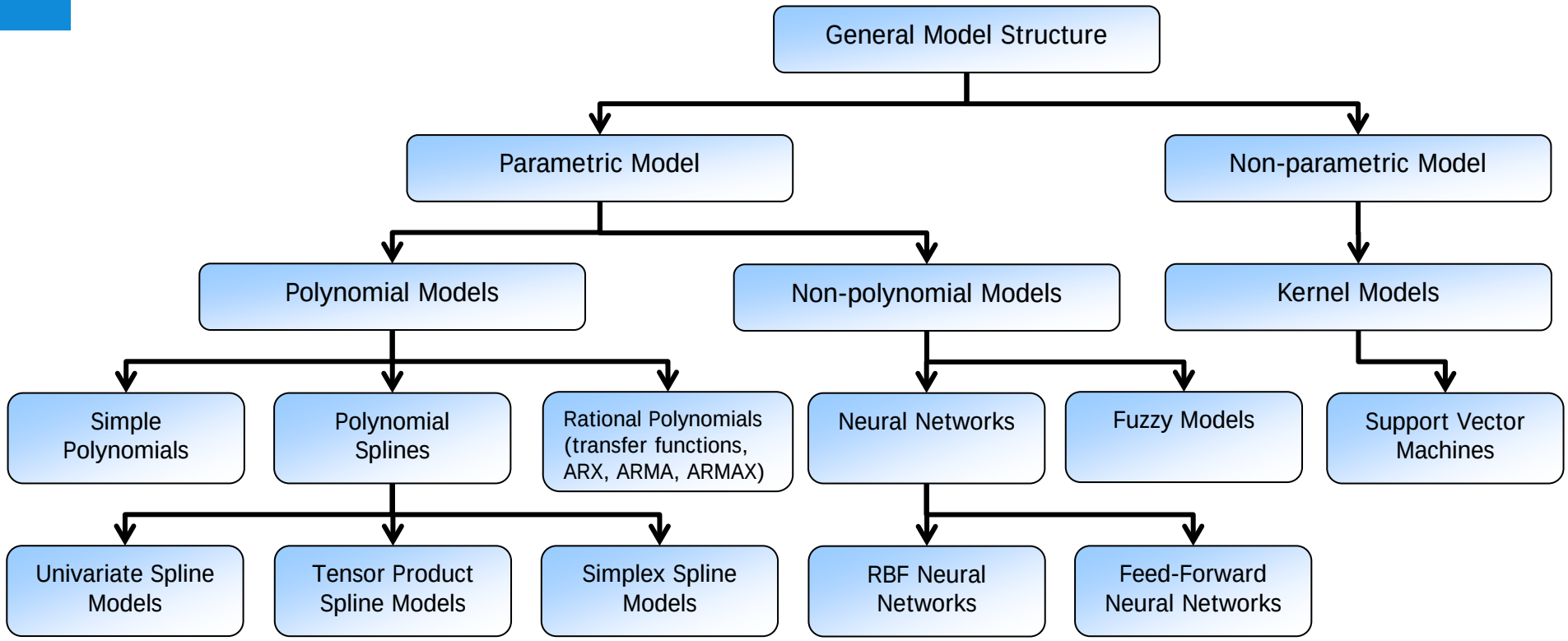
# Parameter Estimation: Introduction

## Parameter estimation problem definition

1. Data analysis: do we have any a-priori knowledge on the data?
   - Any a-priori knowledge on the system can ***and should*** be utilized.
2. Model structure selection: what type of function is $p(x, \theta)$?
   - A model structure must be defined before parameter estimation can commence. There are many possible model structures, e.g. polynomial models, multivariate spline models, neural networks, kernel functions.
3. Estimation of model parameters: how do we minimize the model error?
   - A parameter estimator must be defined which can be used to estimate the parameters of the chosen model structure. There are many possible estimators, e.g. **OLS**, **WLS**, **GLS**, **TLS**, **MLE**.
4. Validation of estimator performance: is $p(x, \theta)$ is a good model?
   - The performance of the estimator must be validated by analyzing model residuals and parameter estimation statistics.

# Model Structure Selection

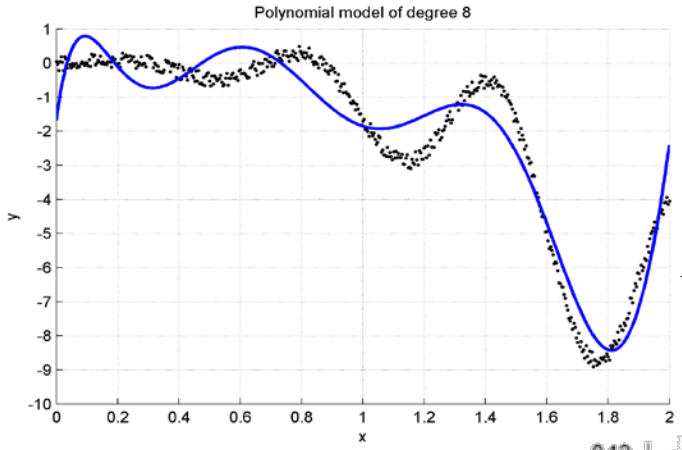
## Classification of (some) model structures



# Model Structure Selection

## Classification of model structures

- **Simple Polynomials**
- Rational Polynomials
- Univariate Splines
- Tensor Product Spline Models
- Simplex Spline Models
- Feed-Forward Neural Networks
- RBF Neural Networks



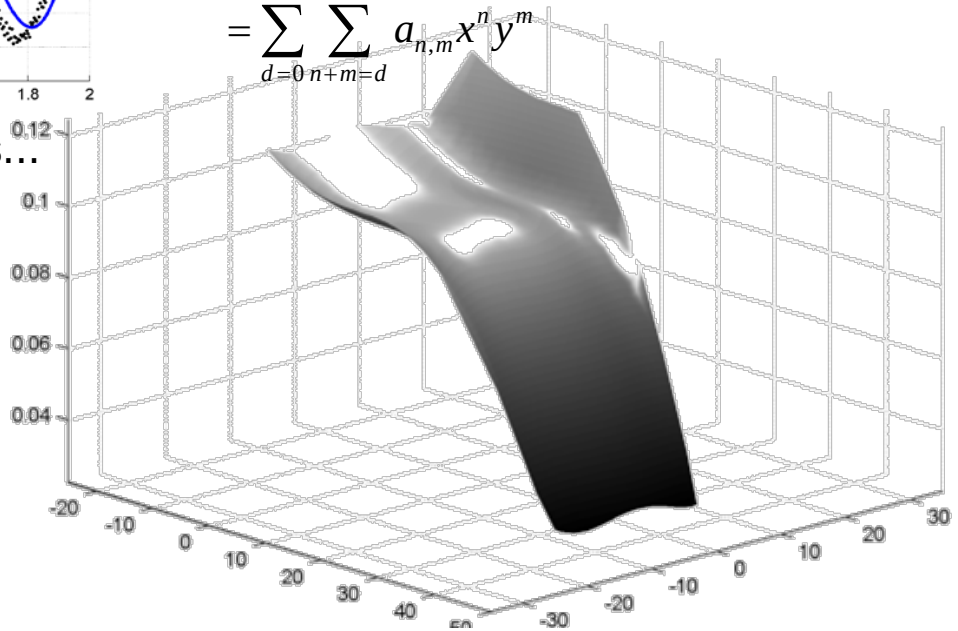

Univariate polynomial models...

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$$

Multivariate polynomial models...

$$p(x,y) = a_0 + a_{1,0}x + a_{0,1}y + a_{2,0}x^2 + a_{1,1}xy + a_{0,2}y^2 + \cdots$$

$$= \sum_{d=0}^{M} \sum_{n+m=d} a_{n,m} x^n y^m$$



# Model Structure Selection

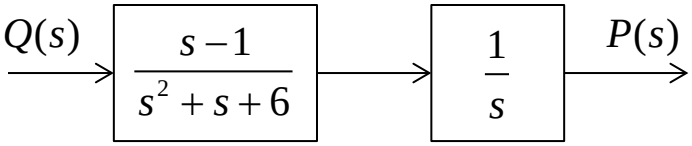
## Classification of model structures

- Simple Polynomials
- **Rational Polynomials**
- Univariate Splines
- Tensor Product Spline Models
- Simplex Spline Models
- Feed-Forward Neural Networks
- RBF Neural Networks

Rational polynomials:

$$r(x) = \frac{p(x)}{q(x)} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d}{b_0 + b_1 x + b_2 x^2 + \cdots + b_m x^m}$$

Example: Transfer functions (Laplace domain!)

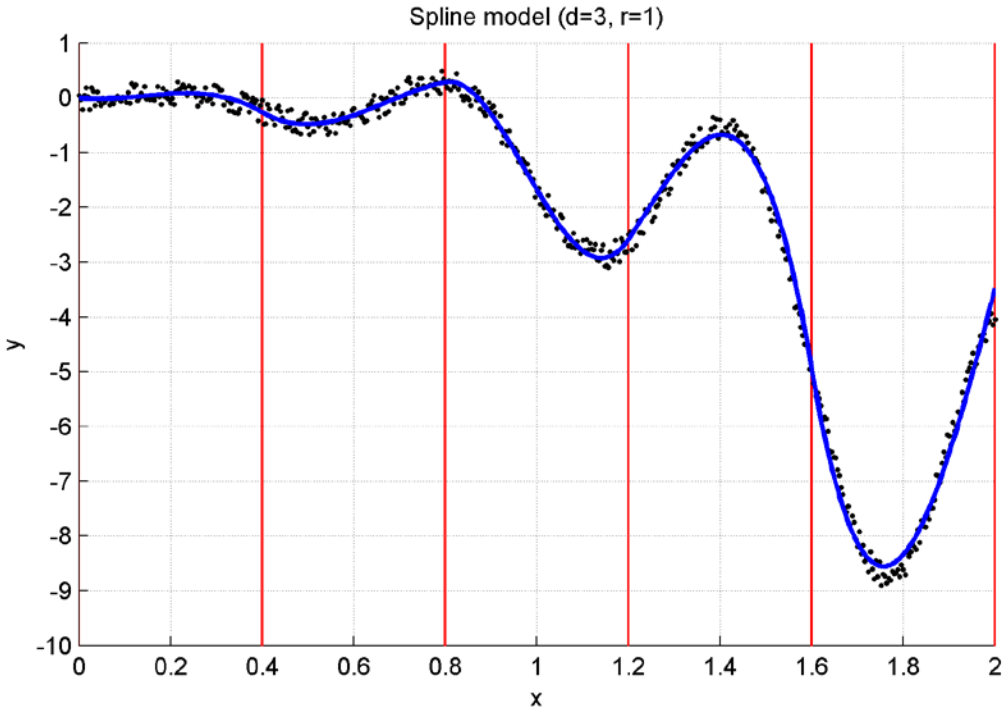



$$R(s) = \frac{P(s)}{Q(s)} = \frac{-1+s}{s\left(s^2+s+6\right)} = \frac{-1+s}{6s+s^2+s^3}$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- **Univariate Splines**
- Tensor Product Splines Models
- Simplex Spline Models
- Feed-Forward Neural Networks
- RBF Neural Networks



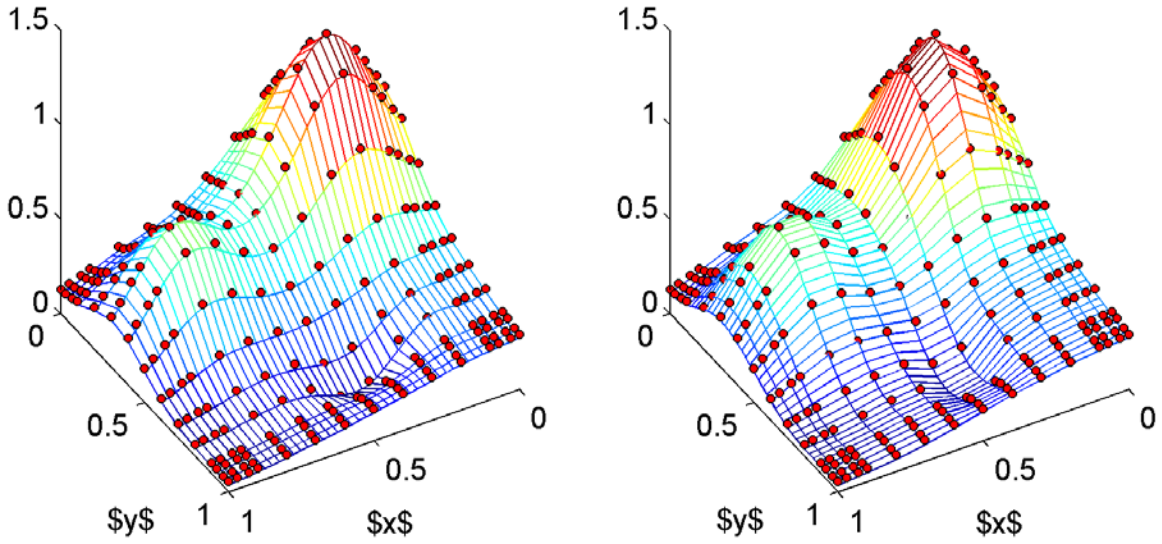

Univariate spline models...

$$p(x) = B^d(x) \cdot c$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- Univariate Splines
- **Tensor Product Spline Models**
- Simplex Spline Models
- Feed-Forward Neural Networks
- RBF Neural Networks



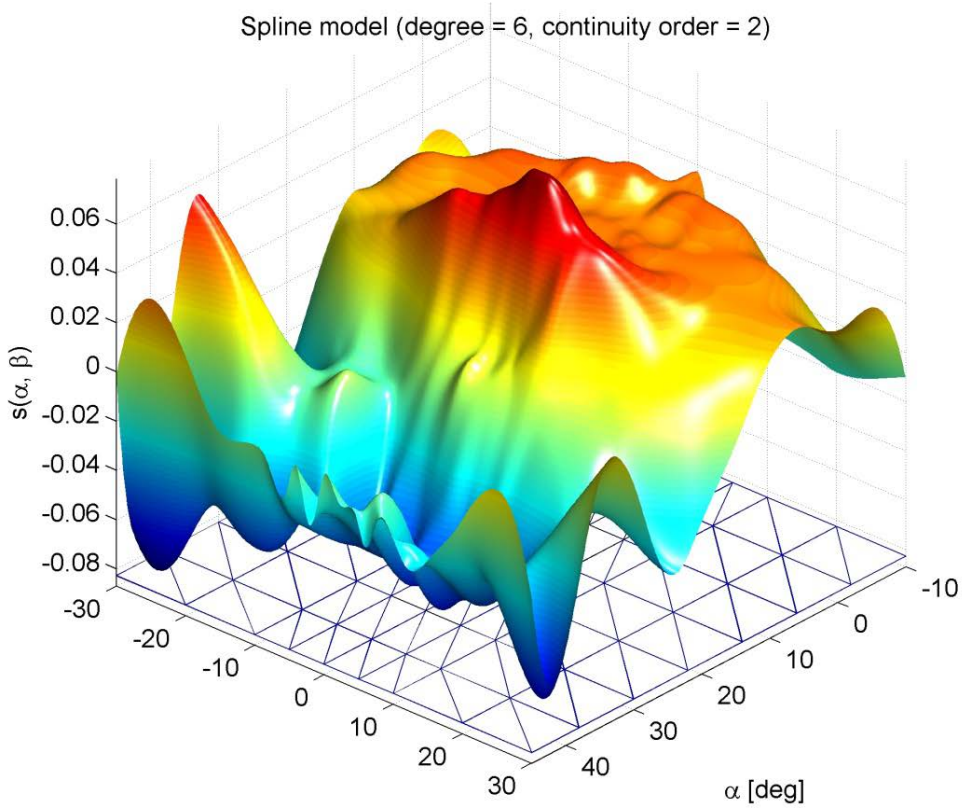

Multivariate tensor-product splines...

$$p(x,y) = \sum_{i=1}^{r}\sum_{j=1}^{s} B(x|t_i,\ldots,t_{i+v})B(y|t_j,\ldots,t_{j+w})c_{ij}$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- Univariate Splines
- Tensor Product Spline Models
- **Simplex Spline Models**
- Feed-Forward Neural Networks
- RBF Neural Networks



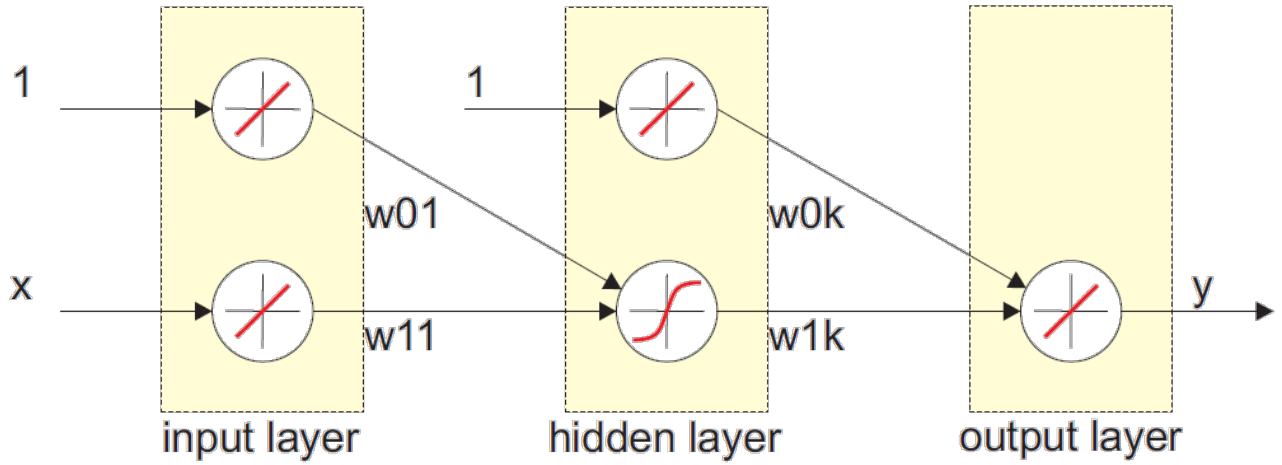

Multivariate simplex spline models

$$p(x,y) = B^d(x,y) \cdot c$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- Univariate Splines
- Tensor Product Spline Models
- Simplex Spline Models
- **Feed-Forward Neural Networks**
- RBF Neural Networks



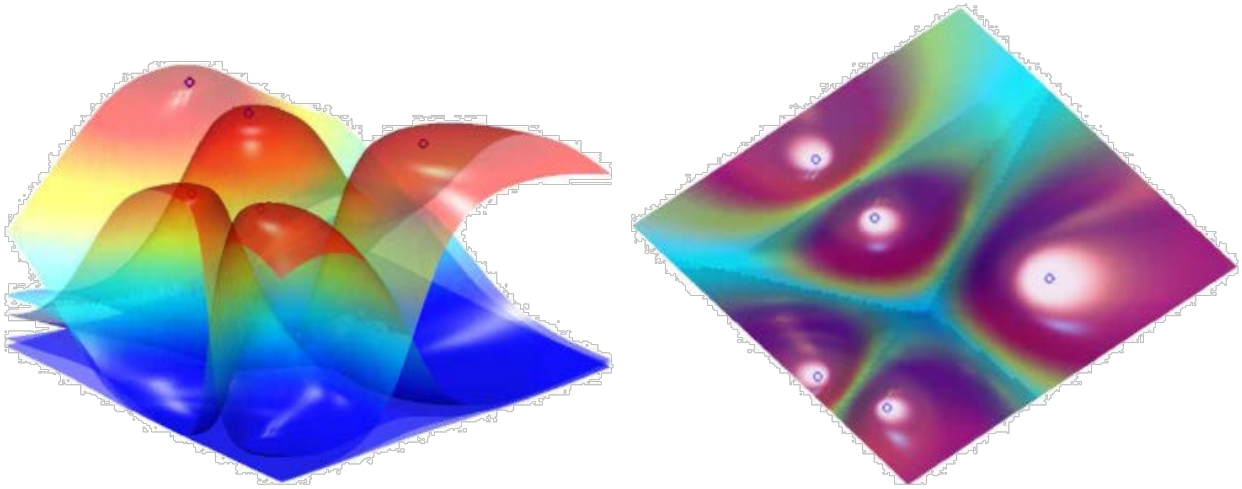

Feed-forward neural networks

$$p_k(x) = \sum_j w_{jk} \phi_j \left( \sum_i w_{ij} x_i \right)$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- Univariate Splines
- Tensor Product Spline Models
- Simplex Spline Models
- Feed-Forward Neural Networks
- **RBF Neural Networks**



Radial Basis Function neural networks...

$$p(x,y) = \sum_{i=1}^{N} c_i \exp\left[\left(v_i\left(x-a\right)\right)^2\left(w_i\left(y-b\right)\right)^2\right]$$

# Model Structure Selection

## Classification of model structures

- Simple Polynomials
- Rational Polynomials
- Univariate Splines
- Tensor Product Spline Models
- Simplex Spline Models
- Feed-Forward Neural Networks
- RBF Neural Networks

**However…**

Many other types of (mixed) model structures exist, such as the structure for pilot model identification:

$$H_p(j\omega) = 1 \times K_p \frac{T_L j\omega + 1}{T_I j\omega + 1} e^{-j\omega\tau_p} \times \frac{\omega_{nm}^2}{(j\omega)^2 + 2\xi_{nm}\omega_{nm} j\omega + \omega_{nm}^2}$$

# Model Structure Selection

## Linear-in-the-parameter functions

An important property that sets certain model structures apart from others is the linear-in-the-parameter property.

**Definition 4.2: Linear-in-the-parameter functions**
A function $p(x,\theta)$ is called linear-in-the-parameters if the first derivative of $p(x,\theta)$ with respect to $\theta$ is a function of only $x$:

$$\frac{\partial p(x,\theta)}{\partial \theta} = f(x)$$

All linear-in-the-parameter functions can be written in Matrix-vector form:

$$p(x,\theta) = A(x)\cdot\theta$$

Optimization problems for linear-in-the-parameter models can be solved using simple linear solvers, which in general are **easier to implement** and have a **lower computational complexity**.

# Parameter Estimation: Linear Regression

## Linear Regression

- The simplest method for parameter estimation is linear regression.
- It is also the most used form!
- Linear regression problems can be solved with **efficient linear solvers**.
- All linear least squares methods are linear regression methods.
- **Linear regression requires a linear-in-the-parameters model structure!**

# Parameter Estimation: Linear Regression

## Linear Regression

Define the linear regression model that best fits a sequence of measurements $y$ as follows:

$$y = A(x) \cdot \theta + \varepsilon$$

with

$y =$ the $N \times 1$ measurement vector

$A(x) =$ the $N \times n$ regression matrix

$x =$ the $m \times 1$ state vector

$\theta =$ the $n \times 1$ parameter vector

$\varepsilon =$ the $N \times 1$ model residual vector

# Parameter Estimation: Linear Regression

## Linear Regression: Regression matrix

For $N$ data points, the regression matrix is constructed as follows:

$$A(x) = \begin{bmatrix} 1 & p_1(x(1)) & \cdots & p_M(x(1)) \\ 1 & p_1(x(2)) & \cdots & p_M(x(2)) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & p_1(x(N)) & \cdots & p_M(x(N)) \end{bmatrix}$$

data points

regressor terms
(model order)

with $p_m(x(n))$ a function (e.g. polynomial) in terms of $x$.

# Parameter Estimation: Linear Regression

## Linear Regression: Regression matrix

The most widely used regression matrix structure is polynomial. In that case $p_m(x(n))$ are ordinary polynomials in terms of $x$:

$$A(x) = \begin{bmatrix} 1 & x_i(1) & x_i^2(1) & \cdots & x_i^n(2)x_j^m(1) & \cdots & x_k^M(1) \\ 1 & x_i(2) & x_i^2(2) & \cdots & x_i^n(2)x_j^m(2) & \cdots & x_k^M(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_i(N) & x_i^2(N) & \cdots & x_i^n(N)x_j^m(N) & \cdots & x_k^M(N) \end{bmatrix}$$

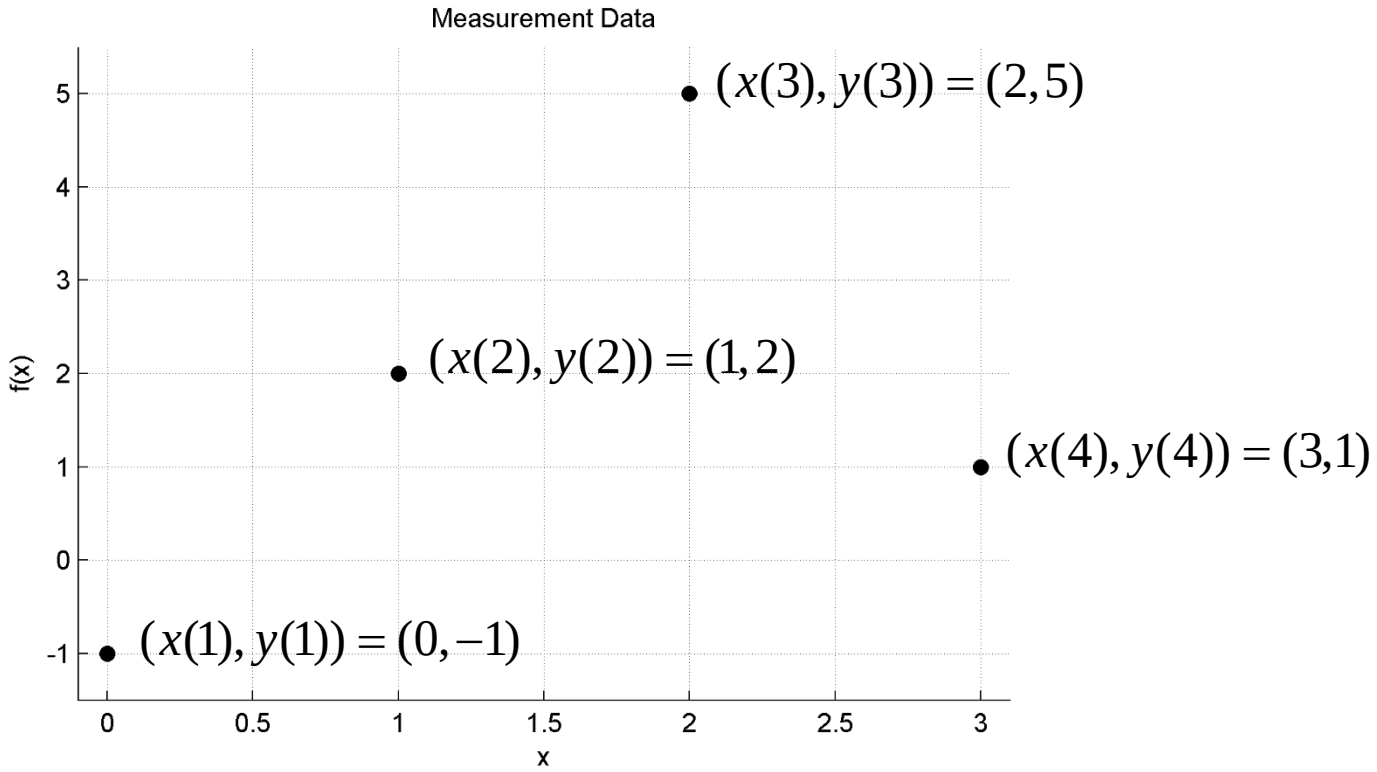
don't forget the cross-terms for multivariate models!

Notice that $A(x)$ can easily get very large; for example, an **M**=6th order **12-D** (e.g. full-state aircraft model structure) with **N**=1.000.000 data (full-flight data recording) points contains 18.5 Billion elements and takes up 149GB in memory!

# Parameter Estimation: Linear Regression

## Linear Regression

**Example 4.1:** Creation of a linear regression model



# Parameter Estimation: Linear Regression

## Linear Regression

**Example 4.1:** Creation of a linear regression model

Let $y$ be a set of observations of an unknown system which we want to model using the linear regression model $A(x)\theta$:

$$y = A(x) \cdot \theta + \varepsilon$$

We have made 4 measurements on the output and the states of the system:

measurement 1: $(x(1), y(1)) = (0, -1)$
measurement 2: $(x(2), y(2)) = (1, 2)$
measurement 3: $(x(3), y(3)) = (2, 5)$
measurement 4: $(x(4), y(4)) = (3, 1)$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Parameter Estimation: Linear Regression

## Linear Regression

**Example 4.1:** Creation of a linear regression model

We assume a quadratic polynomial model structure for $A(x)\theta$:

$$A(x)\cdot\theta = \theta_0 + \theta_1 x + \theta_2 x^2$$

with $\theta_0, \theta_1, \theta_2$ the unknown **model parameters**.

The linear regression model is:

$$y = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

# Parameter Estimation: Linear Regression

## Linear Regression

**Example 4.1:** Creation of a linear regression model

If we substitute the measurement data in the linear regression model:

$$y = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

we get:

$$\begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Parameter Estimation: Linear Regression

## Linear Regression

The linear regression **residual** is:

$$\varepsilon = y - A(x) \cdot \theta$$

The linear regression **estimator** is:

$$\begin{aligned} \hat{\theta} &= \arg\min J\left(y - A(x) \cdot \theta\right) \\ &= \arg\min\ J(\varepsilon) \end{aligned}$$

with $J$ a **cost function**.

**So how do we choose $J$?**

# Parameter Estimation: Linear Regression

Some possible choices of cost functions are:

$$J(x,\theta) = \sum_{i=1}^{N} \varepsilon_i = \sum_{i=1}^{N} y_i - A(x_i)\cdot\theta, \qquad \text{(linear)}$$

$$J(x,\theta) = \sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \left(y_i - A(x_i)\cdot\theta\right)^2, \qquad \text{(quadratic)}$$

$$J(x,\theta) = \sum_{i=1}^{N} \left|\varepsilon_i\right| = \sum_{i=1}^{N} \left|y_i - A(x_i)\cdot\theta\right|, \qquad \text{(absolute value)}$$

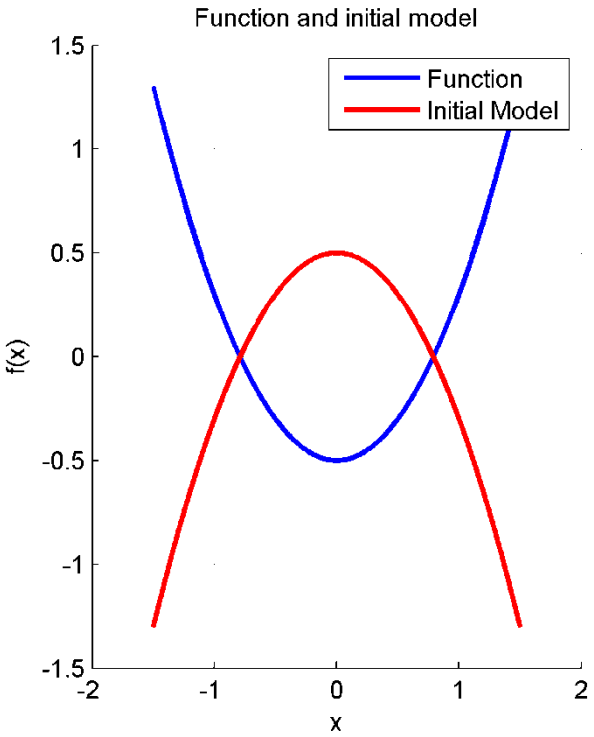
$$J(x,\theta) = \sum_{i=1}^{N} W_{ii}\varepsilon_i^2 = \sum_{i=1}^{N} \left(y_i - A(x_i)\cdot\theta\right)\cdot W \cdot \left(y_i - A(x_i)\cdot\theta\right), \qquad \text{(weighted quadratic)}$$

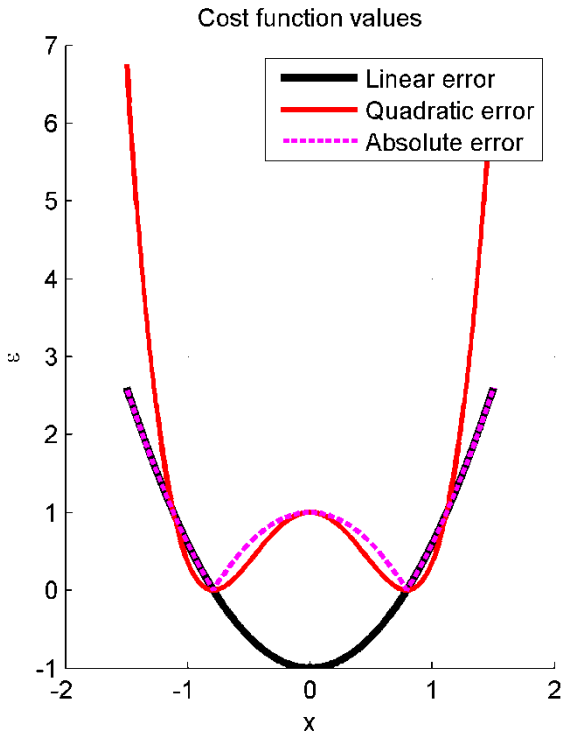
So how do we choose a cost function? Which ones will work, and which ones won't?

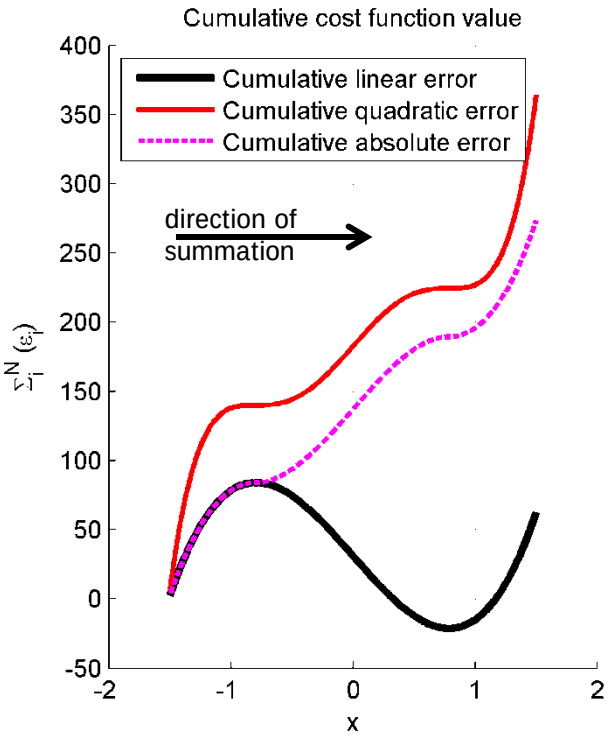
# Parameter Estimation: Linear Regression

Cost function comparison:

$$J(x,\theta)=\sum_{i=1}^{N}\left(y_i-A(x_i)\cdot\theta\right),\qquad J(x,\theta)=\sum_{i=1}^{N}\left(y_i-A(x_i)\cdot\theta\right)^2,\qquad J(x,\theta)=\sum_{i=1}^{N}\left|y_i-A(x_i)\cdot\theta\right|$$

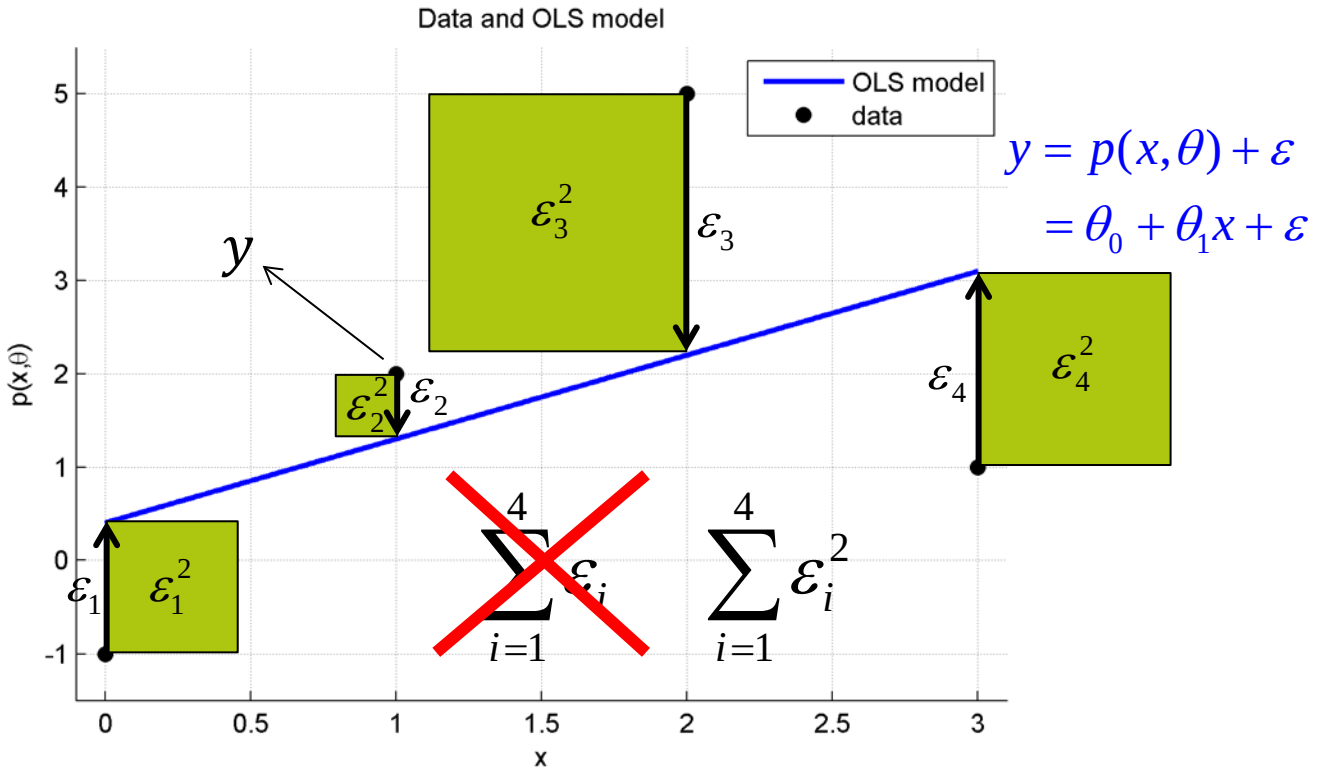


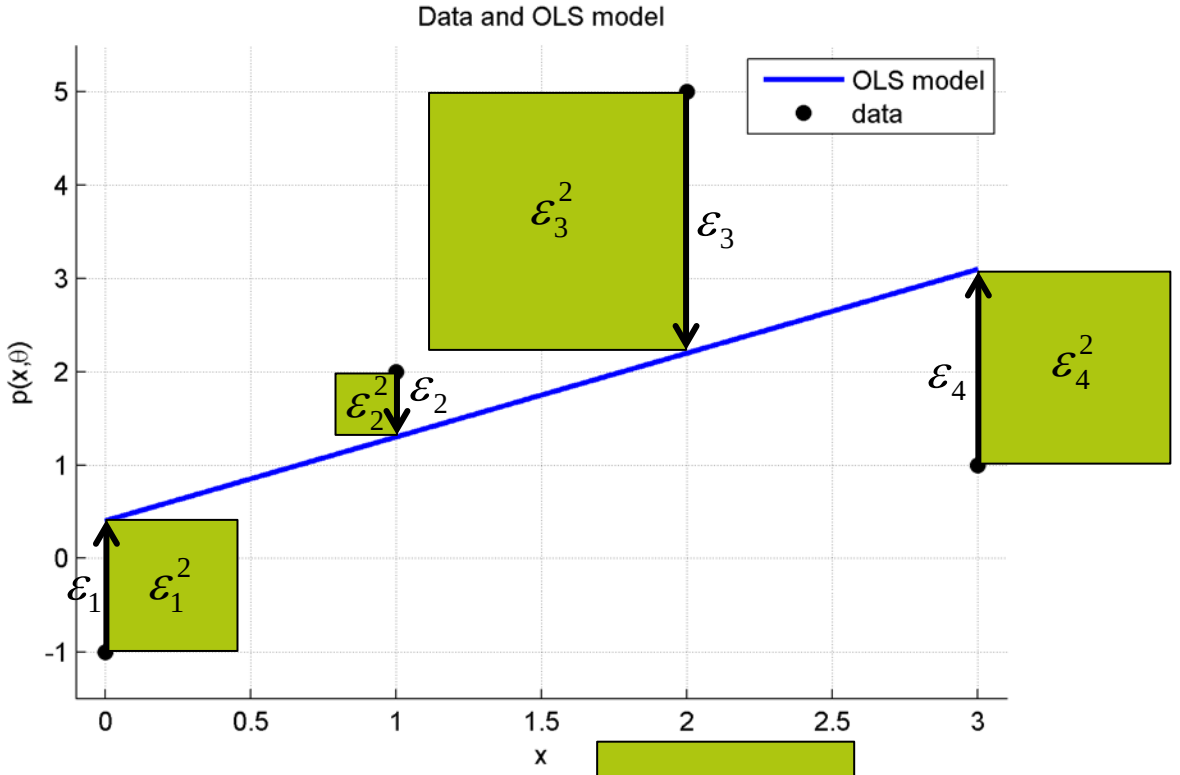



# Least Squares Parameter Estimation

Instead of minimizing the errors directly, we minimize the squares of the errors...



# Least Squares Parameter Estimation




$$\sum_{i=1}^{4} \varepsilon_i^2 = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2$$

# Least Squares Parameter Estimation

## Quadratic cost function leads to least squares solution...

We will use the **convex** quadratic cost function:

$$J(x,\theta) = \sum_{i=1}^{N} \varepsilon_i^{\,2} = \sum_{i=1}^{N} \left( y_i - A(x_i) \cdot \theta \right)^2$$

using matrix notation, this becomes:

$$J(x,\theta) = \varepsilon^T \varepsilon = \left( y - A(x) \cdot \theta \right)^T \left( y - A(x) \cdot \theta \right)$$

The least squares estimator is the solution to the optimization problem

$$\hat{\theta} = \arg\min \varepsilon^T \varepsilon$$

This **convex** cost function has a minimum when

$$\frac{\partial J(x,\theta)}{\partial \theta} = \frac{\partial \varepsilon^T \varepsilon}{\partial \theta} = 0$$

# Least Squares Parameter Estimation

The partial derivative w.r.t. $\theta$ of the quadratic cost function is:

$$\frac{\partial \varepsilon^T \varepsilon}{\partial \theta} = 2\varepsilon^T \frac{\partial \varepsilon}{\partial \theta}$$

$$= 2\left(y - A(x)\cdot\theta\right)^T \cdot -A(x) = 0$$

introduce $\hat{\theta}$ as the particular value for $\theta$ that minimizes the cost function. Dividing this equation by -2 we obtain:

$$\left(y - A(x)\cdot\hat{\theta}\right)^T \cdot A(x) = 0$$

After transposing this equation we get:

$$A^T(x)\left(y - A(x)\cdot\hat{\theta}\right) = 0$$

# Least Squares Parameter Estimation

Expanding the term in brackets gives us:

$$A^T(x)\cdot y - A^T(x)\cdot A(x)\cdot \hat{\theta} = 0$$

which is equal to:

$$A^T(x)\cdot A(x)\cdot \hat{\theta} = A^T(x)\cdot y$$

now left-multiply the left and right hand side with $(A^T(x)\cdot A(x))^{-1}$:

$$\hat{\theta} = \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot y$$

We have derived the **ordinary least squares estimator** for $\theta$ !

# Least Squares Parameter Estimation

1. Obtain state and measurement data: $(x, y)$

2. Formulate linear regression model structure: $p(x,\theta) = A(x)\cdot\theta$

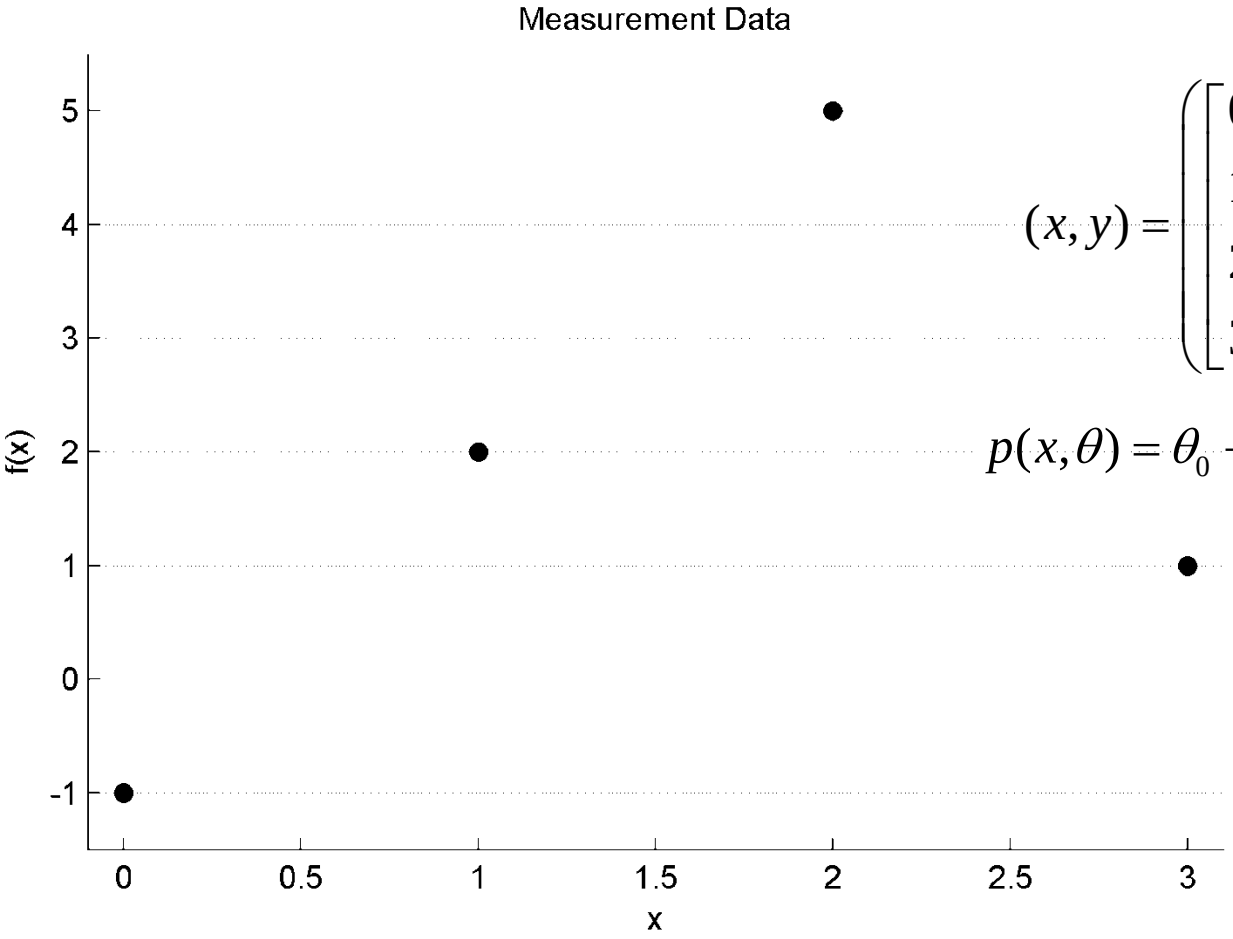
3. Formulate regression matrix: $$A(x) = \begin{bmatrix} 1 & p_1(x(1)) & \cdots & p_M(x(1)) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & p_1(x(N)) & \cdots & p_M(x(N)) \end{bmatrix}$$

4. Formulate least squares estimator: $$\hat{\theta}_{OLS} = \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot y$$

5. Evaluate/Validate model: $$p(x,\hat{\theta}_{OLS}) = A(x)\cdot\hat{\theta}_{OLS}$$

# Parameter Estimation: Introduction

**Exercise:** The human parameter estimator



# Least Squares Parameter Estimation

## Linear Regression

**Example 4.3:** Solving a least squares problem **for yourself**.

We have the following measurement sequence:

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

And we want to use a linear polynomial model structure:

$$p(x, \theta) = \theta_0 + \theta_1 x$$

**Question:** What is the least squares estimate for $\theta_0$, $\theta_1$?

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.3:** Solving a least squares problem (**Answer**).

The linear regression model structure is:

$$p(x,\theta) = \theta_0 + \theta_1 x$$

$$= \begin{bmatrix} 1 & x \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

The Regression matrix is:

$$A(x) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

$$(x,y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.3:** Solving a least squares problem (**Answer**).

The OLS estimator is:

$$\hat{\theta}_{OLS} = \left( A^T(x) \cdot A(x) \right)^{-1} A^T(x) y$$

$$=$$

$$=$$

$$=$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.3:** Solving a least squares problem (**Answer**).

The OLS estimator is:

$$\hat{\theta}_{OLS} = \left( A^T(x) \cdot A(x) \right)^{-1} A^T(x) y$$

$$= \left( \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^T \cdot \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \cdot A^T(x) \cdot y$$

$$= \begin{bmatrix} 14/20 & -6/20 \\ -6/20 & 4/20 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.9 \end{bmatrix}$$

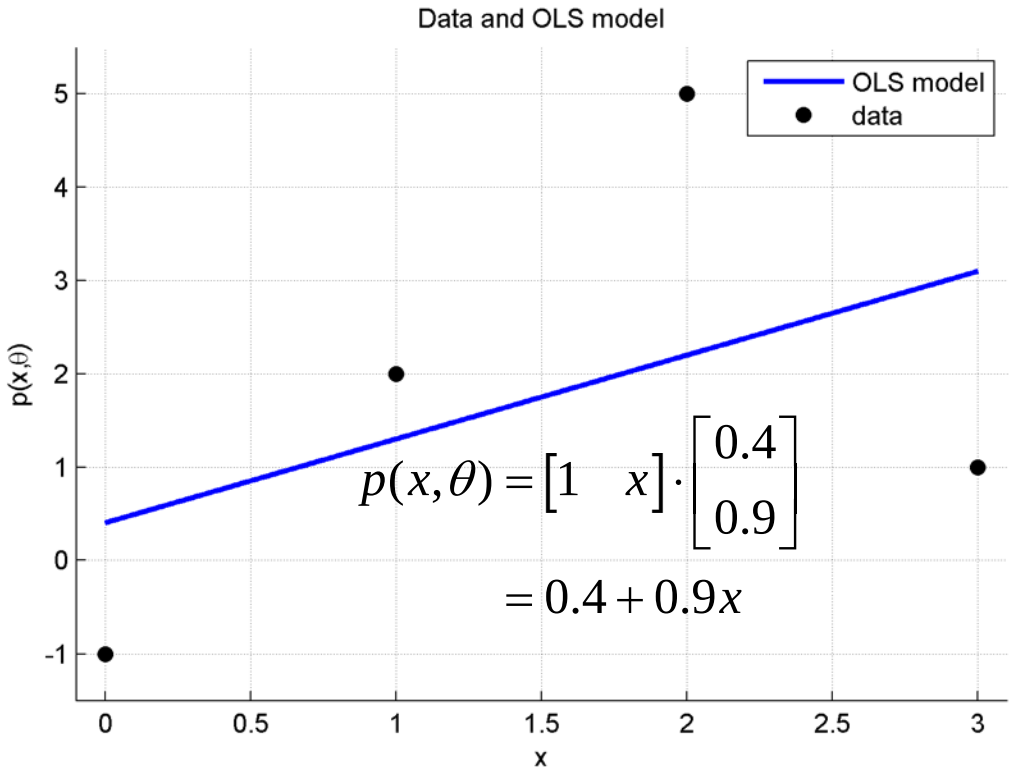
$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} 4 & 6 \\ 6 & 14 \end{bmatrix}^{-1} = \begin{bmatrix} 14/20 & -6/20 \\ -6/20 & 4/20 \end{bmatrix}$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.3:** Solving a least squares problem (**Answer**).




$$p(x,\theta) = \begin{bmatrix} 1 & x \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ 0.9 \end{bmatrix} = 0.4 + 0.9x$$

$$(x,y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.4:** Solving a least squares problem.

Recall the linear regression model we formulated for **Example 4.2**:

$$y = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

with the set of measurement data we obtained earlier this becomes:

$$\begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

$$= A(x) \cdot \theta + \varepsilon$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.4:** Solving a least squares problem.

The least squares estimator is:

$$\hat{\theta}_{OLS} = \left( A^T(x) \cdot A(x) \right)^{-1} \cdot A^T(x) \cdot y$$

with the set of measurement data we obtained earlier this becomes:

$$\hat{\theta}_{OLS} = \left( \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^T \cdot \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix}$$

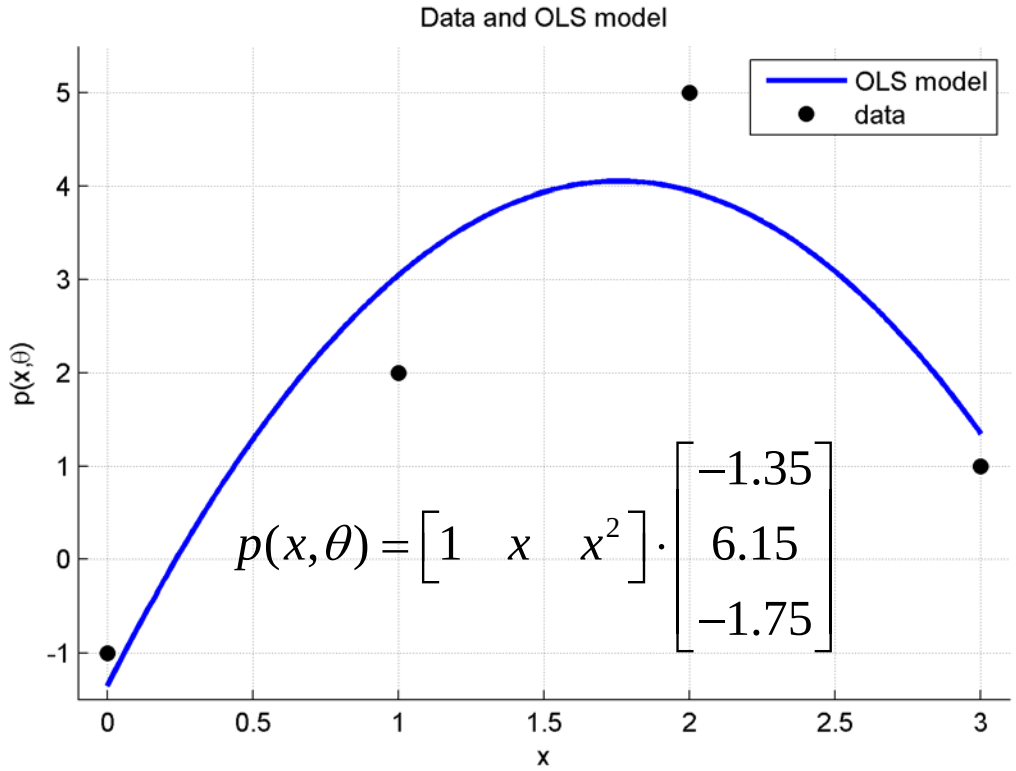
$$= \begin{bmatrix} -1.35 \\ 6.15 \\ -1.75 \end{bmatrix}$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

# Least Squares Parameter Estimation

## Linear Regression

**Example 4.4:** Solving a least squares problem.



# Least Squares Parameter Estimation

## Analysis of the ordinary least squares estimator

- How do we now that the least squares estimator is a good estimator?
- What is a “good” estimator?

A good estimator is a **BLUE estimator** (Best Linear Unbiased Estimator). For an **unbiased estimator** the following must hold:

$$B\left\{\hat{\theta}\right\} = E(\hat{\theta}) - \theta = 0$$

with $B\left\{\hat{\theta}\right\}$ the estimator **bias**, and with $\theta$ the **true** parameter that is being estimated.

**What? Why?**

# Least Squares Parameter Estimation

## The Best Linear Unbiased Estimator

Write the estimated parameters in terms of the 'true' parameters + residual:

$$\hat{\theta}_{OLS} = \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot y$$

$$= \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\left(A(x)\theta+\varepsilon\right)$$

$$= \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)A(x)\cdot\theta + \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\varepsilon$$

$$= \theta + \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\varepsilon$$

**important result!**

$$B\left\{\hat{\theta}\right\} = E(\hat{\theta}) - \theta = 0$$

Taking the **expectancy** of left and right hand terms:

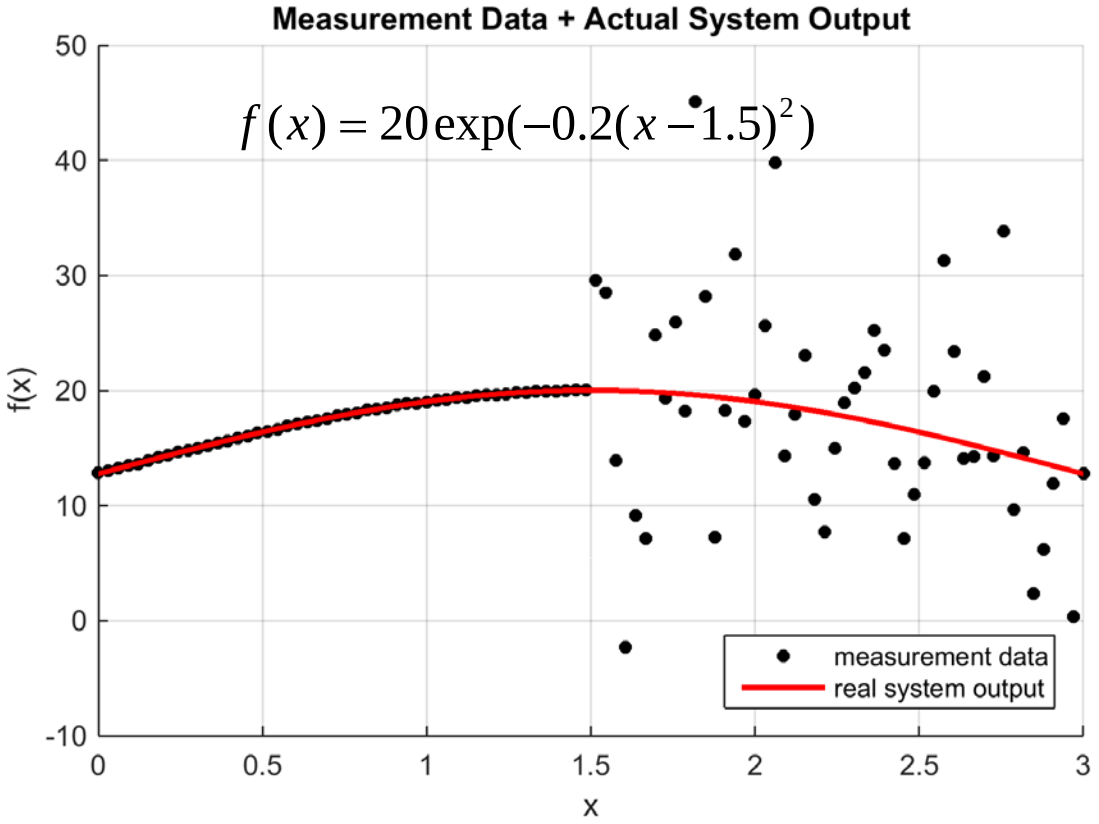
$$E\{\hat{\theta}_{OLS}\} = E\{\theta\} + \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot E\{\varepsilon\}$$

Clearly, we have an unbiased estimator if $E\{\hat{\theta}_{OLS}\} = E\{\theta\}$ which implies $E\{\varepsilon\} = 0$. This is why **all** least squares estimators assume $E\{\varepsilon\} = 0$ !

# Least Squares Parameter Estimation

## Analysis of the OLS estimator

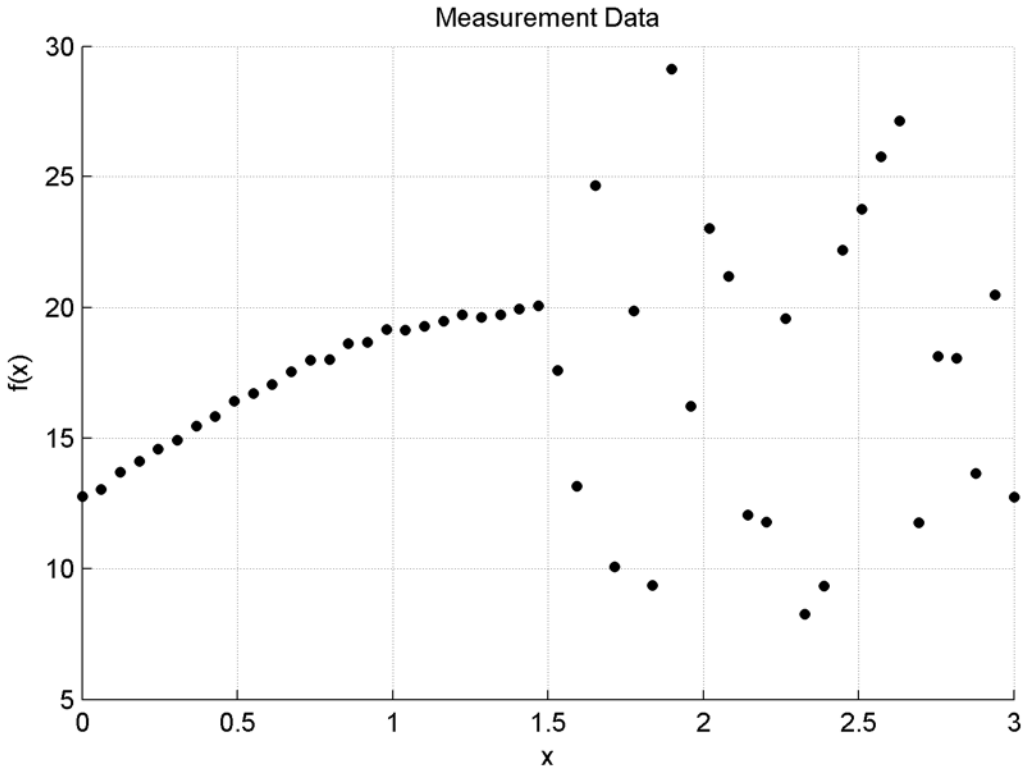
Sensor measurements can contain noise of varying magnitude…



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

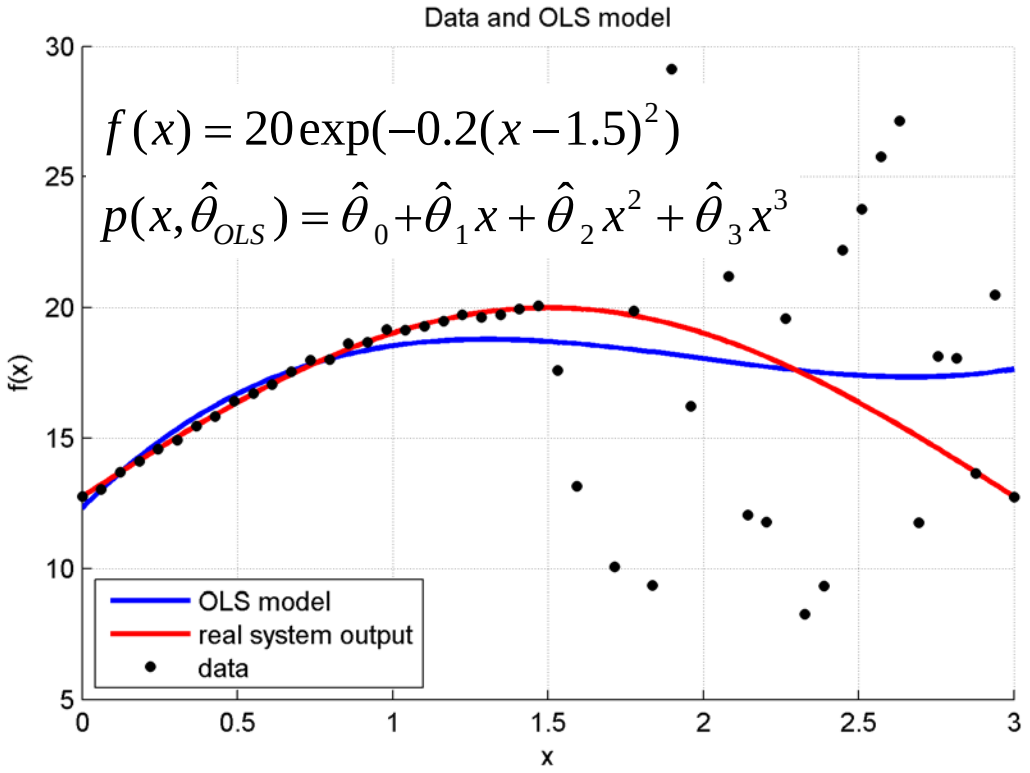
Sensor measurements can contain noise of varying magnitude...



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

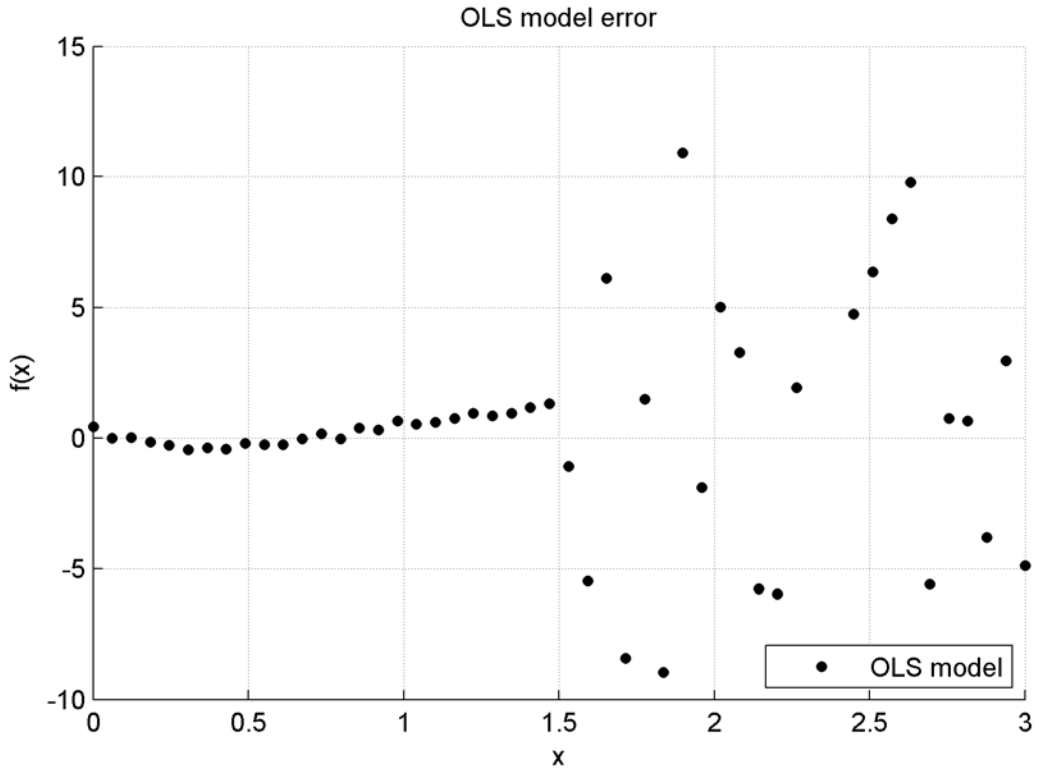
The OLS estimator can deal with zero-mean constant-variance sensor noise…



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

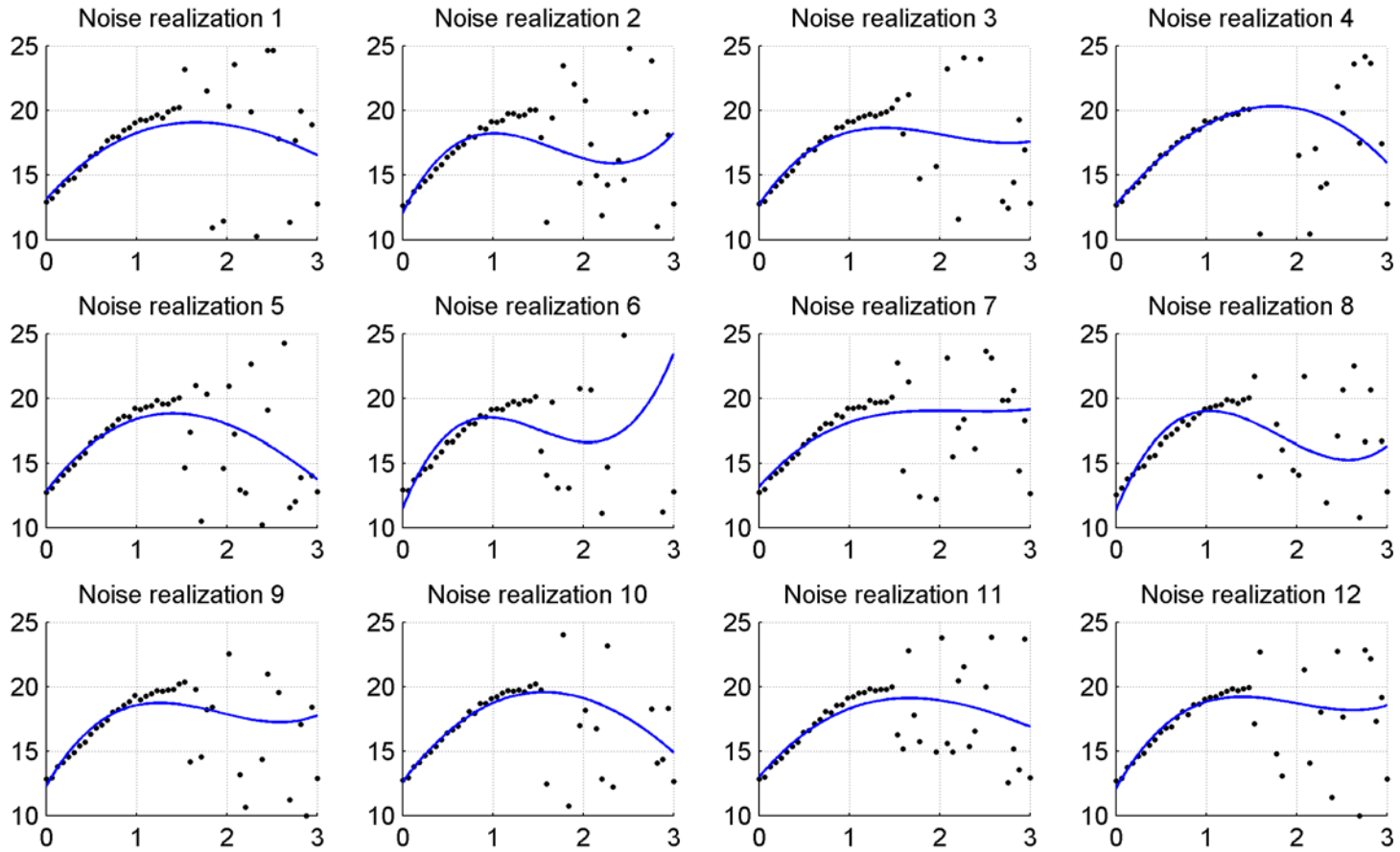
However, model residuals very often do not resemble white noise…



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

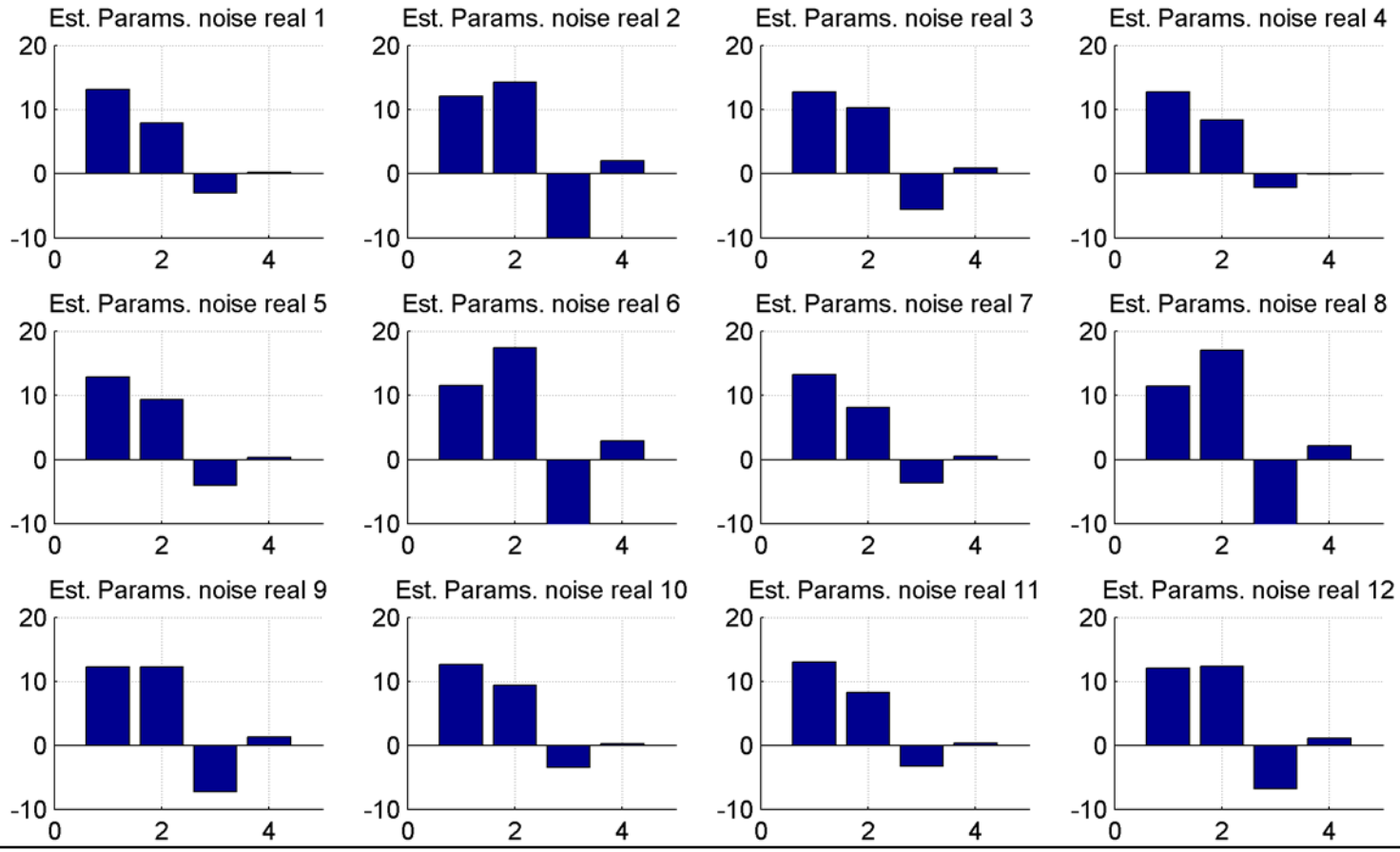
Different sensor noise realizations lead to completely different models!



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

This is caused by wildly varying values for the estimated parameters!



# Least Squares Parameter Estimation

## Analysis of the OLS estimator

Can we in some way predict this **noise sensitivity** or **variability** of our OLS estimator?

$$\hat{\theta}_{OLS} = \left( A^T(x) \cdot A(x) \right)^{-1} A^T(x) \cdot y$$

In fact, the estimator itself contains this information!

This information is contained in the **parameter covariance matrix**:

$$Cov\left\{ \hat{\theta}_{OLS} \right\} = \sigma^2 \left( A^T(x) \cdot A(x) \right)^{-1} = P$$

$n$ = number of measurements

$k$ = number of regressor terms

with $\sigma^2$ the variance of the residual estimated with: $\sigma^2 \approx \hat{\sigma}^2 = \dfrac{\varepsilon^T \varepsilon}{n-k}$

# Least Squares Parameter Estimation

## Analysis of the OLS estimator

Derivation of the covariance matrix $Cov\{\hat{\theta}_{OLS}\} = E\left\{\left(\hat{\theta}_{OLS} - \theta\right)\left(\hat{\theta}_{OLS} - \theta\right)^T\right\}$

Start with the definition of the estimator error $\hat{\theta}_{OLS} - \theta$ :

$$\hat{\theta}_{OLS} - \theta = \left(A^T(x) \cdot A(x)\right)^{-1} A^T(x) \cdot \varepsilon$$

**see sheet 52!**

Substitute in $E\left\{\left(\hat{\theta}_{OLS} - \theta\right)\left(\hat{\theta}_{OLS} - \theta\right)^T\right\}$:

**Residual covariance matrix**

$$E\left\{(\hat{\theta}_{OLS} - \theta)(\hat{\theta}_{OLS} - \theta)^T\right\} =$$

$$= E\left\{\left(A^T(x)A(x)\right)^{-1} A^T(x)\varepsilon\varepsilon^T A(x)\left(A^T(x)A(x)\right)^{-1}\right\}$$

$$= \left(A^T(x)A(x)\right)^{-1} A^T(x) E\left\{\varepsilon \cdot \varepsilon^T\right\} A(x)\left(A^T(x)A(x)\right)^{-1}$$

only $\varepsilon$ is stochastic!

$$= \left(A^T(x)A(x)\right)^{-1} A^T(x)\left(\sigma^2 I\right) A(x)\left(A^T(x)A(x)\right)^{-1}$$

$$= \sigma^2\left(A^T(x)A(x)\right)^{-1} = \sigma^2 P$$

$\varepsilon$ = white noise, so zero cross-correlation:

$$E\left\{\varepsilon \cdot \varepsilon^T\right\} = \sigma^2 I$$

# Least Squares Parameter Estimation

## Analysis of the OLS estimator

The parameter variances are the elements on the main diagonal of $Cov\{\hat{\theta}_{OLS}\}$:

$$Var\left\{\hat{\theta}_{OLS}\right\} = diag\left(Cov\left\{\hat{\theta}_{OLS}\right\}\right)$$

The parameter **covariances** are the off-diagonal elements of $Cov\{\hat{\theta}_{OLS}\}$:

$$Cov\left(\hat{\theta}_{OLS}(i), \hat{\theta}_{OLS}(j)\right) = Cov\left\{\hat{\theta}_{OLS}\right\}_{i,j}$$

# Least Squares Parameter Estimation

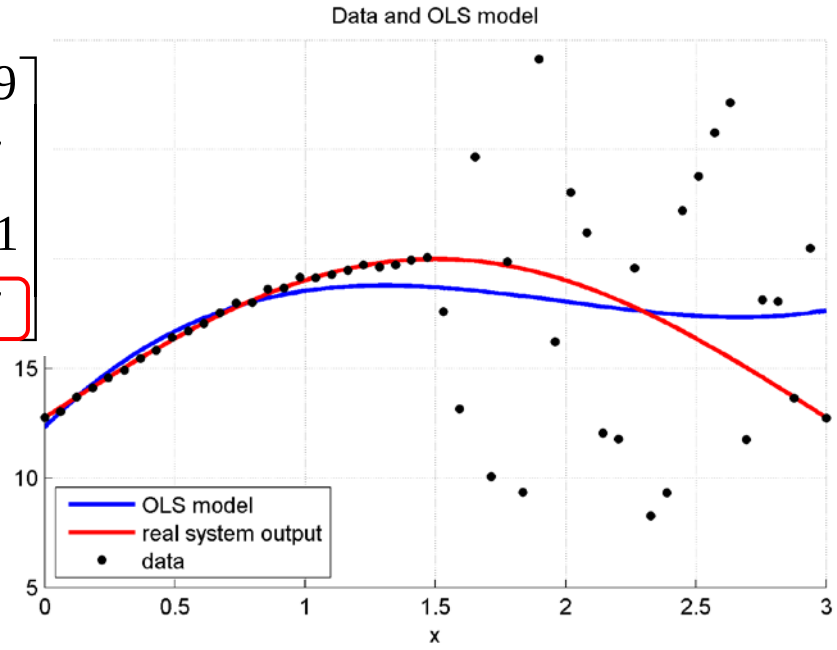
## Analysis of the OLS estimator

**Example 4.5:** Parameter (co)variances

For the earlier used dataset we find the following parameter covariance matrix:

$$Cov\left\{\hat{\theta}_{OLS}\right\} = \sigma^2\left(A^T(x)\cdot A(x)\right)^{-1}$$

$$= \begin{bmatrix} 0.28 & -0.68 & 0.45 & -0.09 \\ -0.68 & 2.35 & -1.77 & 0.37 \\ 0.45 & -1.77 & 1.43 & -0.31 \\ -0.09 & 0.37 & -0.31 & 0.07 \end{bmatrix}$$




And the following parameter variances:

$$Var\left\{\hat{\theta}_{OLS}\right\} = \begin{bmatrix} 0.28 & 2.35 & 1.43 & 0.07 \end{bmatrix}^T$$

# Least Squares Parameter Estimation

## Analysis of the OLS estimator

**Example 4.5:** Parameter (co)variances

$$Cov\left\{\hat{\theta}_{OLS}\right\} = \begin{array}{c} \\ \hat{\theta}_0 \\ \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{array} \begin{array}{c} \begin{array}{cccc} \hat{\theta}_0 & \hat{\theta}_1 & \hat{\theta}_2 & \hat{\theta}_3 \end{array} \\ \begin{bmatrix} 0.28 & -0.68 & 0.45 & -0.09 \\ -0.68 & 2.35 & -1.77 & 0.37 \\ 0.45 & -1.77 & 1.43 & -0.31 \\ -0.09 & 0.37 & -0.31 & 0.07 \end{bmatrix} \end{array}$$

Which parameters are **correlated** most?

Answer: $Cov\left\{\hat{\theta}_1, \hat{\theta}_2\right\} = -1.77$

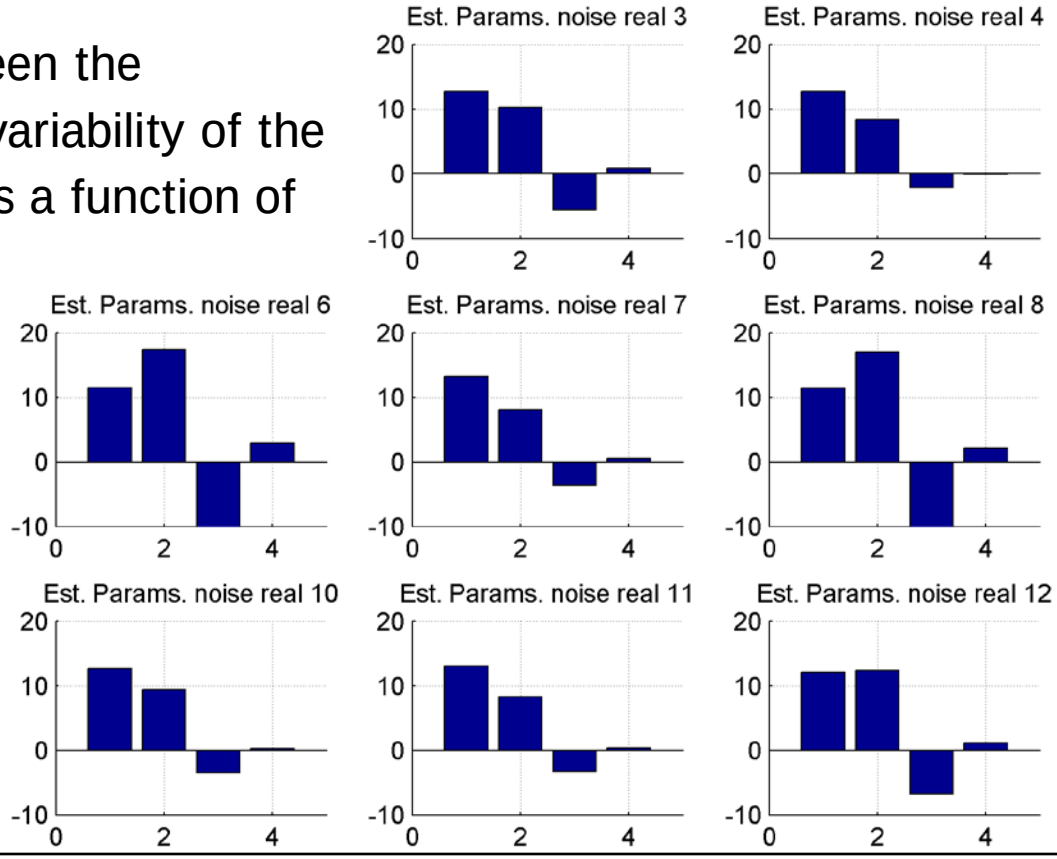
# Least Squares Parameter Estimation

## Analysis of the OLS estimator

**Example 4.5:** Parameter (co)variances

Observe the relationship between the parameter variances, and the variability of the parameter estimation results as a function of the noise:

$$Var\left\{\hat{\theta}_{OLS}\right\} = \begin{bmatrix} 0.28 \\ 2.35 \\ 1.43 \\ 0.07 \end{bmatrix}$$



# Least Squares Parameter Estimation

## Analysis of the OLS estimator: Conclusion

The OLS estimator assumes that:

1) $\varepsilon$ has a constant variance for all measurements:

$$E\left\{\varepsilon\varepsilon^T\right\} = \sigma^2 I$$

This is the **residual covariance matrix**, with the scalar $\sigma$ the noise standard deviation, and with $I$ the $N \times N$ identity matrix.

2) $\varepsilon$ is zero mean white noise:

$$E\left\{\varepsilon\right\} = 0$$

These are necessary conditions to make the OLS estimator BLUE!

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

Normally, the linear regression model error (Equation Error!) is given by:

$$\varepsilon = y - A(x) \cdot \theta$$

Let's assume for now that $\varepsilon$ is not white noise, but instead a (weighted) noise process as follows:

$$\varepsilon = W^{1/2} \nu$$

with $W^{1/2}$ the square root of a **diagonal** $N \times N$ constant noise scaling matrix $W$. Think of this matrix as containing unique scaling factors for each individual residual.

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

We construct the weighting matrix $W$ using **a-priori** information on the system. If we are making measurements on the system at locations:

$$x = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T$$

And we know the corresponding standard deviations of the sensor noise in these states:

$$\sigma = \begin{bmatrix} \sigma_1 & \sigma_2 & \cdots & \sigma_N \end{bmatrix}^T$$

Then the matrix $W$ is constructed as follows:

$$W = diag\left(\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2\right) = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_N^2 \end{bmatrix}$$

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

Substitute $\varepsilon = W^{1/2} \cdot \nu$ in

$$\varepsilon = Y - A(x) \cdot \theta$$

Gives

$$W^{1/2} \cdot \nu = y - A(x) \cdot \theta$$

The new **white noise** residual becomes:

$$\nu = W^{-1/2} \cdot \left( y - A(x) \cdot \theta \right)$$

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

The new residual is:

$$\nu = W^{-1/2} \cdot \left( y - A(x) \cdot \theta \right)$$

Which has a least squares optimum at $\hat{\theta}$ if:

$$\frac{\partial \nu^T \nu}{\partial \theta} = 2\nu^T \frac{\partial \nu}{\partial \theta}$$

$$-2\left[ W^{-1/2} \cdot \left( y - A(x) \cdot \theta \right) \right]^T \cdot W^{-1/2} \cdot A(x) = 0$$

rearranging, and noticing that $\left(W^{-1/2}\right)^T = W^{-1/2}$ and $W^{-1/2}W^{-1/2} = W^{-1}$ leads to:

$$y^T \cdot \left( W^{-1} \right)^T \cdot A(x) - \theta^T \cdot A^T(x) \cdot \left( W^{-1} \right)^T \cdot A(x) = 0$$

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

We further rearrange

$$y^T \cdot \left(W^{-1}\right)^T \cdot A(x) - \theta^T \cdot A^T(x) \cdot \left(W^{-1}\right)^T \cdot A(x) = 0$$

into:

$$A^T(x) \cdot W^{-1} \cdot y = A^T(x) \cdot W^{-1} \cdot A(x) \cdot \theta$$

Multiplying the right and left hand side with $A^T(x) \cdot W^{-1} \cdot A(x)$ results in a new estimator:

$$\begin{aligned} \hat{\theta} &= \left(A^T(x) \cdot W^{-1} \cdot A(x)\right)^{-1} \cdot A^T(x) \cdot W^{-1} \cdot y \\ &= P_{WLS} \cdot A^T(x) \cdot W^{-1} \cdot y \end{aligned}$$

# Weighted Least Squares Estimation

## Derivation of the Weighted Least Squares (WLS) Estimator

For unbiased white, scaled, residuals (i.e. $E\{Wv\} = 0$) the **weighted least squares estimator** (WLS) is defined as follows:

$$\begin{aligned}\hat{\theta}_{WLS} &= \left(A^T(x)\cdot W^{-1}\cdot A(x)\right)^{-1}\cdot A^T(x)\cdot W^{-1}\cdot y \\ &= P_{WLS}\cdot A^T(x)\cdot W^{-1}\cdot y\end{aligned}$$

With $W$ the **diagonal** $N \times N$ constant noise scaling matrix.

Note that if we let $W = I$ , which implies the noise is of constant magnitude for all data points, the WLS estimator simplifies into the OLS estimator:

$$\begin{aligned}\hat{\theta}_{WLS}\Big|_{W=I} &= \left(A^T(x)\cdot\frac{1}{\sigma^2}I^{-1}\cdot A(x)\right)^{-1}\cdot A^T(x)\cdot\frac{1}{\sigma^2}I^{-1}\cdot y \\ &= \left(A^T(x)\cdot A(x)\right)^{-1}\cdot A^T(x)\cdot y \\ &= \hat{\theta}_{OLS}\end{aligned}$$

# Weighted Least Squares Estimation

1. Obtain state, measurement, noise statistics data: $(x, y, \sigma)$

2. Formulate linear regression model structure: $p(x,\theta) = A(x)\cdot\theta$

3. Formulate regression matrix: $$A(x) = \begin{bmatrix} 1 & p_1(x(1)) & \cdots & p_M(x(1)) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & p_1(x(N)) & \cdots & p_M(x(N)) \end{bmatrix}$$

4. Formulate residual weighting matrix: $W = diag(\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2)$

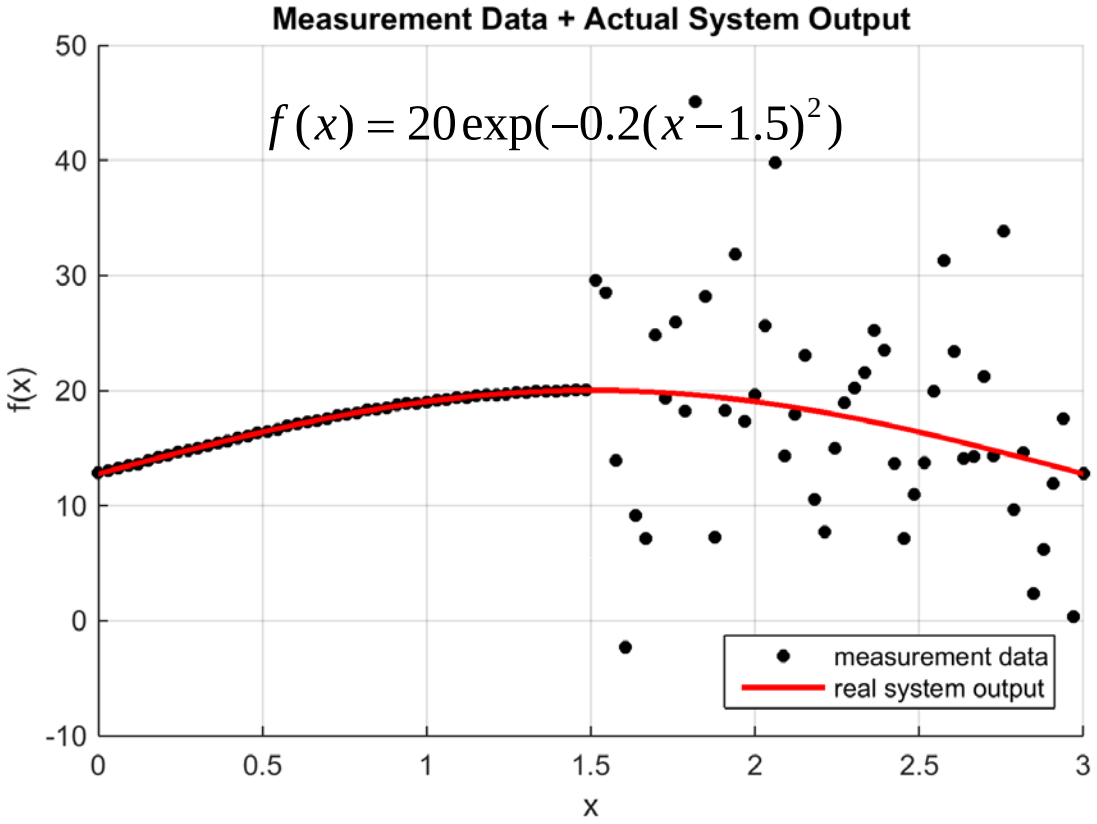
5. Formulate weighted least squares estimator: $\hat{\theta}_{WLS} = \left(A^T(x)\cdot W^{-1}\cdot A(x)\right)^{-1}\cdot A^T(x)\cdot W^{-1}\cdot y$

6. Evaluate model: $p(x,\hat{\theta}_{WLS}) = A(x)\cdot\hat{\theta}_{WLS}$

# Weighted Least Squares Estimation

## Analysis of the WLS estimator

We return to the non-constant noise variance dataset:



# Weighted Least Squares Estimation

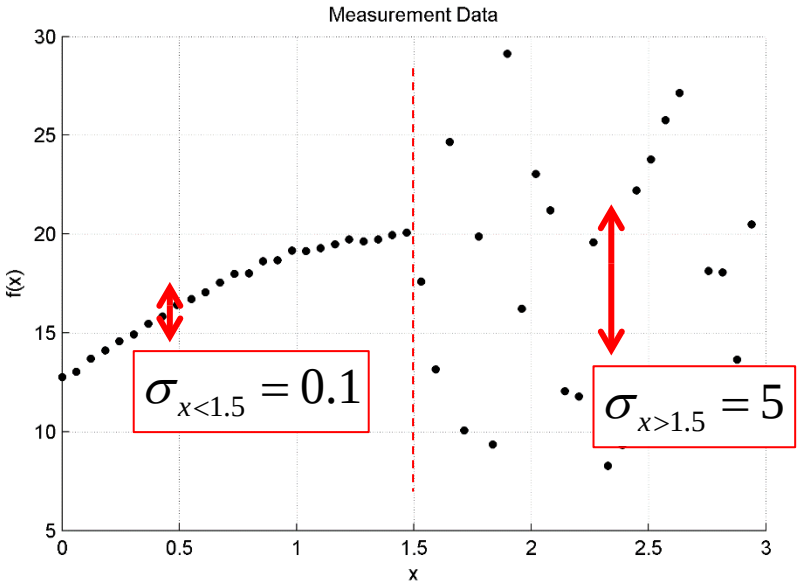
## Analysis of the WLS estimator

We will now proceed to define a WLS estimator for the given dataset, and use a third order model structure:

$$y = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + W \cdot \nu$$

We now have to define $W$.

For this, we have to analyze the OLS model residual, or otherwise have prior knowledge of sensor noise characteristics.



# Weighted Least Squares Estimation

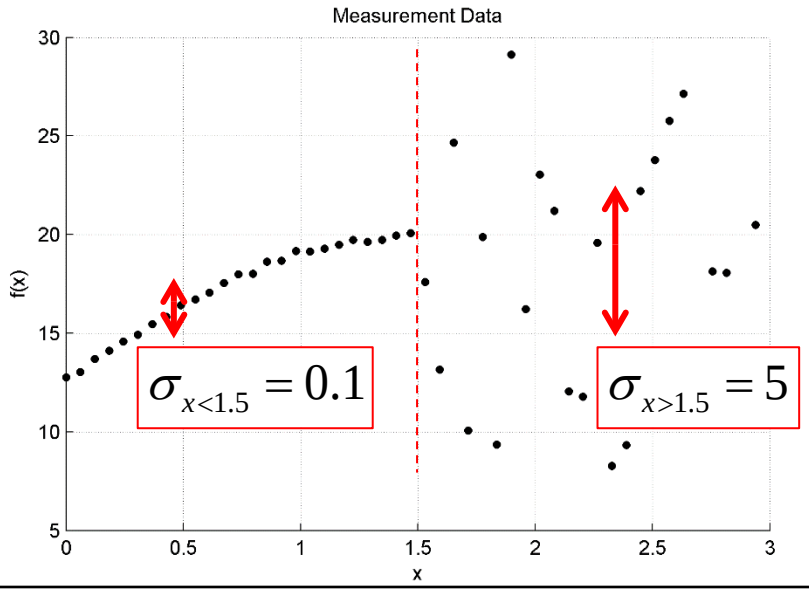
## Analysis of the WLS estimator

Construction of weight matrix $W$ may require calculating a OLS solution first.

In this particular case, we notice that the standard deviation of the noise increases sharply for $x > 1.5$.

If the data is sorted along $x$, then we find for $W$:

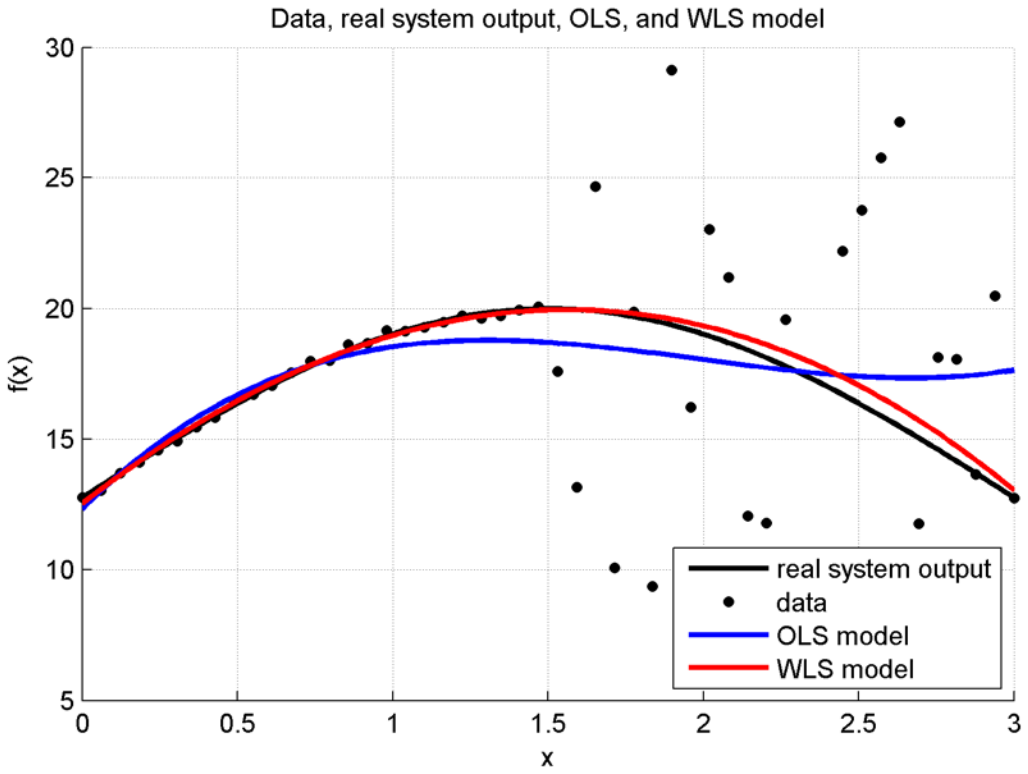
$$W = \begin{bmatrix} \sigma^2_{x<1.5} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & 0 & 0 & \vdots \\ \vdots & \ddots & \sigma^2_{x<1.5} & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \sigma^2_{x>1.5} & \ddots & \vdots \\ \vdots & 0 & 0 & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \sigma^2_{x>1.5} \end{bmatrix}$$



# Weighted Least Squares Estimation

## Analysis of the WLS estimator

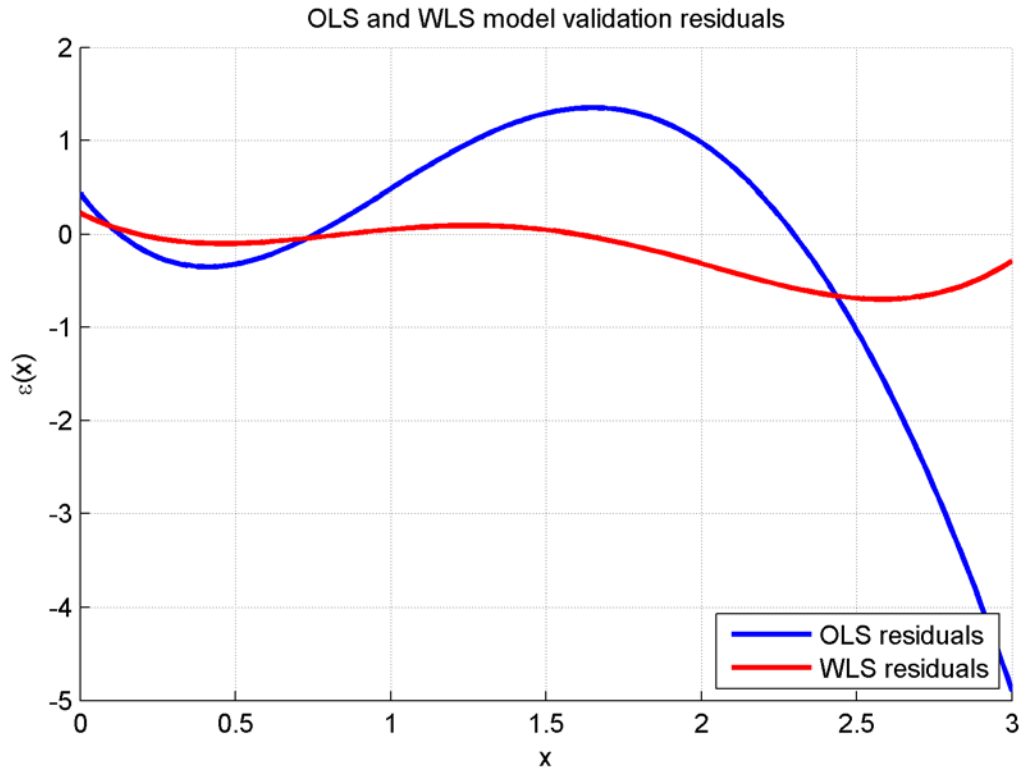
The resulting WLS model is much closer to the real system output:



# Weighted Least Squares Estimation

## Analysis of the WLS estimator

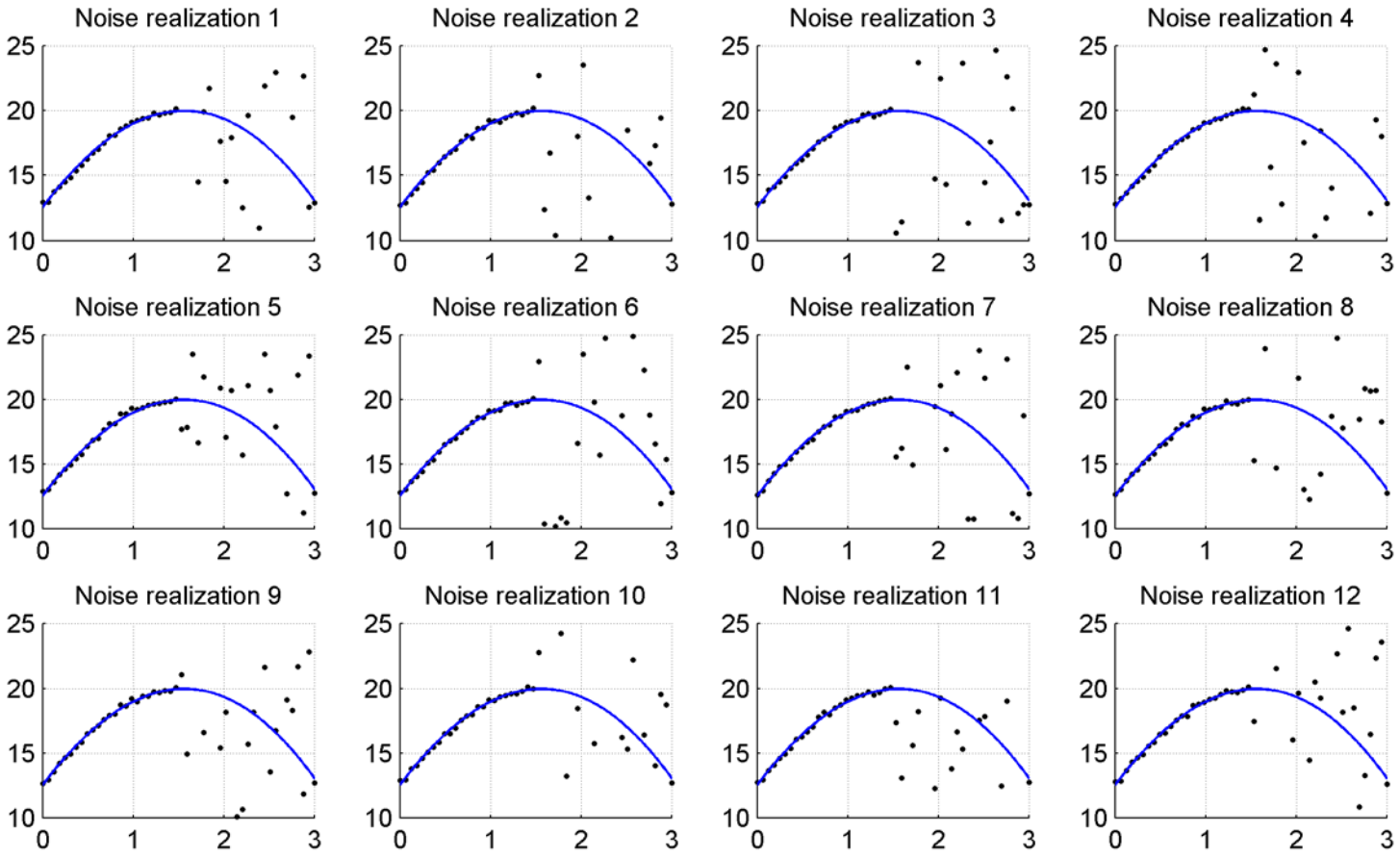
The WLS model residual is significantly smaller than that of the OLS model:



# Weighted Least Squares Estimation

## Analysis of the WLS estimator

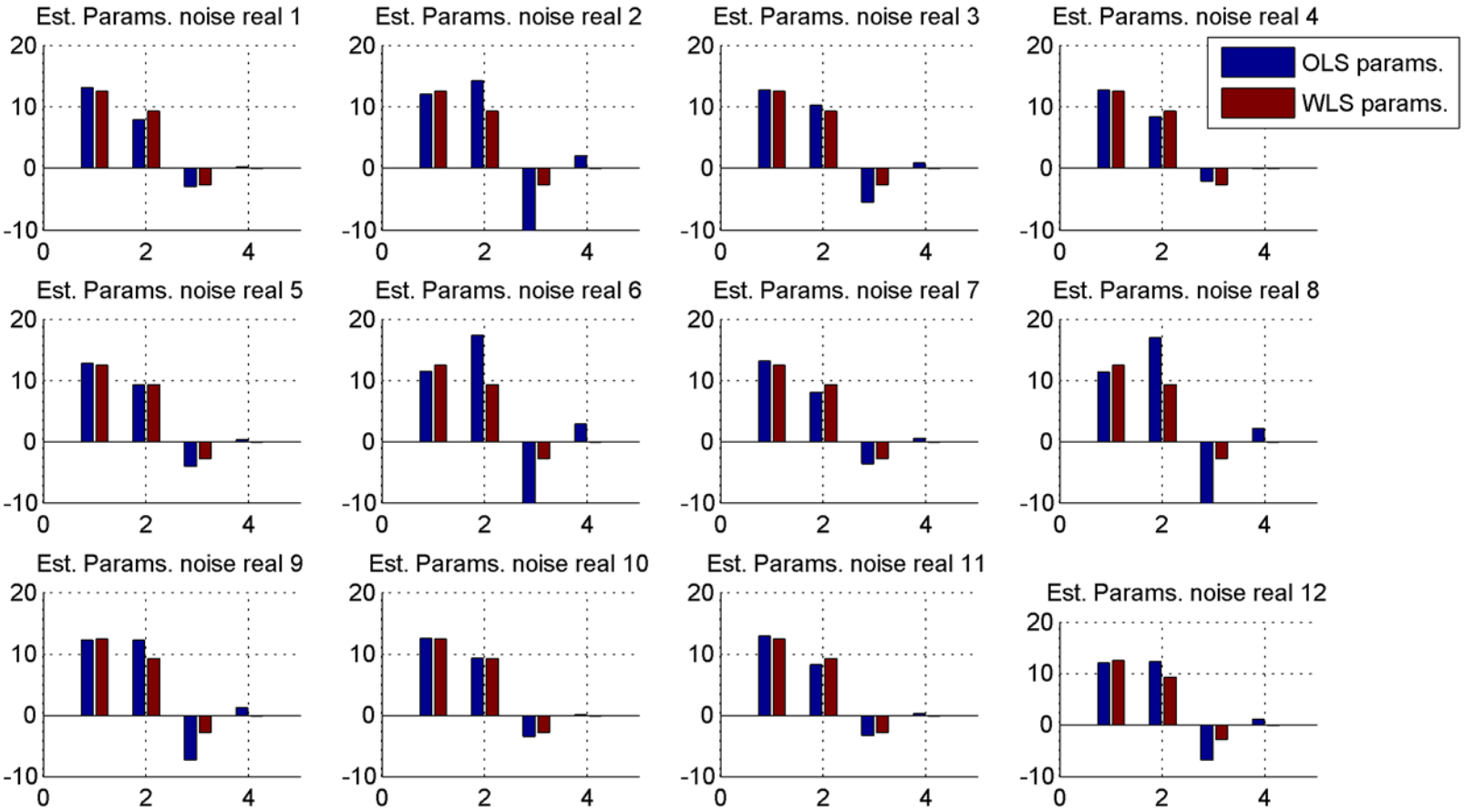
Different sensor noise realizations lead to virtually identical models!



# Weighted Least Squares Estimation

## Analysis of the WLS estimator

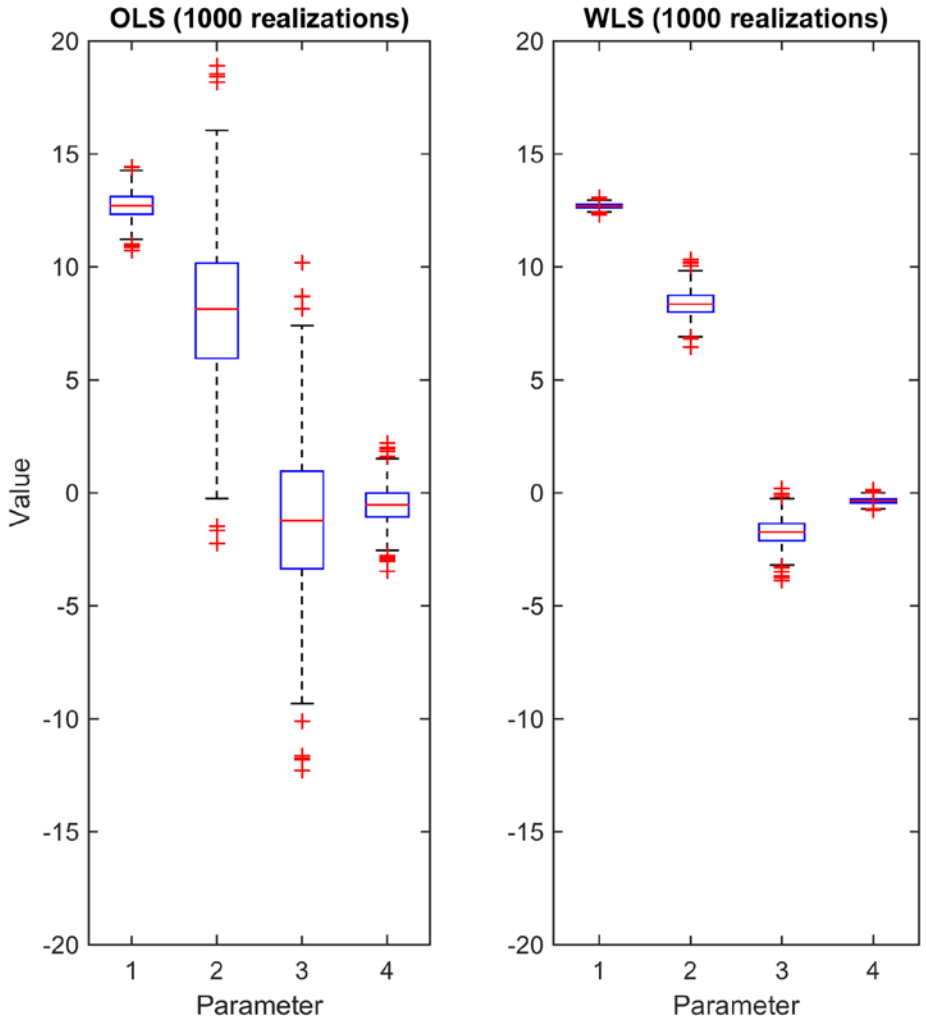
The WLS estimates are much less influenced by noise than OLS estimates!



# Weighted Least Squares Estimation

The box plot shows the OLS and WLS parameter estimation statistics for 1000 noise realizations.

Red lines are the medians, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers. Red crosses are considered outliers.



# Least Squares Parameter Estimation

## Analysis of the WLS estimator

**Example 4.6:** Parameter (co)variances

We find the following WLS parameter covariance matrix:

$$Cov\left\{\hat{\theta}_{WLS}\right\} = \left(A^T(x)\cdot W^{-1}\cdot A(x)\right)^{-1}$$

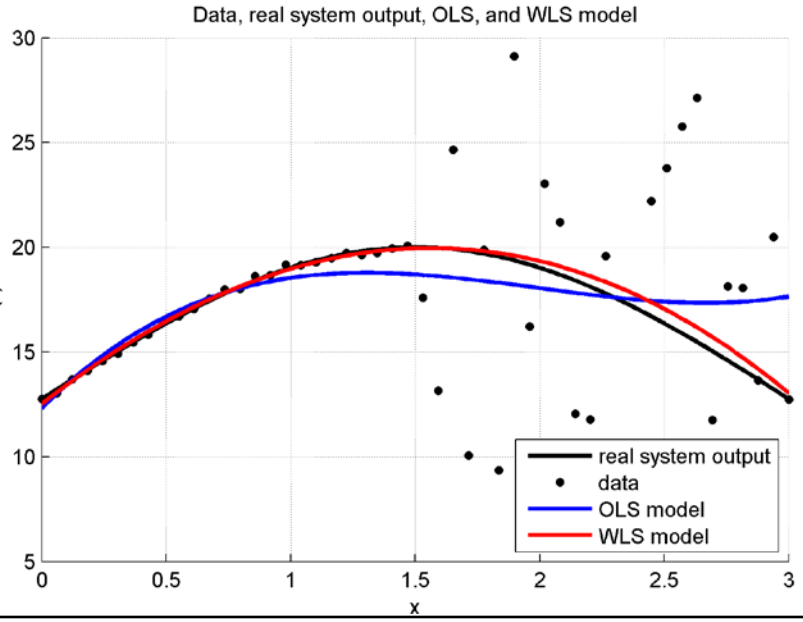
$$= \begin{bmatrix} 0.03 & -0.11 & 0.09 & -0.02 \\ -0.11 & 0.48 & -0.44 & 0.10 \\ 0.09 & -0.44 & 0.44 & -0.10 \\ -0.02 & 0.10 & -0.10 & 0.02 \end{bmatrix}$$

And the following parameter WLS variances:

$$Var\left\{\hat{\theta}_{WLS}\right\} = \begin{bmatrix} 0.03 & 0.48 & 0.44 & 0.02 \end{bmatrix}^T$$

Compare this with OLS parameter variances:

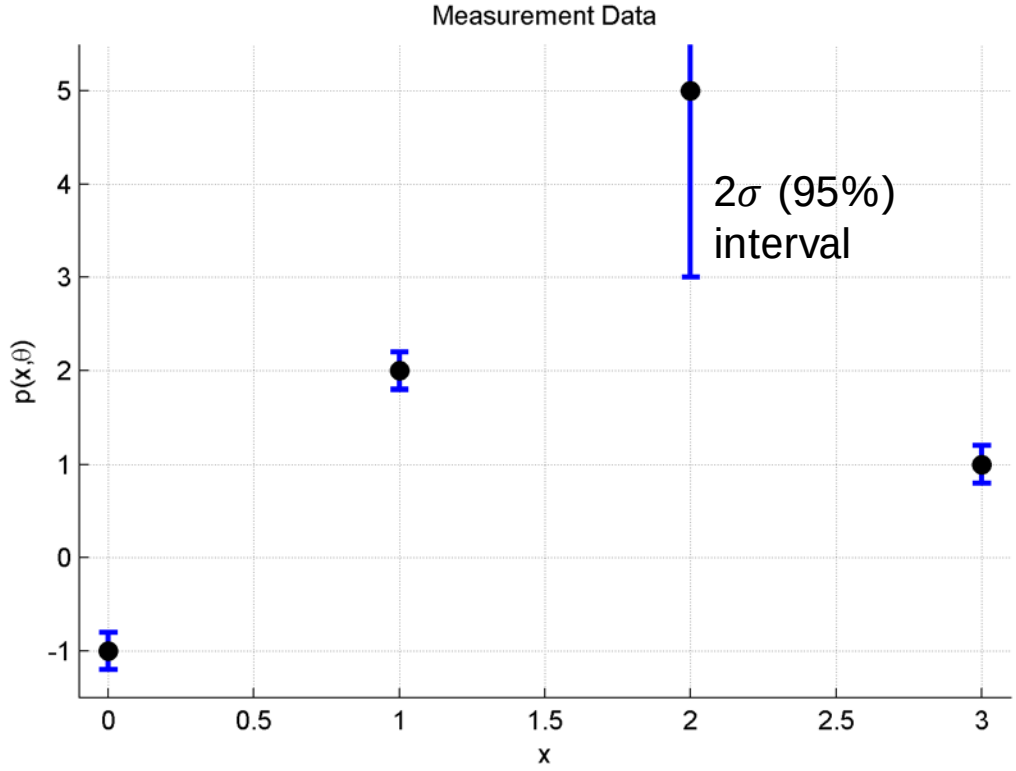
$$Var\left\{\hat{\theta}_{OLS}\right\} = \begin{bmatrix} 0.28 & 2.35 & 1.43 & 0.07 \end{bmatrix}^T$$



# Weighted Least Squares Estimation

**Example 4.7:** Weighted least squares estimator

Assume that we have a set of 4 measurements, and we know the expected variance of the measurements at the measurement locations:




$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right),$$

$$\sigma = \begin{bmatrix} 0.1 \\ 0.1 \\ 1.0 \\ 0.1 \end{bmatrix}$$

# Weighted Least Squares Estimation

**Example 4.7:** Weighted least squares estimator

We want to estimate the parameters of a quadratic linear regression model:

$$y = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \varepsilon$$

The weight matrix is constructed using the a-priori noise values for the measurements:

$$W = \left(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2\right) = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

$$\sigma = \begin{bmatrix} 0.1 \\ 0.1 \\ 1.0 \\ 0.1 \end{bmatrix}$$

# Weighted Least Squares Estimation

**Example 4.7:** Weighted least squares estimator

The WLS estimator is:

$$\hat{\theta}_{WLS} = \left( A^T(x) W^{-1} A(x) \right)^{-1} A^T(x) W^{-1} y$$

$$= \left( \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^T \cdot W^{-1} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^T \cdot W^{-1} \cdot \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1.08 \\ 4.21 \\ -1.18 \end{bmatrix}$$

$$(x, y) = \left( \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \right)$$

$$\sigma = \begin{bmatrix} 0.1 \\ 0.1 \\ 1.0 \\ 0.1 \end{bmatrix}$$

# Weighted Least Squares Estimation

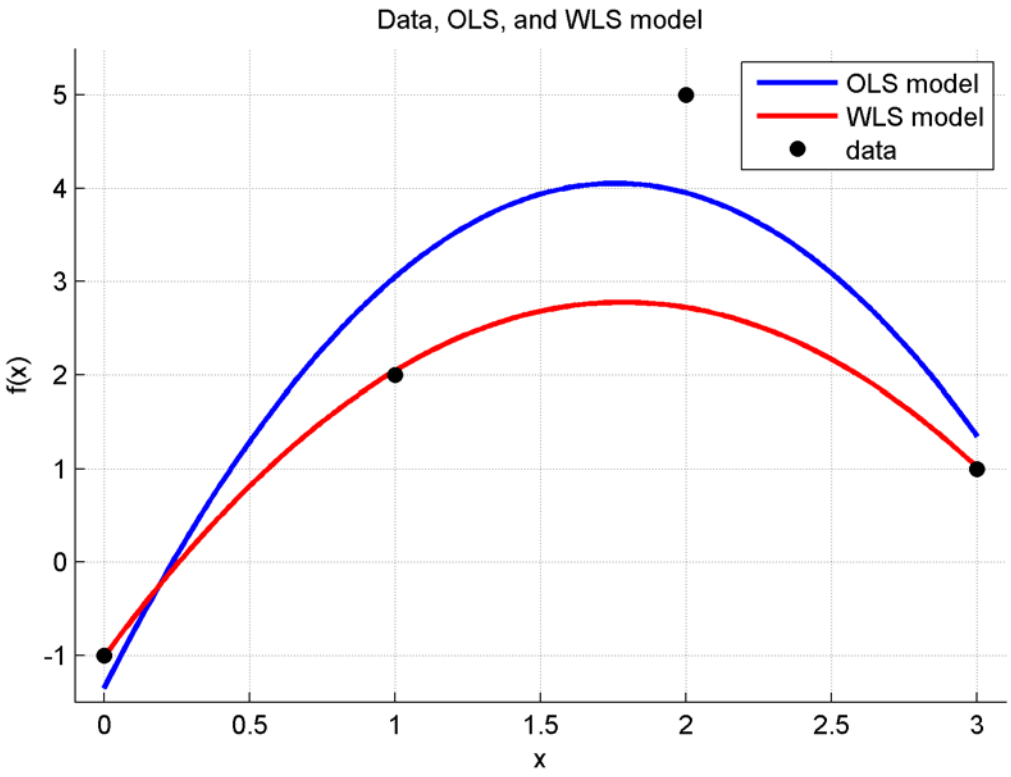
**Example 4.7:** Weighted least squares estimator

The OLS model was:

$$p(x,\hat{\theta}_{OLS}) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} -1.35 \\ 6.15 \\ -1.75 \end{bmatrix}$$

The WLS model is:

$$p(x,\hat{\theta}_{WLS}) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \cdot \begin{bmatrix} -1.08 \\ 4.21 \\ -1.18 \end{bmatrix}$$



# Generalized Least Squares Estimation

The generalized least squares (**GLS**) estimator is a generalization of the weighted least squares estimator which allows **correlated residuals**:

$$E\left\{\varepsilon\varepsilon^T\right\} = \Sigma$$

with $\Sigma$ the residual covariance matrix which can be used as a **non-diagonal weighting** matrix:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 & \cdots & \sigma_1\sigma_N \\ \sigma_2\sigma_1 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sigma_{N-1}\sigma_N \\ \sigma_N\sigma_1 & \cdots & \sigma_N\sigma_{N-1} & \sigma_N^2 \end{bmatrix}$$

# Generalized Least Squares Estimation

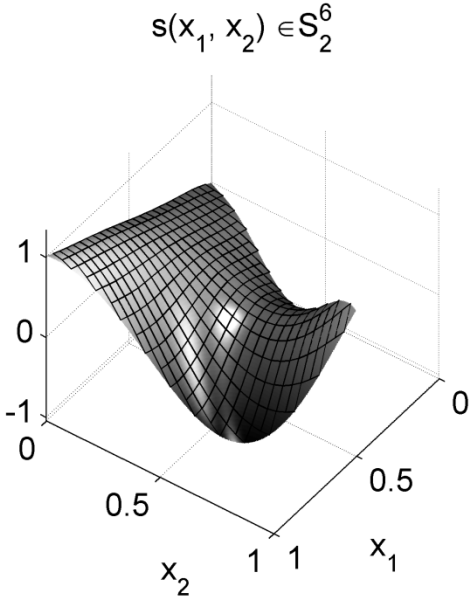
The generalized least squares estimator in most cases is implemented in a two-stage estimation approach. In the first stage, the residual covariance matrix is estimated:

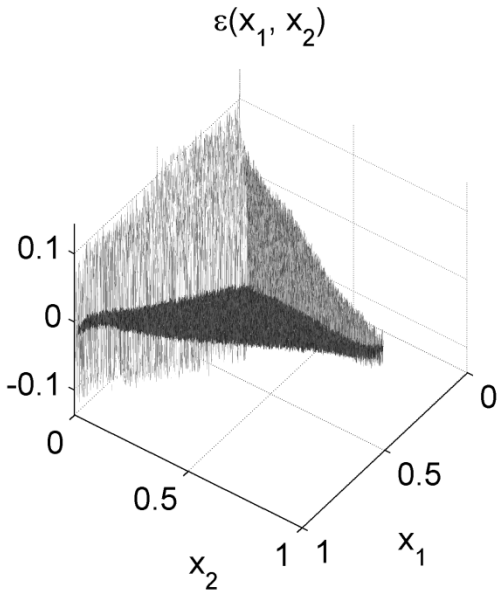
$$\Sigma = E\left\{\left(y - A(x)\hat{\theta}_{OLS}\right)\left(y - A(x)\hat{\theta}_{OLS}\right)^T\right\}$$

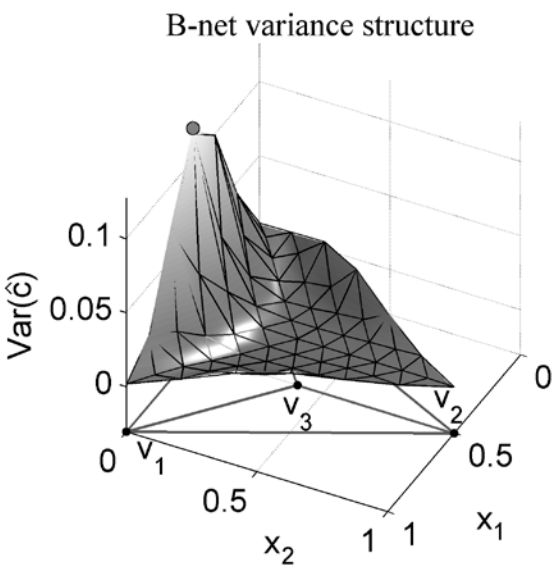
We then use the exact same procedure to formulate the estimator as with the weighted least squares estimator, but instead use $\Sigma$ as the weighting matrix:

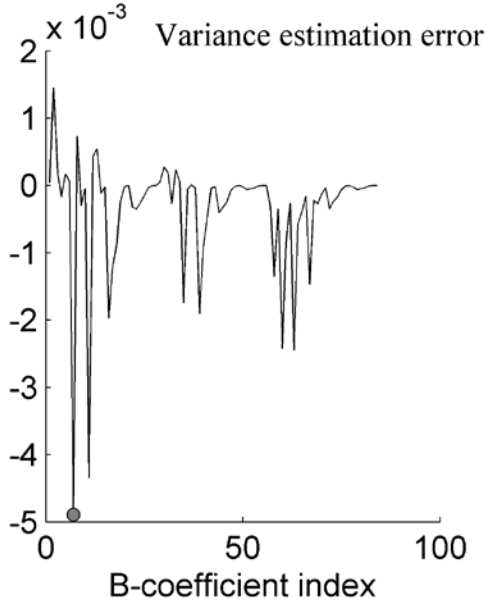
$$\hat{\theta}_{GLS} = \left(A^T(x)\cdot\Sigma^{-1}\cdot A(x)\right)^{-1} A^T(x)\cdot\Sigma^{-1}\cdot y$$

# Generalized Least Squares Estimation



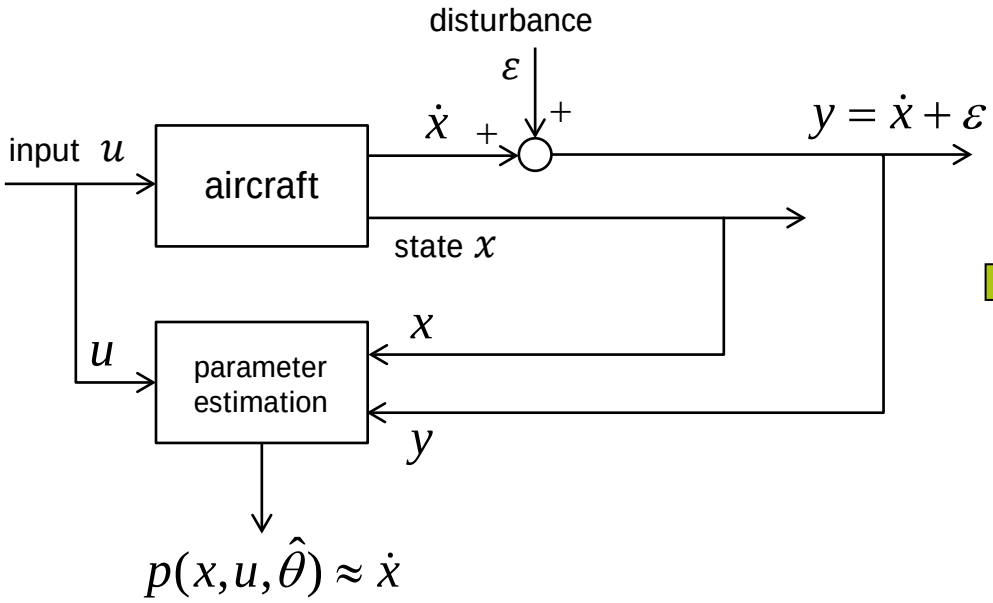



# Limits of OLS Estimators

**Equation Error** approach to parameter estimation:




**Note:**
no process noise
no disturbance on $x$

We assume we can measure the state derivative (plus noise): $y = \dot{x} + \varepsilon$.

We try to model the system output: $\dot{x} \cong p(x,u,\theta)$.

The **Equation Error** is: $\varepsilon = y - p(x,u,\theta)$

# Limits of OLS Estimators

Remember BLUE: $E\{\hat{\theta}_{OLS}\} := E\{\theta\}$

Write the estimated parameters in terms of the 'true' parameters + residual:

$$\begin{aligned}\hat{\theta}_{OLS} &= \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot y \\ &= \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\left(A(x)\theta+\varepsilon\right) \\ &= \left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)A(x)\cdot\theta+\left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\varepsilon \\ &= \theta+\left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot\varepsilon\end{aligned}$$

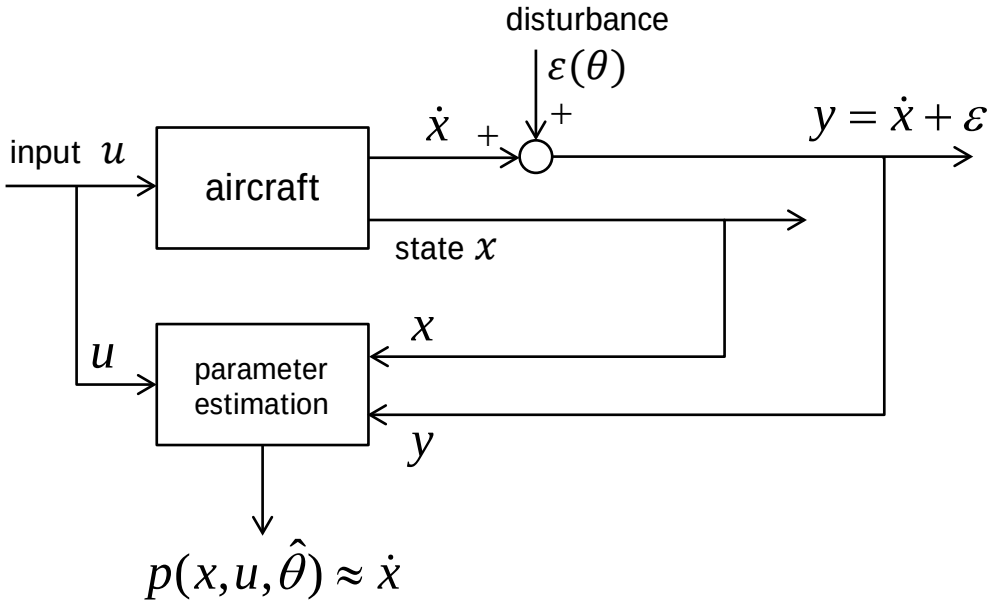
Taking the expectancy of left and right hand terms:

$$E\{\hat{\theta}_{OLS}\} = E\{\theta\}+\left(A^T(x)\cdot A(x)\right)^{-1} A^T(x)\cdot E\{\varepsilon\}$$

So for BLUE we must have $E\{\hat{\theta}_{OLS}\} = E\{\theta\}$ implying $E\{\varepsilon\} = 0$.

# Limits of OLS Estimators

Everything changes when the real system is not in the model set $p(x,u,\theta)$…




**Note:** $\varepsilon(\theta)$ can be seen as a model structure deficiency!

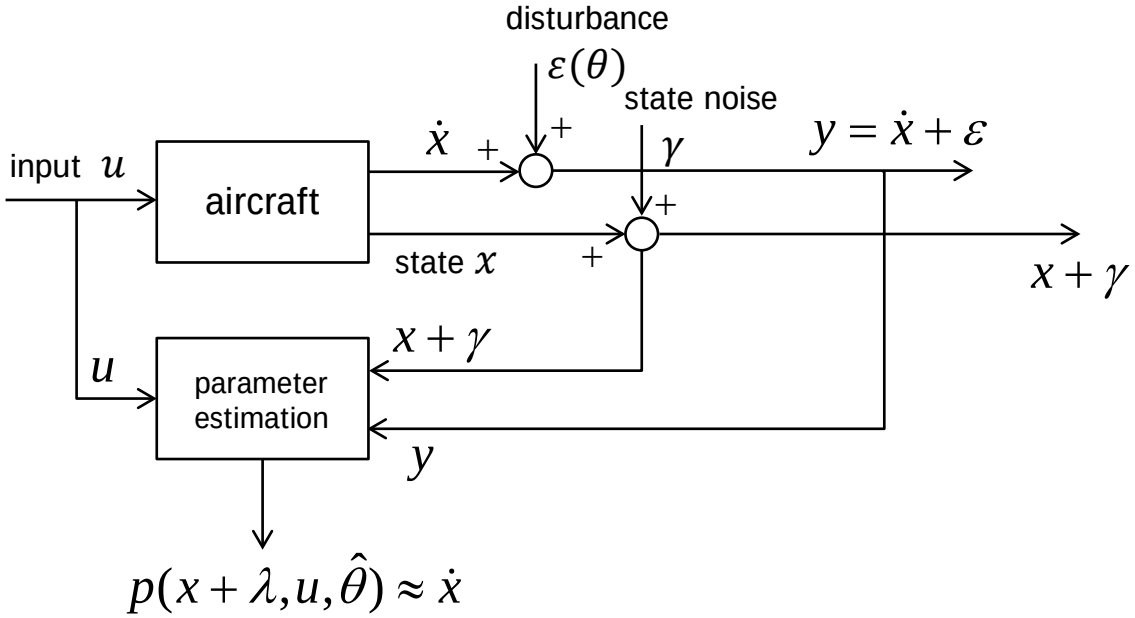
System output is approximated by our model $\dot{x} \cong p(x,u,\theta)$.

The **Equation Error** now is: $\varepsilon(\theta) = y - p(x,u,\theta)$ ⇨ depends on parameters!

non-zero mean ⇦ $E\{\varepsilon(\theta)\} \neq 0,\ E\{\varepsilon(\theta)\varepsilon^T(\theta)\} = \Sigma(\theta)$ ⇨ correlated residual

# Limits of OLS Estimators

In addition, there might be (unknown) state or system noise acting on $x$ :




System output is approximated by our model $\dot{x} \cong p(x + \lambda, u, \theta)$.

The **Equation Error** now is: $\varepsilon(\theta) = y - p(x + \lambda, u, \theta)$ ⇨ regressors are uncertain!

We now have a combined state/parameter estimation problem…

# Output Error & Maximum Likelihood

## Principle of Maximum Likelihood

What do we do when the residual is not uncorrelated zero-mean white noise, or if there is state noise?

$$y = A(x+\lambda)\cdot\theta + \varepsilon(\theta) \Rightarrow \; E\left\{\varepsilon(\theta)\right\} \neq 0, \;\; E\left\{\varepsilon(\theta)\varepsilon^T(\theta)\right\} = \Sigma(\theta,\lambda)$$

We **violate** the most important assumption made in least squares estimation: the residuals should not be correlated (i.e. should be orthogonal) with the estimated parameters.

We need a **fundamentally different** estimator approach!

This approach is **Maximum Likelihood Estimation** (MLE)

# Output Error & Maximum Likelihood

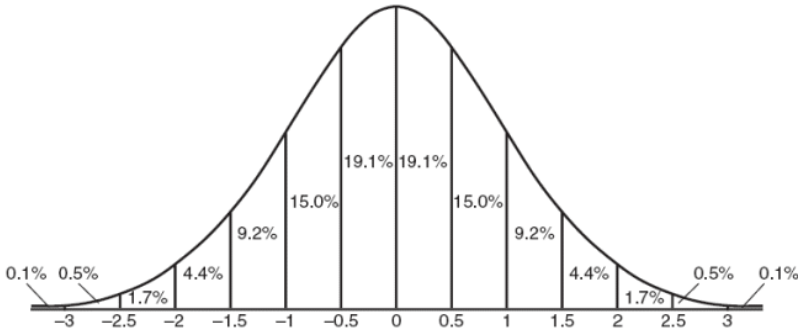
## The likelihood function

We have a set of observations $y(1), y(2), \ldots, y(N)$. Assume that this set of observations are **samplings** of an *unknown* probability density function (PDF) $f_0(\cdot)$ that is parameterized as follows:

$$f_0(\bullet) = f_0(\bullet \mid \theta_0)$$

Now we want to **maximize the likelihood** of observing $y(1), y(2), \ldots, y(N)$, given the **unknown** parameter $\theta_0$, by trying to find an estimator $\hat{\theta}$ that is as close to $\theta_0$ as possible.

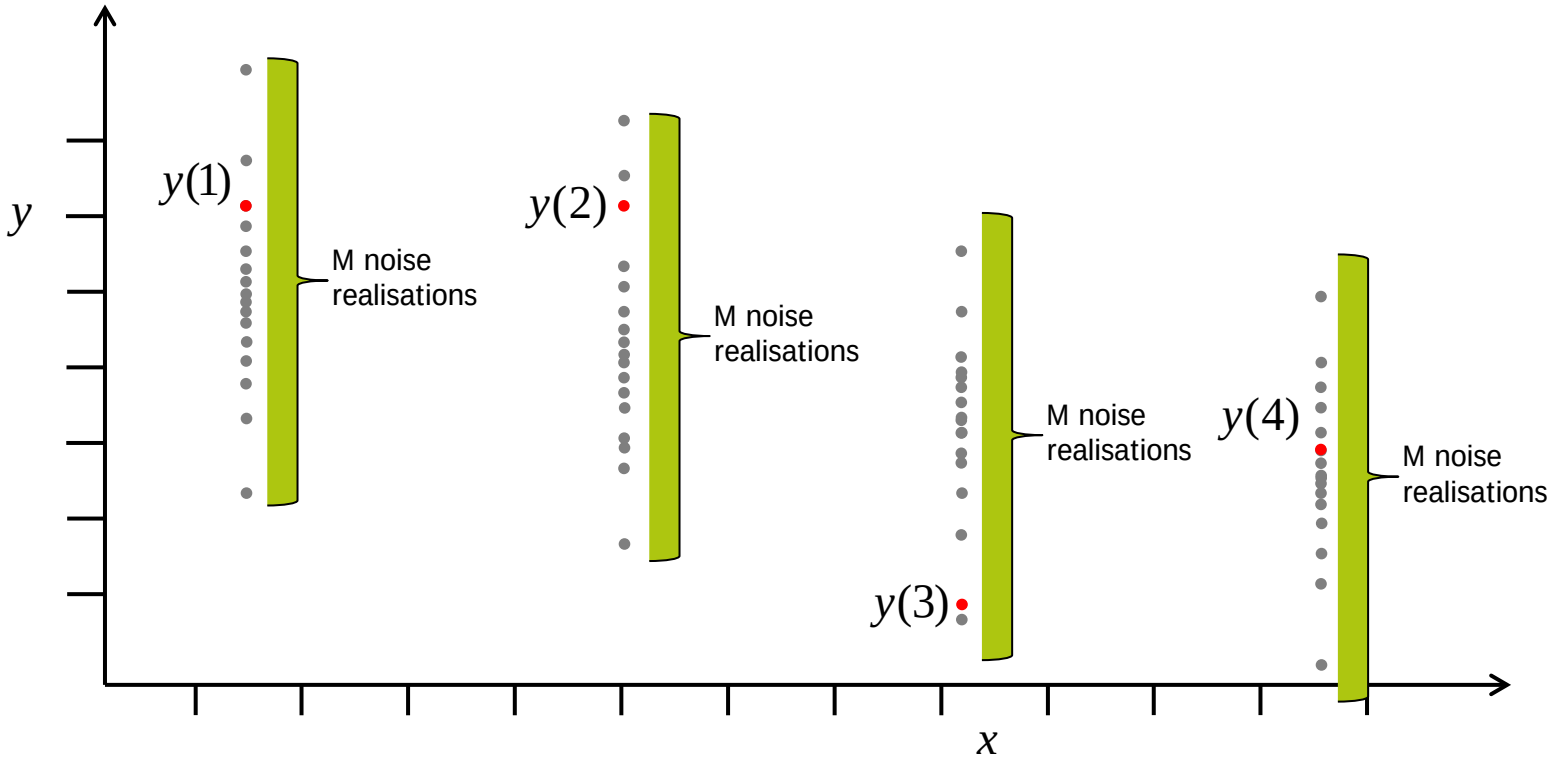
Example of PDF: Normal Distribution




$$f_N(y, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

# Output Error & Maximum Likelihood

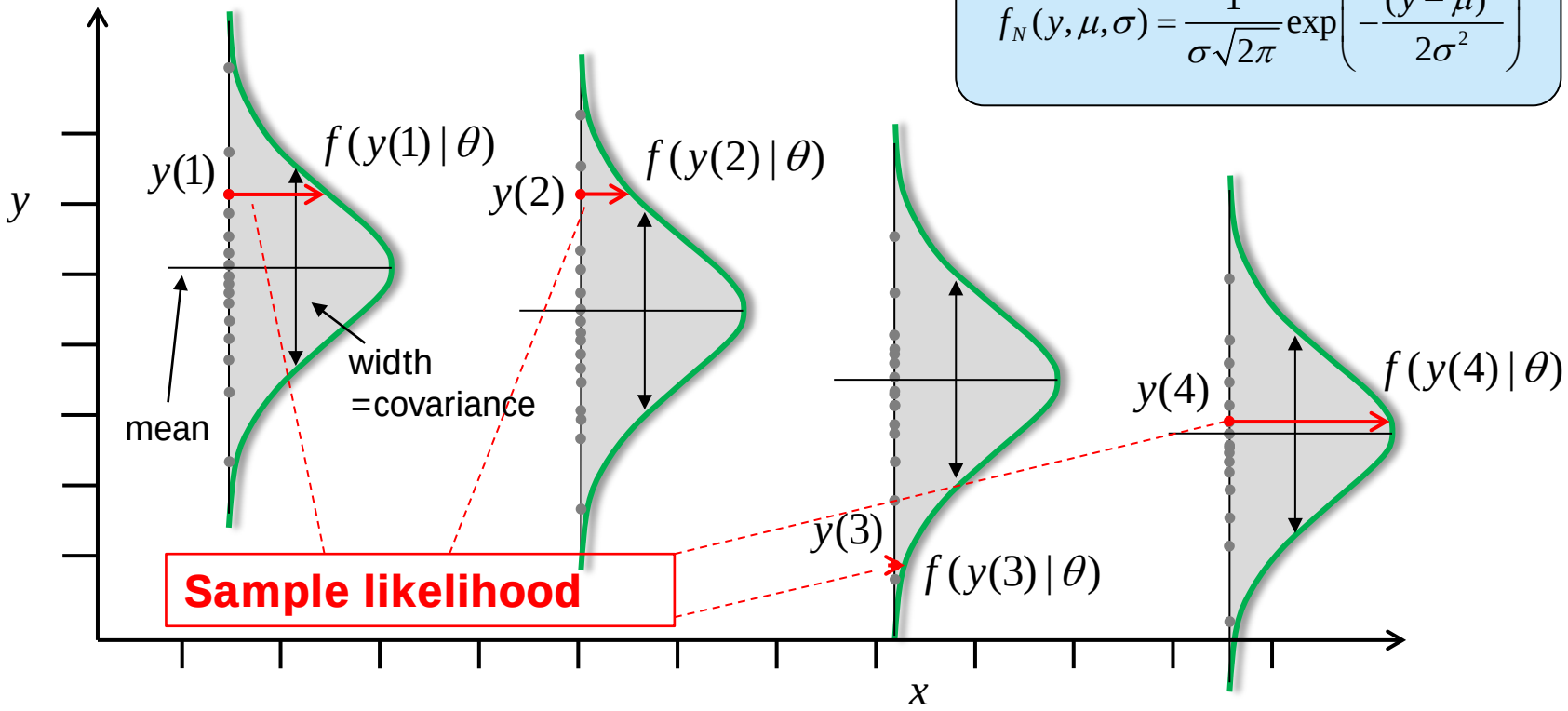
Realize: a measurement is just a sample...



# Output Error & Maximum Likelihood

## MLE: approximating the real likelihood function

$$f_N(y,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$




**Note:** the probability density functions (PDF) are **assumed** to have the same shape (e.g. normal distribution) and width (covariance), but distribution mean depends on $x$!

# Output Error & Maximum Likelihood

## The Likelihood Function

We define a new joint probability density function which approximates $f_0$ :

$$f_0(\bullet|\,\theta_0) \approx f\,(y(1), y(2), \ldots, y(N)\,|\,\theta)$$

If the observations are **independent and identically distributed**, we are allowed to use **Bayes rule** on the joint PDF:

$$\begin{aligned} f\,(y(1), y(2), \ldots, y(N)\,|\,\theta) &= f\,(y(1)\,|\,\theta) \cdot f\,(y(2)\,|\,\theta) \cdots f\,(y(N)\,|\,\theta) \\ &= \prod_{i=1}^{N} f\,(y(i)\,|\,\theta) \end{aligned}$$

This is known as the **Likelihood Function:**

$$L(y\,|\,\theta) = \prod_{i=1}^{N} f\,(y(i)\,|\,\theta)$$

# Output Error & Maximum Likelihood

## The Maximum Likelihood Estimator (MLE)

The Maximum Likelihood (ML) estimator maximizes the likelihood function:

$$\hat{\theta}_{ML} = \arg\max_{\theta} L(y \mid \theta)$$

$$= \arg\max_{\theta} \prod_{i=1}^{N} f\left(y(i) \mid \theta\right)$$

$$f_N(y,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

So how do we solve this for a *linear in the parameters* model structure?

We assume that $y$ is **normally distributed** with mean $A(x)\cdot\theta$ and (co)variance $\Sigma(\theta)$ and then find for $L(y|\theta)$:

$$L(y \mid \theta) = \frac{1}{2\pi^{N/2}\sqrt{\left|\Sigma(\theta)\right|}} \exp\left(-\frac{1}{2}\left(y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(y - A(x)\cdot\theta\right)\right)$$

standard deviation

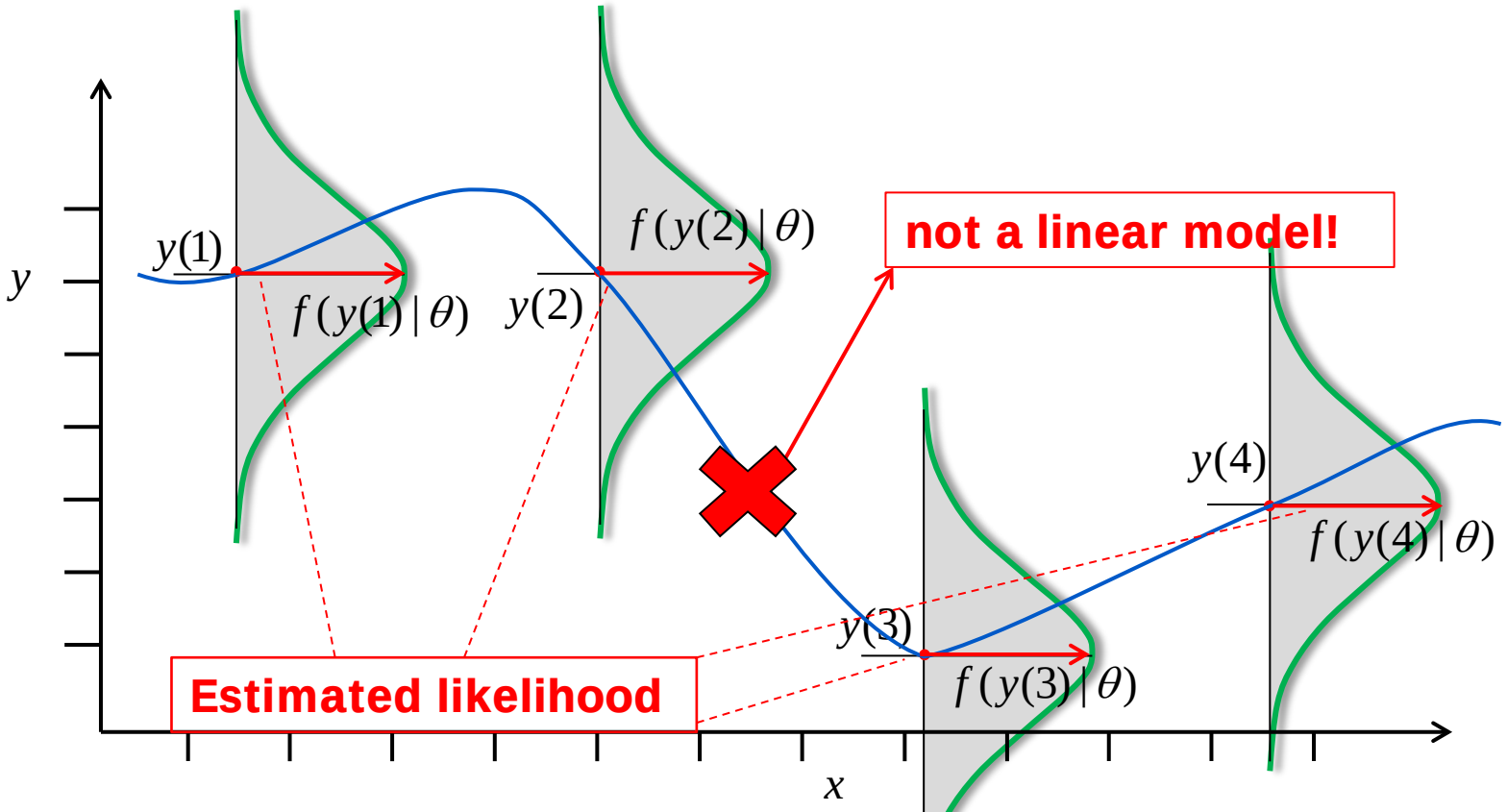
obser-vations

means

inverse of residual Covariance

# Output Error & Maximum Likelihood

Consider a MLE estimator for a **linear** model structure: $p(x,\theta) = \theta_0 + \theta_1 x$



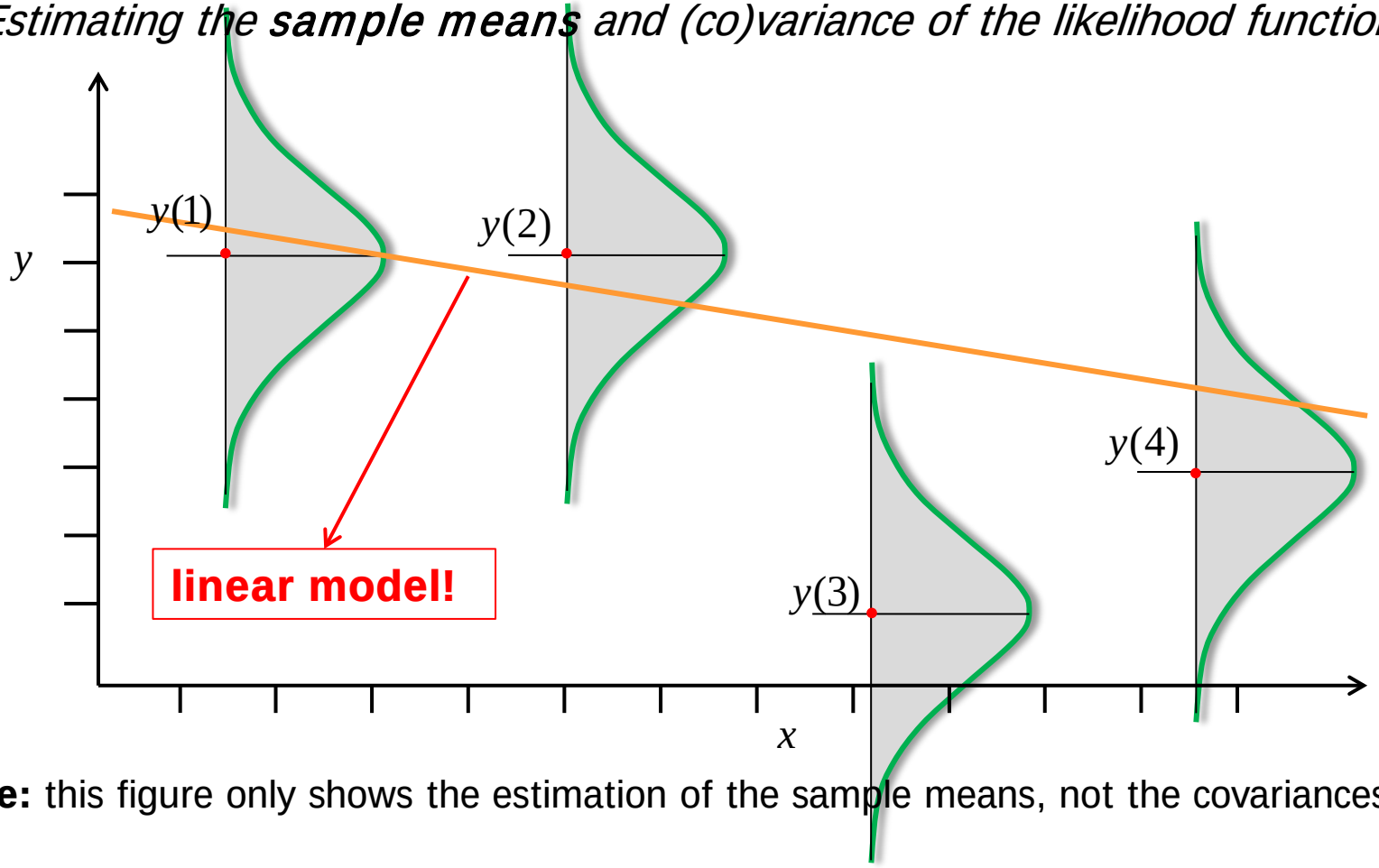

**Question:** So this selection of PDFs maximizes likelihood right?

**Answer:** Yes, but these sample means are **not possible** for a **linear** model structure...

# Output Error & Maximum Likelihood

Consider a MLE estimator for a **linear** model structure: $p(x,\theta) = \theta_0 + \theta_1 x$

*Estimating the **sample means** and (co)variance of the likelihood function*

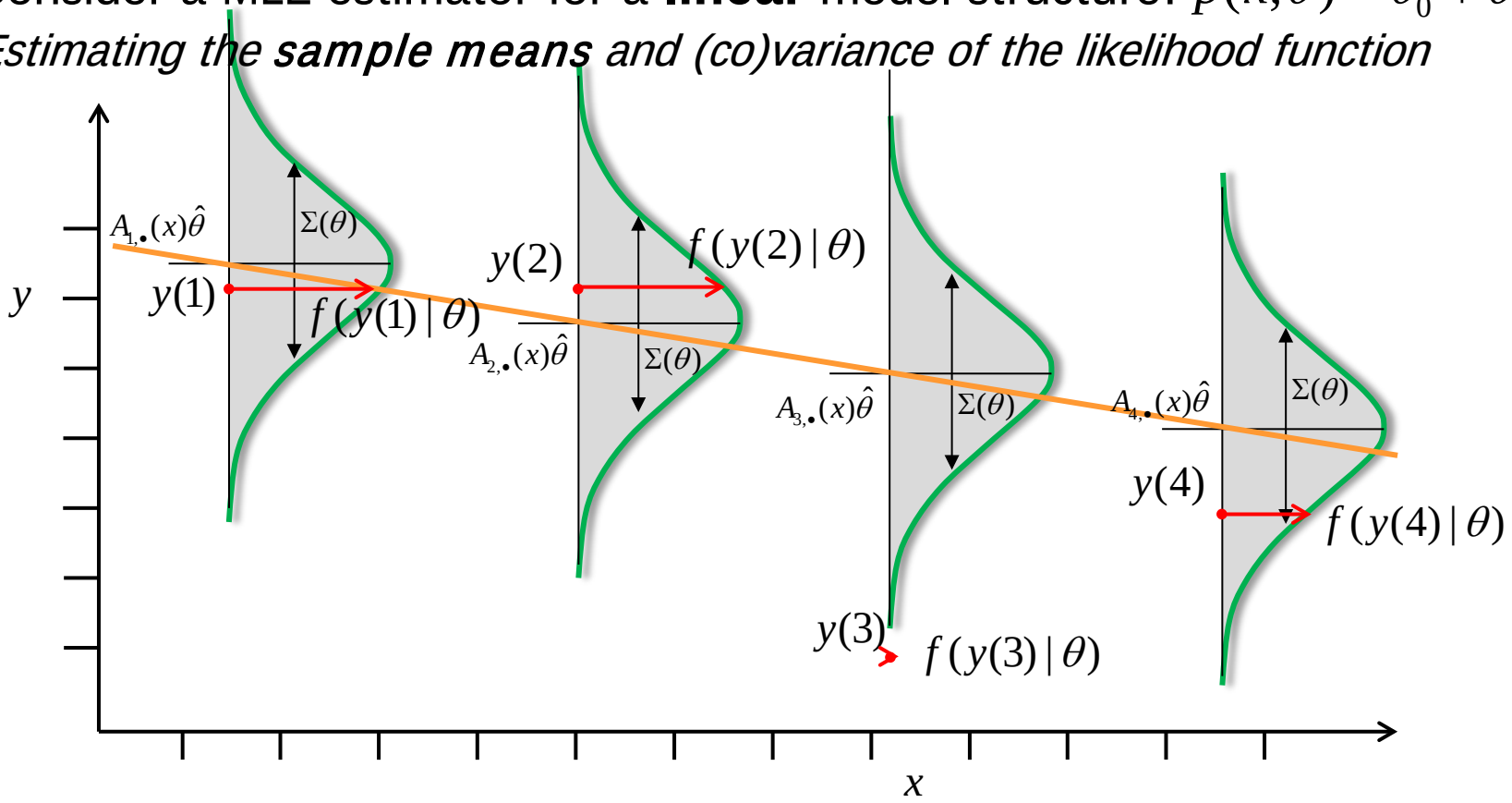



**Note:** this figure only shows the estimation of the sample means, not the covariances!

# Output Error & Maximum Likelihood

Consider a MLE estimator for a **linear** model structure: $p(x,\theta) = \theta_0 + \theta_1 x$

*Estimating the **sample means** and (co)variance of the likelihood function*




**Note:** the MLE estimator has tried to maximize the likelihood of the observations, given the linear model structure assumption that we made.

# Output Error & Maximum Likelihood

## The Maximum Likelihood Estimator

It is more convenient to work with the natural logarithm of the likelihood function...

$$\ln L(y \mid \theta) = \ln \frac{1}{2\pi^{N/2}\sqrt{|\Sigma(\theta)|}} \exp\left( -\frac{1}{2}\left(y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(y - A(x)\cdot\theta\right)\right)$$

$$= \ln(2\pi)^{N/2}\left|\Sigma(\theta)\right|^{1/2} + \frac{1}{2}\left(y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(y - A(x)\cdot\theta\right)$$

The maximum likelihood estimator is not influenced by the natural logarithm as it is monotonously increasing:

$$\hat{\theta}_{ML} = \arg\max_{\theta} L(y \mid \theta) = \arg\max_{\theta} \ln\left(L(\theta \mid y)\right)$$

$$= \arg\max_{\theta} \left[ \ln(2\pi)^{N/2}\left|\Sigma(\theta)\right|^{1/2} + \frac{1}{2}\left(y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(y - A(x)\cdot\theta\right)\right]$$

# Output Error & Maximum Likelihood

## The Maximum Likelihood Estimator

To solve a ML problem, we need to have some a-priori knowledge of $\Sigma(\theta)$ or we can use a multi-stage estimation approach to estimate it indirectly (more about this later...)

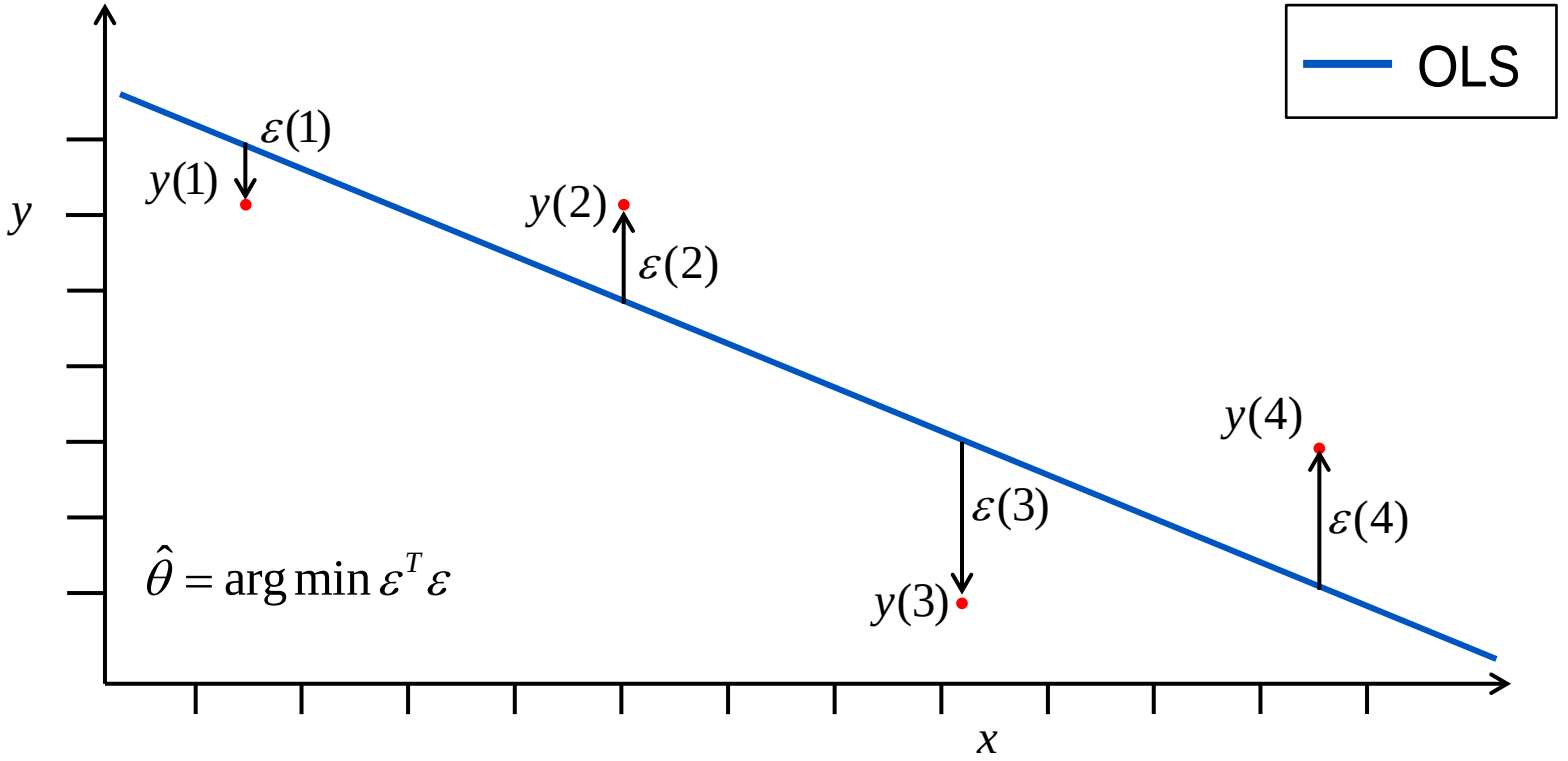
$$\hat{\theta}_{ML} = \arg\max_{\theta} \left[ \ln(2\pi)^{N/2} \left|\Sigma(\theta)\right|^{1/2} + \frac{1}{2}\left(y - A(x)\cdot\theta\right)^{T} \Sigma(\theta)^{-1}\left(y - A(x)\cdot\theta\right) \right]$$

Once we have a structure for $\Sigma(\theta)$ we can solve the above optimization problem using a **nonlinear solver** such as the Newton-Raphson (see Klein & Morelli, p185), Levenberg-Marquardt, Nelder-Mead Simplex, etc.

Many of these algorithms have standard Matlab implementations. For example, the `fminsearch` method uses the Nelder-Mead Simplex method to solve unconstrained nonlinear optimization problems, while `fmincon` uses SQP to solve constrained nonlinear optimization problems.

# Output Error & Maximum Likelihood

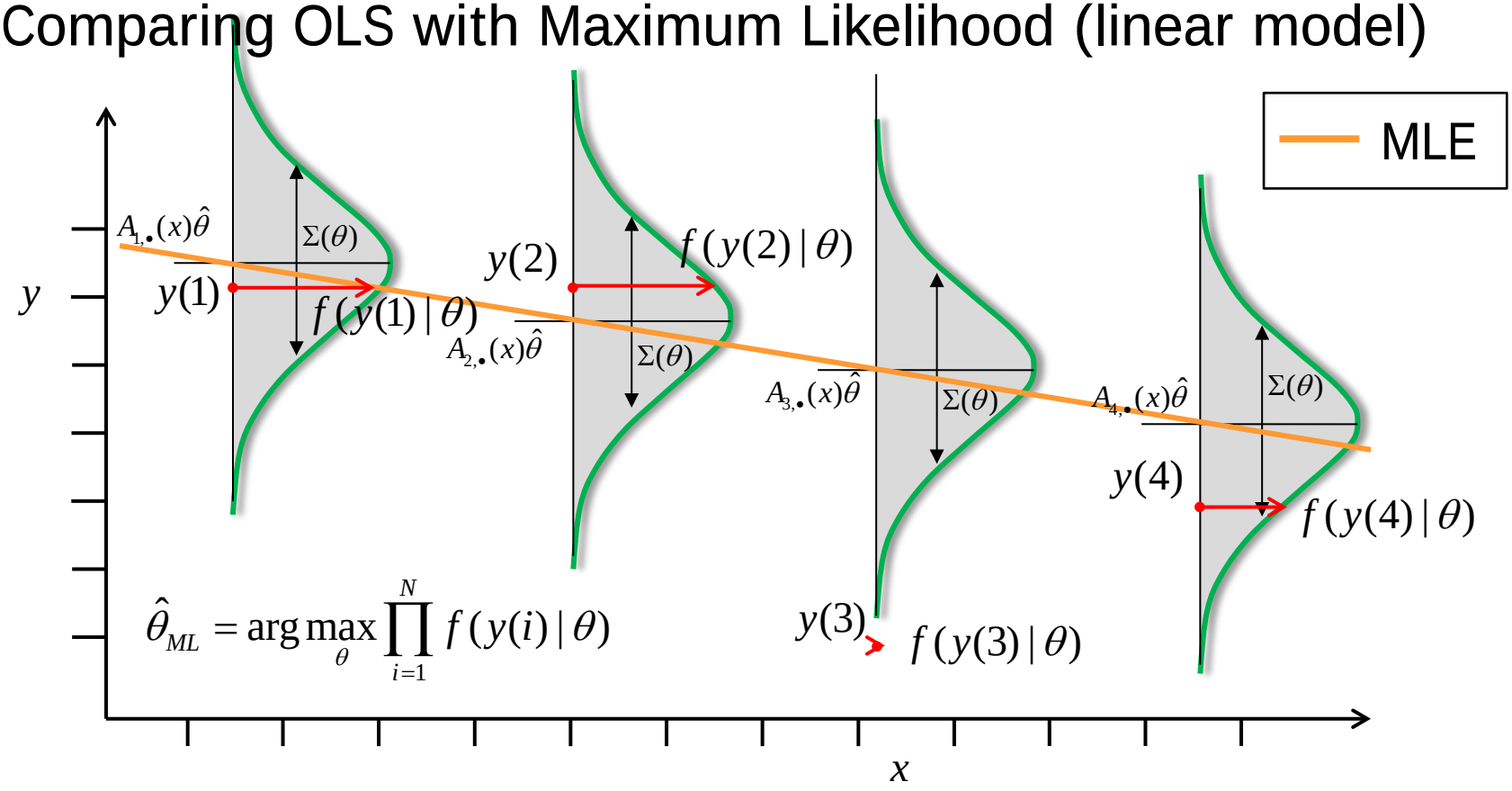
## Comparing OLS with Maximum Likelihood (linear model)




**OLS: Minimizing the sum of squared errors**

# Output Error & Maximum Likelihood

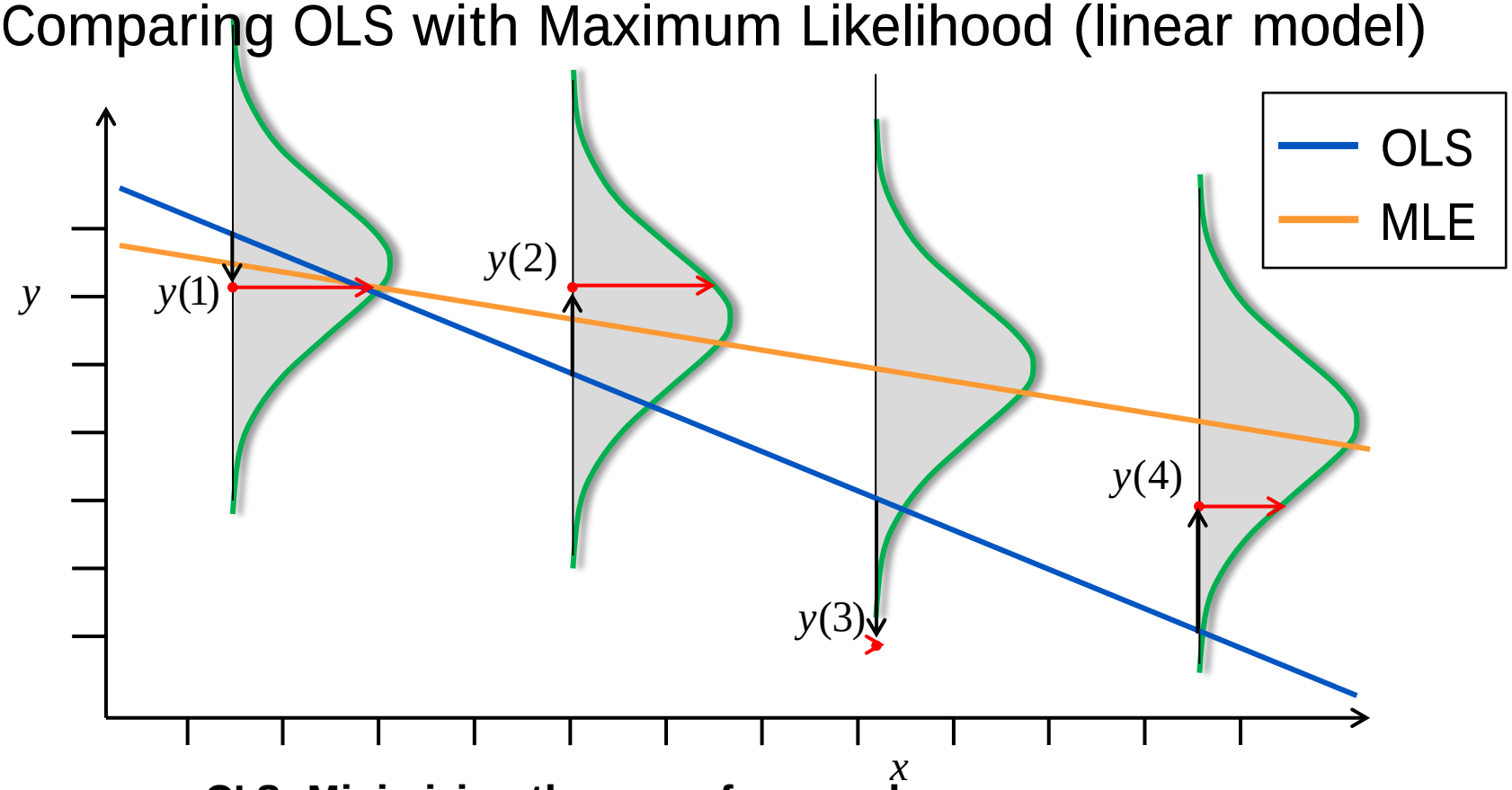
## Comparing OLS with Maximum Likelihood (linear model)




**MLE: Maximizing the product of likelihood functions**

# Output Error & Maximum Likelihood

Comparing OLS with Maximum Likelihood (linear model)




**OLS: Minimizing the sum of squared errors**

**MLE: Maximizing the product of likelihood functions**

# Output Error & Maximum Likelihood

## MLE with Relaxation Techniques

We can significantly simplify MLE by using a “Relaxation technique”. In that case, we approximate the residual covariance matrix $\hat{\Sigma}$ iteratively for each measurement point $k$:

$$\hat{\Sigma}(\hat{\theta}_k, k) = \frac{1}{N_k} \sum_{k=1}^{N_k} \Big[ y(\hat{\theta}_k, k) - p(\hat{\theta}_k, k) \Big] \Big[ y(\theta, k) - p(\hat{\theta}_k, k) \Big]^T$$

The new expression for the maximum likelihood estimator reduces to the iterative form:

$$\hat{\theta}_{k+1} = \arg\max_{\theta} \left[ \ln(2\pi)^{N/2} \left| \hat{\Sigma}(\hat{\theta}_k, k) \right|^{1/2} + \frac{1}{2} \Big( y(\hat{\theta}_k, k) - p(\hat{\theta}_k, k) \Big)^T \hat{\Sigma}(\hat{\theta}_k, k)^{-1} \Big( y(\hat{\theta}_k, k) - p(\hat{\theta}_k, k) \Big) \right]$$

**Note:** $p(\hat{\theta}_k, k) = A(x) \cdot \hat{\theta}_k$ in the case of a linear in the parameters model!

# Simplifications of Output Error

## Maximum Likelihood Estimator

Note that the general least squares estimator is a maximum likelihood estimator if $\Sigma$ is not a function of $\theta$ (uncorrelated residuals!):

$$\hat{\theta}_{ML} = \arg\max_{\theta} \left[ \ln(2\pi)^{N/2} \left|\Sigma(\theta)\right|^{1/2} + \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(Y - A(x)\cdot\theta\right) \right]$$

Which can be calculated as follows:

$$0 = \frac{\partial}{\partial\theta}\left( \ln(2\pi)^{N/2} \left|\Sigma\right|^{1/2} + \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T \Sigma^{-1}\left(Y - A(x)\cdot\theta\right) \right)$$

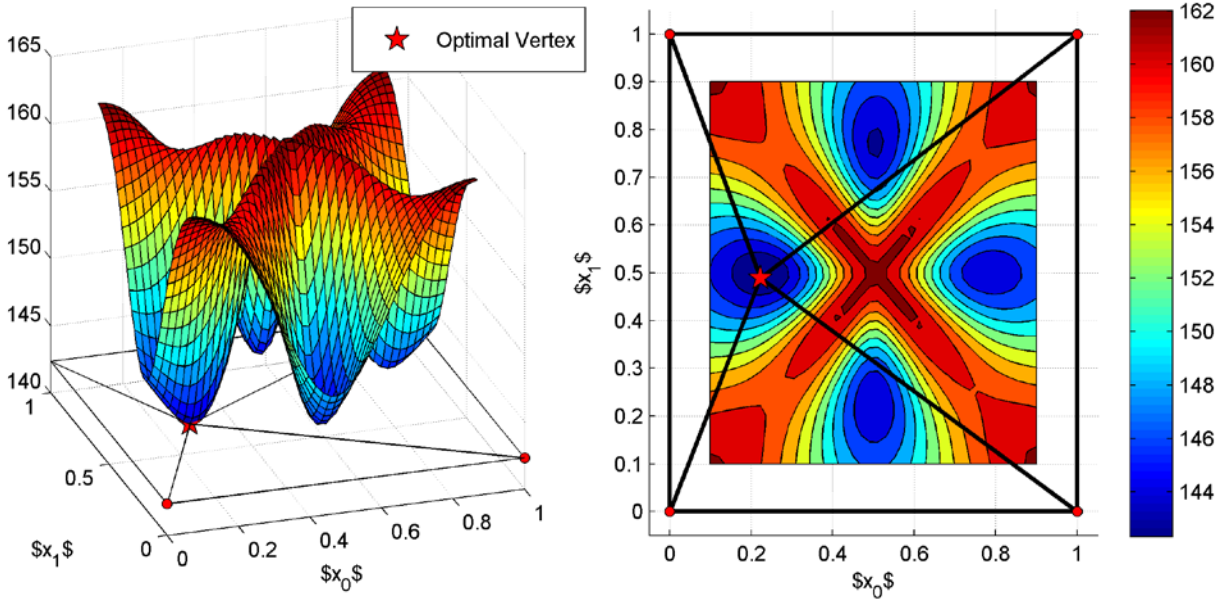
$$= \frac{\partial}{\partial\theta}\left( \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T \Sigma^{-1}\left(Y - A(x)\cdot\theta\right) \right)$$

Under further assumptions on $\Sigma$ this reduces to all other least squares estimators!

# Output Error & Maximum Likelihood

## The Maximum Likelihood Estimator

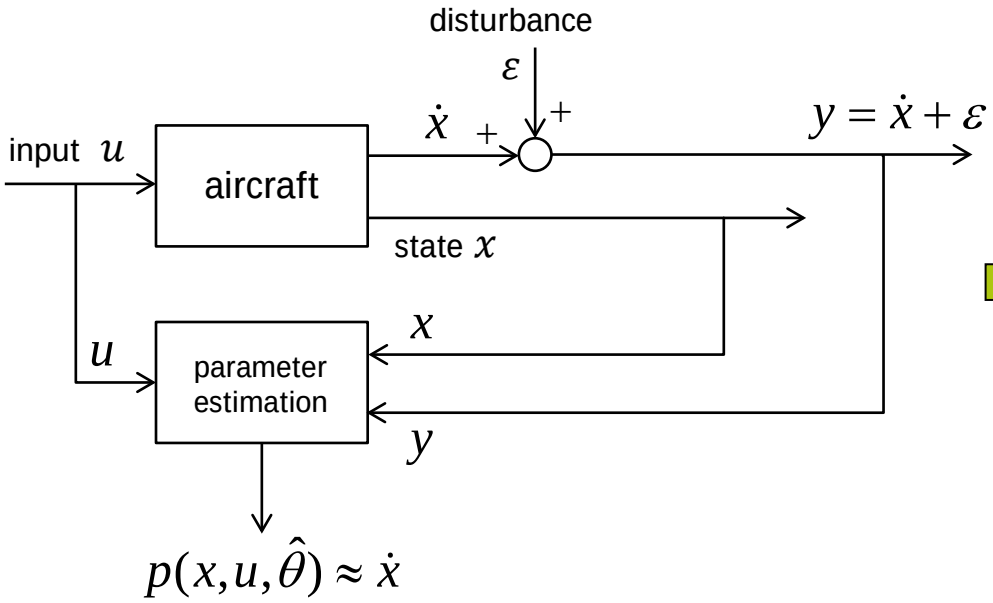
The Maximum Likelihood cost function can be non-convex…




Non-convex cost functions lead to **locally optimal solutions**. In that case, the only way of obtaining the global optimum is by using a Global Optimization method such as **Interval Analysis**.

# Parameter Estimation: Introduction

**Returning to the Equation Error** approach to parameter estimation:




**Note:**
no process noise
no disturbance on $x$

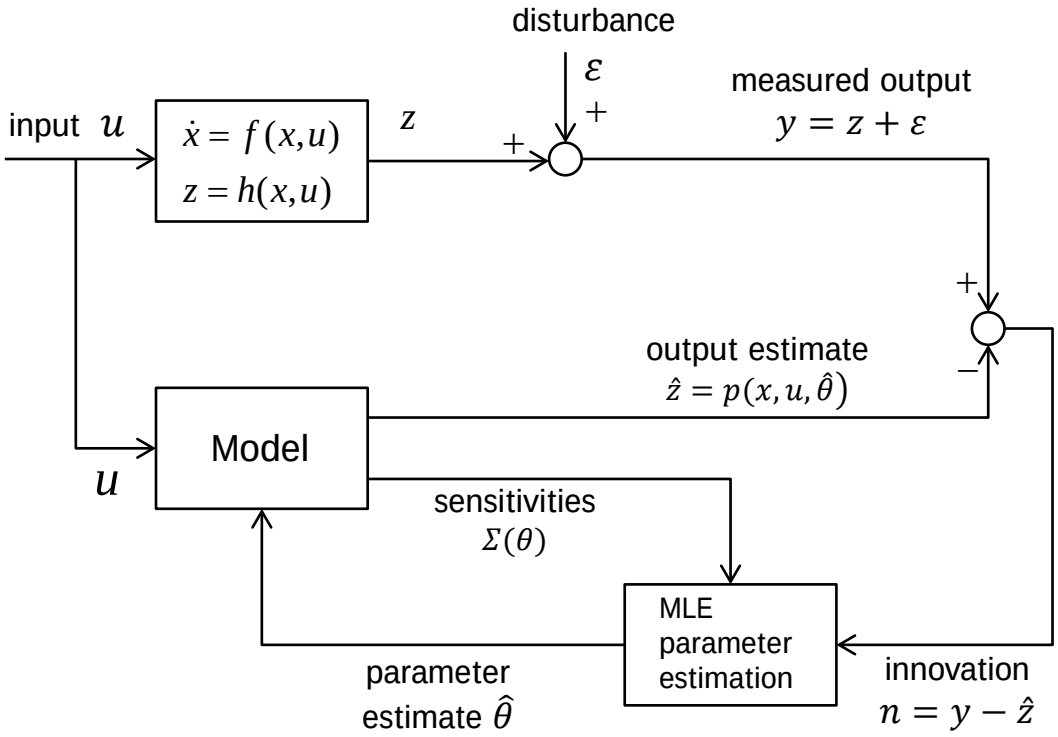
We assume we can measure the state derivative (plus noise): $y = \dot{x} + \varepsilon$.

We assume that we can measure the state $x$ without error!

The **Equation Error** is: $\varepsilon = y - p(x,u,\theta)$

# Output Error & Maximum Likelihood

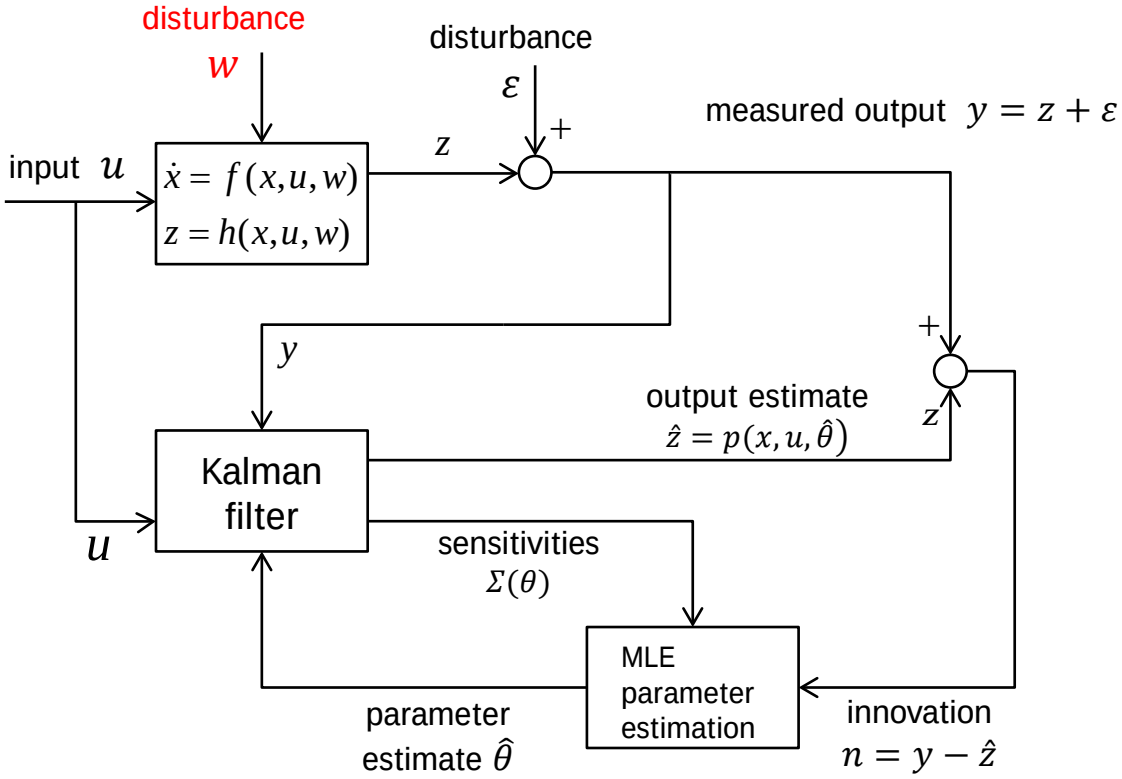
The MLE estimator can be used with the **Output Error** method:




Note that the OE method is iterative! The parameter estimator not only requires the innovation $n$, but also the measurement covariance matrix $\Sigma(\theta)$...

# Output Error vs. Filter Error

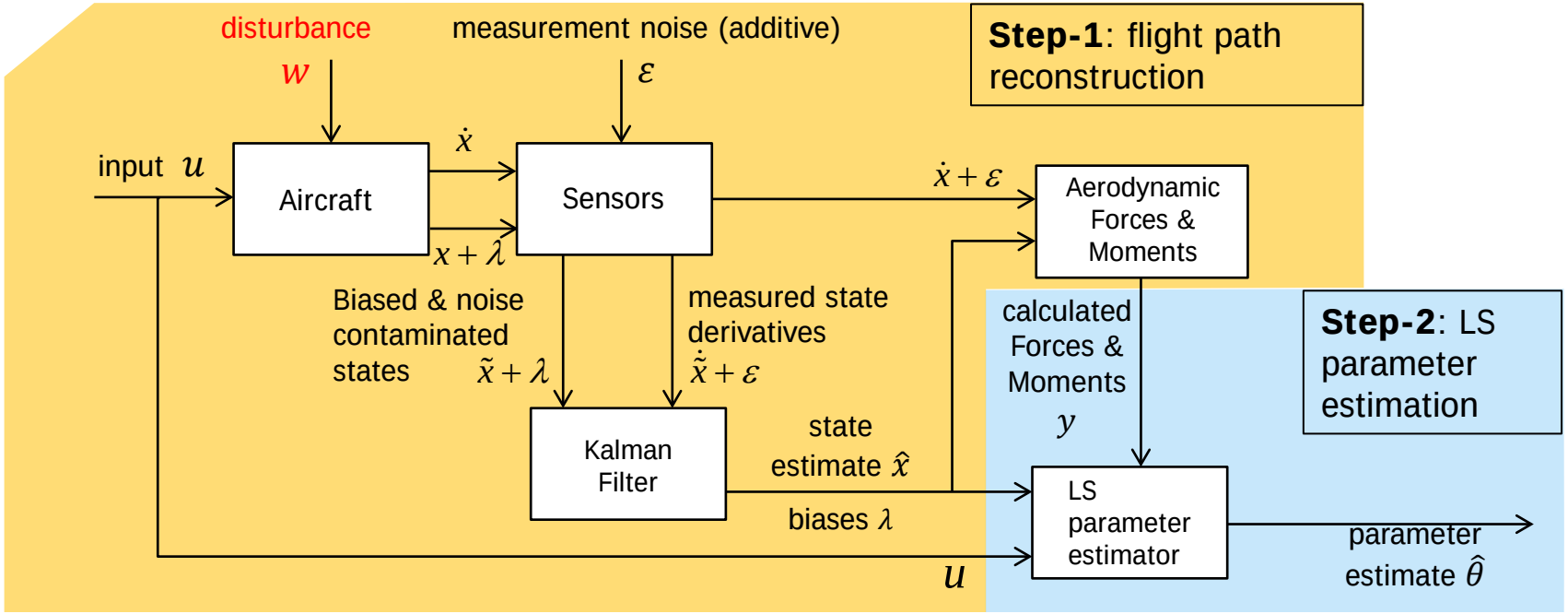
With process noise the **Filter Error** approach must be used:




The states are now assumed unknown/biased. Instead of a model producing the output, we now have a Kalman filter that is used to reconstruct the states and predict the model output.

# The Two-Step Method

The **Two-Step** method is a simplification of the Filter Error method:

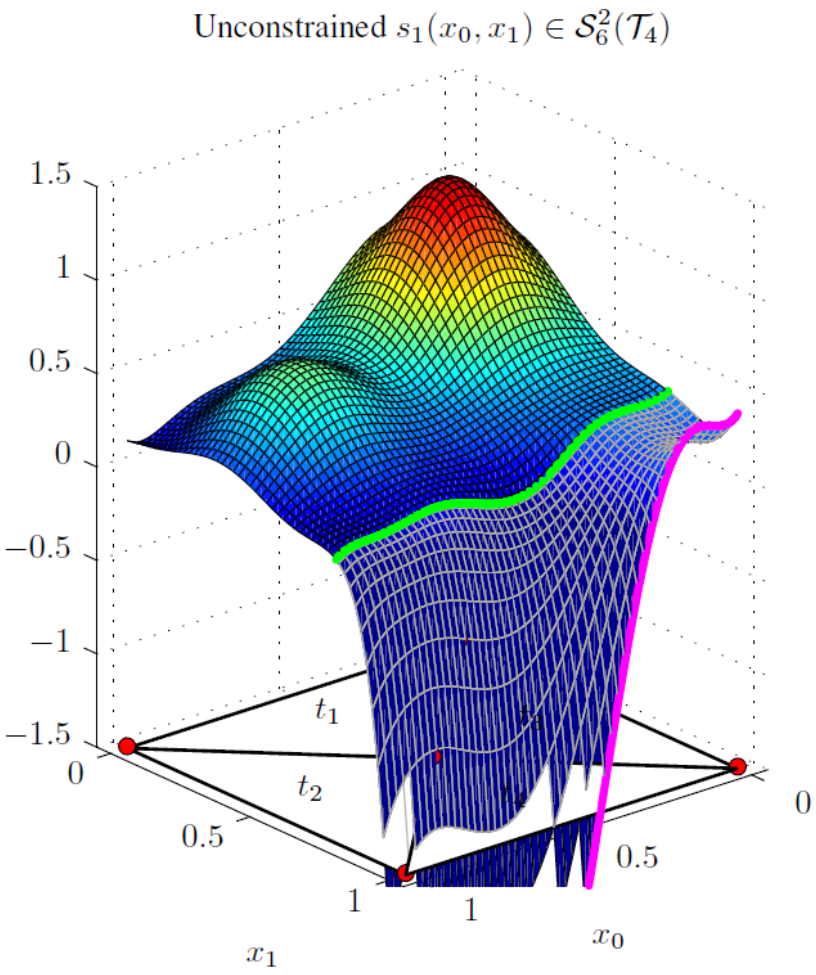


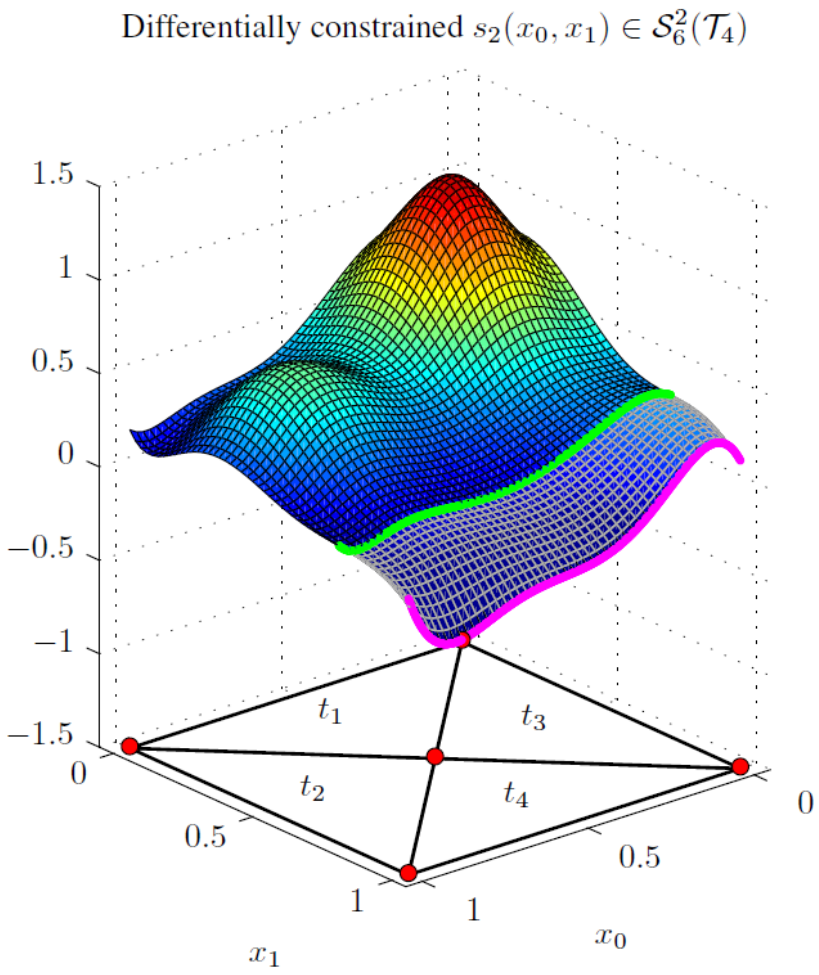

Assumption: we can directly **measure** state derivatives $\dot{x}$ (Accelerations!).

We use $\dot{x}$ together with estimated state $\hat{x}$ and biases $\lambda$ to calculate aerodynamic forces and moments: $y$

The **Equation Error** then is: $\varepsilon = y - A(\hat{x}, u)\theta$

# Constrained Least Squares Estimation



# Constrained Least Squares Estimation

## Equality constrained OLS estimation

Returning back to the OLS cost function:

$$J(x,\theta) = \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T\left(Y - A(x)\cdot\theta\right)$$

Now assume that the following constraint equation must hold for the estimated parameters: $H \cdot \theta = \mathrm{b}$.
The **constrained** cost function minimum can be found as follows:

$$\hat{\theta}_{ECOLS} = \arg\min_{\theta} \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T\left(Y - A(x)\cdot\theta\right), \quad \text{subject to } H\cdot\theta = b$$

This **equality constrained optimization** problem can be solved using many different methods.
Among the most powerful is the **Lagrange Multiplier method**...

# Constrained Least Squares Estimation

## Equality constrained OLS estimation

We want to solve the constrained OLS optimization problem:

$$\hat{\theta}_{ECOLS} = \arg\min_{\theta} \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T\left(Y - A(x)\cdot\theta\right), \quad \text{subject to } H\cdot\theta = b$$

For this we formulate a **Lagrangian** $\Gamma(\theta,\lambda)$ as follows:

$$\Gamma(\theta,\lambda) = \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T\left(Y - A(x)\cdot\theta\right) + \lambda^T\cdot H\cdot\theta$$

with $\lambda$ a vector of **Lagrangian multipliers**.

The constrained optimum is located at the location $(\theta,\lambda)$ for which the following partial derivatives are zero:

$$\frac{\partial\Gamma(\theta,\lambda)}{\partial\theta} = 0, \qquad \frac{\partial\Gamma(\theta,\lambda)}{\partial\lambda} = b$$

# Constrained Least Squares Estimation

## Equality constrained OLS estimation

We have the Lagrangian $\Gamma(\theta,\lambda)$:

$$\Gamma(\theta,\lambda)=\frac{1}{2}\left(Y-A(x)\cdot\theta\right)^T\left(Y-A(x)\cdot\theta\right)\ +\lambda^T\cdot H\cdot\theta$$

The partial derivatives of the Lagrangian are:

$$\frac{\partial\Gamma(\theta,\lambda)}{\partial\theta}=-\left(Y-A(x)\cdot\theta\right)^T\cdot A(x)+\hat{\lambda}^T\cdot H=0,$$

$$\frac{\partial\Gamma(\theta,\lambda)}{\partial\lambda}=H\cdot\theta=b$$

Some reorganizing leaves us with:

$$\frac{\partial\Gamma(\theta,\lambda)}{\partial\theta}=A^T(x)\cdot A(x)\cdot\theta-A^T(x)\cdot Y+H^T\cdot\hat{\lambda}=0,$$

$$\frac{\partial\Gamma(\theta,\lambda)}{\partial\lambda}=H\cdot\theta=b$$

# Constrained Least Squares Estimation

## Equality constrained OLS estimation

We can reformulate the partial derivatives as follows

$$A^T(x)\cdot A(x)\cdot\theta + H^T\cdot\hat{\lambda} = A^T(x)\cdot Y,$$
$$H\cdot\theta = b$$

Which can easily reformulated into the Karush-Kuhn-Tucker (KKT) matrix:

$$\begin{bmatrix} A^T(x)\cdot A(x) & H^T \\ H & 0 \end{bmatrix}\cdot\begin{bmatrix}\theta \\ \hat{\lambda}\end{bmatrix} = \begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

The constrained OLS estimator is:

$$\begin{bmatrix}\hat{\theta}_{ECOLS} \\ \hat{\lambda}\end{bmatrix} = \begin{bmatrix} A^T(x)\cdot A(x) & H^T \\ H & 0 \end{bmatrix}^{+}\cdot\begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

Note: Pseudo inverse!

# Constrained Least Squares Estimation

## Equality constrained OLS estimation

The OLS B-coefficient / Lagrangian estimator is:

$$\begin{bmatrix} \hat{\theta}_{ECOLS} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} A^T(x)\cdot A(x) & H^T \\ H & 0 \end{bmatrix}^+ \cdot \begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

This is equivalent to:

$$\begin{bmatrix} \hat{\theta}_{ECOLS} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} \cdot \begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

with $C_1$ the top-left $\mathrm{m} \times \mathrm{m}$ matrix block. The constrained OLS estimator is:

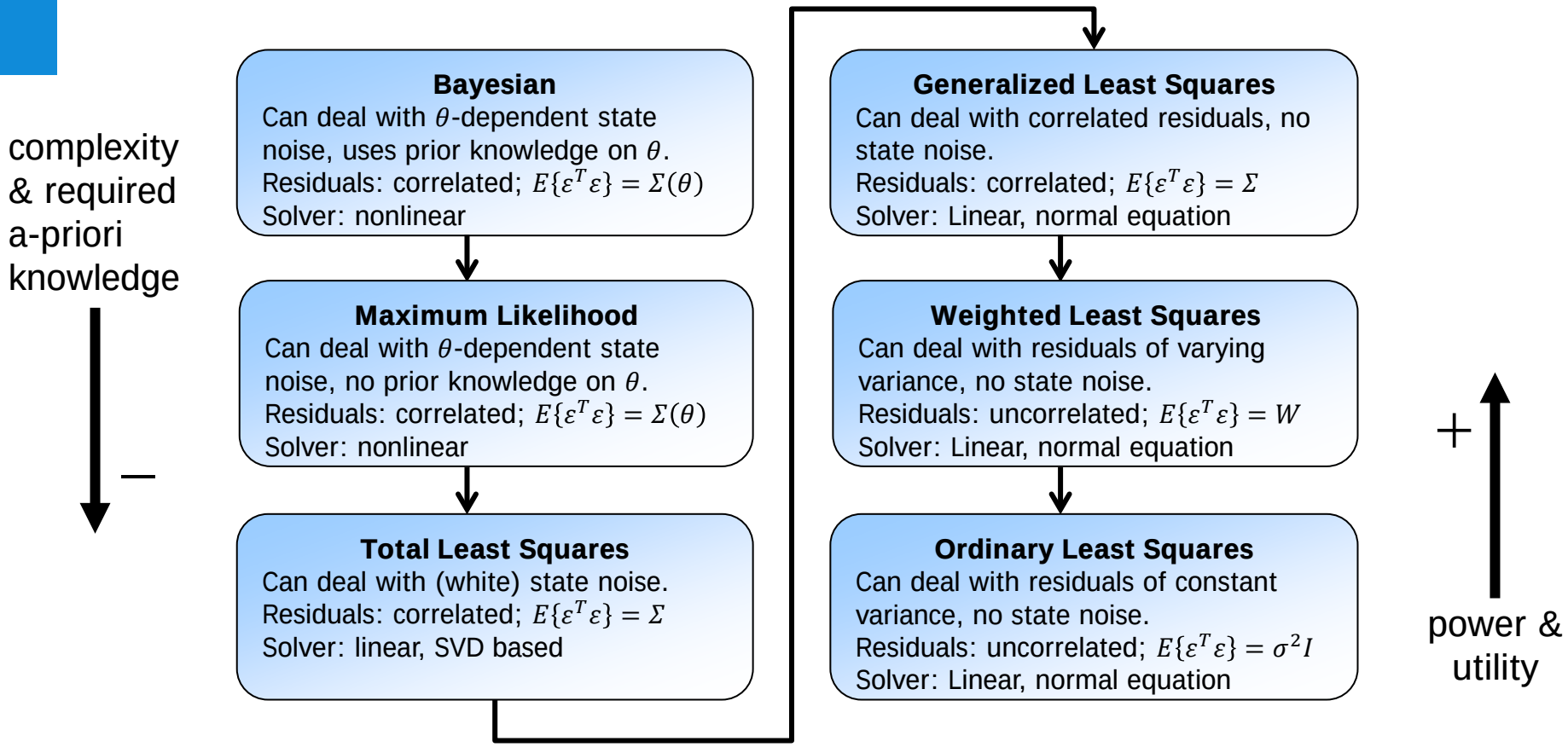
$$\hat{\theta}_{ECOLS} = C_1 \cdot A^T(x)\cdot Y + C_2 \cdot b$$

With statistics:

$$Cov\left\{\hat{\theta}_{ECOLS}\right\} = C_1, \quad Var\left\{\hat{\theta}_{ECOLS}\right\} = diag(C_1)$$

# Summary Parameter Estimators

## Properties of parameter estimators



# Recursive Least Squares Estimation

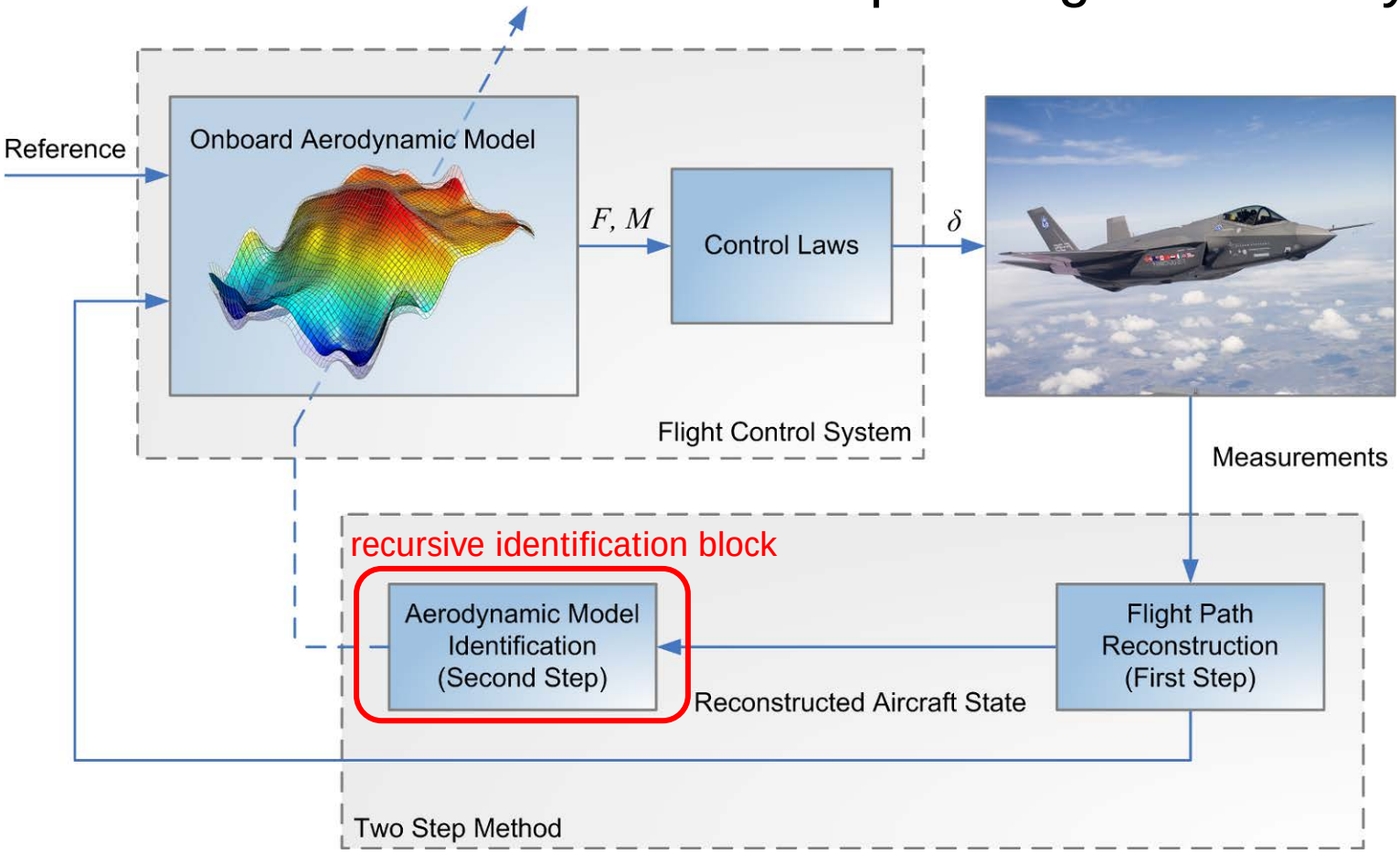
## Definition of recursive identification

**Definition 4.5: Recursive Identification**

Recursive identification is an online system identification process where new incoming data is used to update an existing model.

# Recursive Least Squares Estimation

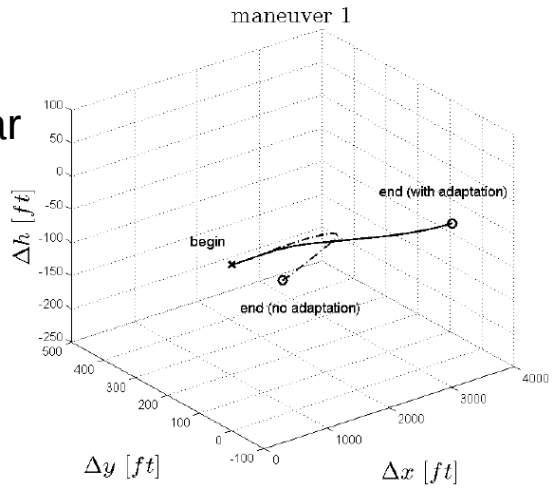
Recursive identification in an adaptive flight control system

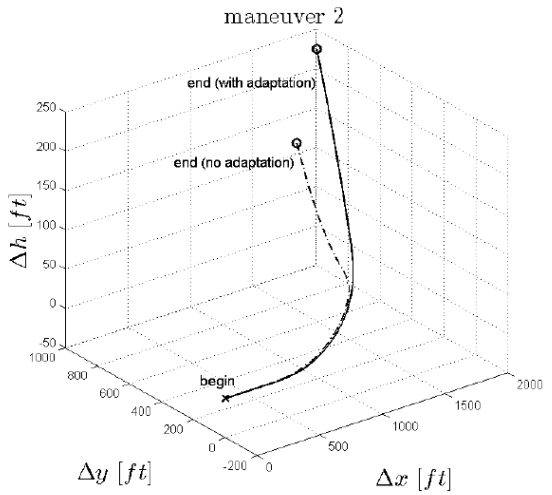


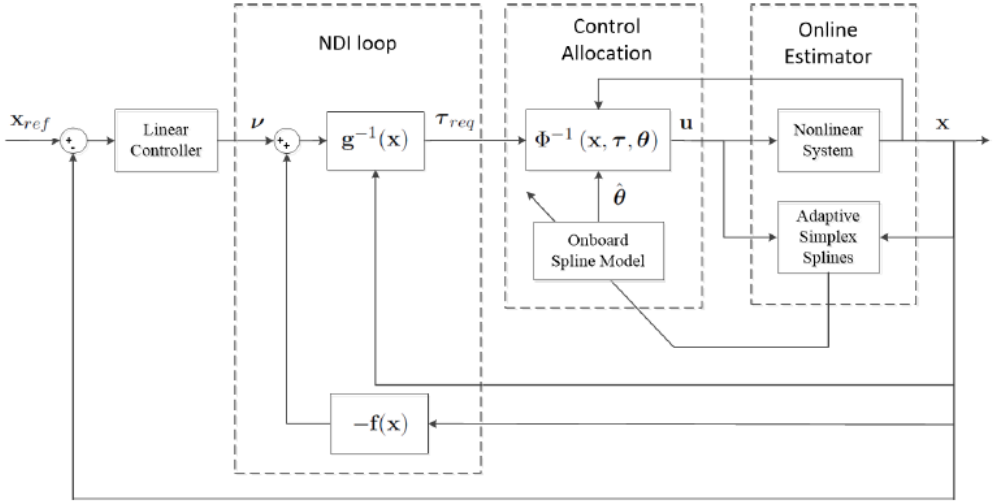
# Recursive Least Squares Estimation

**Example:**

- Spline Based Adaptive Nonlinear Dynamic Inversion (SANDI) Controller for the F-16.(*)
- Uses an Ordinary Recursive Least Squares Estimator with adaptive forgetting.




*(*)Tol, de Visser, Sun, van Kampen and Chu, Multivariate spline based adaptive control of high performance aircraft with aerodynamic uncertainties, AIAA Journal of Guidance, Control, and Dynamics, 2015*

# Recursive Least Squares Estimation

## Recursive Least Squares

- The most widely used recursive identification method is **recursive least squares (RLS)**.
- Recursive least squares is a modification of the ordinary/weighted least squares estimator into a form in which new data can be **efficiently** used to update an existing model.

**Matlab Demo**

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

First, we introduce the following notational simplifications for the regression matrix and regression matrix update:

$$A_k = A(x_k)$$

$$A_{k+1} = \begin{bmatrix} A_k \\ a_{k+1} \end{bmatrix}$$

$$a_{k+1} = a(x_{k+1})$$

The following simplified notation is introduced for the measurement vector update:

$$Y_k = \begin{bmatrix} y_1 & y_2 & \cdots & y_k \end{bmatrix}^T$$

$$Y_{k+1} = \begin{bmatrix} Y_k \\ y_{k+1} \end{bmatrix}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

We start with the quadratic cost function which can be split into a new part and an old part as follows:

$$J(x_{k+1},\theta_{k+1}) = \left(Y_{k+1} - A_{k+1}\cdot\theta_{k+1}\right)^T\left(Y_k - A_{k+1}\cdot\theta_{k+1}\right)$$

$$= \underbrace{\left(Y_k - A_k\cdot\theta_{k+1}\right)^T\left(Y_k - A_k\cdot\theta_{k+1}\right)}_{\text{old data}} + \underbrace{\left(y_{k+1} - a_{k+1}\cdot\theta_{k+1}\right)^T\left(y_{k+1} - a_{k+1}\cdot\theta_{k+1}\right)}_{\text{new data}}$$

The updated least squares estimator is then given by:

$$\hat{\theta}_{k+1} = \arg\min\left(J\left(x_{k+1},\theta_{k+1}\right)\right)$$

$$A_{k+1} = \begin{bmatrix} A_k \\ a_{k+1} \end{bmatrix},\quad Y_{k+1} = \begin{bmatrix} Y_k \\ y_{k+1} \end{bmatrix}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

The cost function has an optimum when its derivative towards $\theta_{k+1}$ is 0:

$$\frac{\partial J(x_{k+1},\theta_{k+1})}{\partial \theta_{k+1}} = -2\left(Y_k - A_k \cdot \hat{\theta}_{k+1}\right)^T \cdot A_k + -2\left(y_{k+1} - a_{k+1} \cdot \hat{\theta}_{k+1}\right)^T \cdot a_{k+1} = 0$$

This can be reformulated as:

$$0 = \left(Y_k{}^T \cdot A_k - \hat{\theta}_{k+1}^T \cdot A_k^T \cdot A_k\right) + \left(y_{k+1}^T \cdot a_{k+1} - \hat{\theta}_{k+1}^T \cdot a_{k+1}^T \cdot a_{k+1}\right)$$

$$= \left(A_k{}^T \cdot Y_k - A_k^T \cdot A_k \cdot \hat{\theta}_{k+1}\right) + \left(a_{k+1}^T \cdot y_{k+1} - a_{k+1}^T \cdot a_{k+1} \cdot \hat{\theta}_{k+1}\right)$$

take transpose!

Move all terms containing $\hat{\theta}_{k+1}$ to the left hand side:

$$\hat{\theta}_{k+1} \cdot \left(A_k^T \cdot A_k + a_{k+1}^T \cdot a_{k+1}\right) = A_k{}^T \cdot Y_k + a_{k+1}^T \cdot y_{k+1}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

We then find the following expression for the updated parameter estimator:

$$\hat{\theta}_{k+1} = \left( A_k^T \cdot A_k + a_{k+1}^T \cdot a_{k+1} \right)^{-1} \left( A_k{}^T \cdot Y_k + a_{k+1}^T \cdot y_{k+1} \right)$$

Now define the updated parameter covariance matrix as follows:

$$P_{k+1} = \left( A_k^T \cdot A_k + a_{k+1}^T \cdot a_{k+1} \right)^{-1}$$

Which allows us to simplify the updated parameter estimator as follows

$$\hat{\theta}_{k+1} = P_{k+1} \cdot \left( A_k{}^T \cdot Y_k + a_{k+1}^T \cdot y_{k+1} \right)$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

Observe that the expression for the updated parameter covariance matrix consists of an old and a new part:

$$P_{k+1} = \underbrace{\left( A_k^T \cdot A_k \right)}_{\text{old}} + \underbrace{a_{k+1}^T \cdot a_{k+1}}_{\text{new}} \Bigg)^{-1}$$

The old part is actually the **inverse** of the **old** parameter covariance matrix:

$$A_k^T \cdot A_k = P_k^{-1}$$

Thus, we can write the **inverse of the update** covariance matrix as follows:

$$\begin{aligned} P_{k+1}^{-1} &= A_k^T \cdot A_k + a_{k+1}^T \cdot a_{k+1} \\ &= P_k^{-1} + a_{k+1}^T \cdot a_{k+1} \end{aligned}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

The inverse of the updated parameter covariance matrix was found to be:

$$P_{k+1}^{-1} = P_k^{-1} + a_{k+1}^T \cdot a_{k+1}$$

This leads to alternative expression for the inverse of the **old** parameter covariance matrix:

$$P_k^{-1} = P_{k+1}^{-1} - a_{k+1}^T \cdot a_{k+1}$$

Trick!

We will now look for an expression for $\hat{\theta}_{k+1} = P_{k+1}\left(A_k{}^T \cdot Y_k + a_{k+1}^T \cdot y_{k+1}\right)$ that contains the 'old' parameter values $\hat{\theta}_k$. Remember the OLS estimator:

$$\hat{\theta}_k = P_k \cdot A_k{}^T \cdot Y_k$$

Right-multiplication with $P_k^{-1}$ results in:

$$P_k^{-1} \cdot \hat{\theta}_k = A_k{}^T \cdot Y_k$$

Trick!

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

Return now to the updated parameter estimator:

$$\hat{\theta}_{k+1} = \left( A_k^T \cdot A_k + a_{k+1}^T \cdot a_{k+1} \right)^{-1} \cdot \left( A_k{}^T \cdot Y_k + a_{k+1}^T \cdot y_{k+1} \right)$$
$$= P_{k+1} \cdot \left( P_k^{-1} \cdot \hat{\theta}_k + a_{k+1}^T \cdot y_{k+1} \right)$$

$$P_k^{-1} \hat{\theta}_k = A_k{}^T \cdot Y_k$$

Using the expression we found for $P_k^{-1}$ this can be reformulated as follows:

$$\hat{\theta}_{k+1} = P_{k+1} \cdot \left( P_{k+1}^{-1} \hat{\theta}_k - a_{k+1}^T \cdot a_{k+1} \cdot \hat{\theta}_k + a_{k+1}^T \cdot y_{k+1} \right)$$
$$= \hat{\theta}_k + P_{k+1} a_{k+1}^T \cdot y_{k+1} - P_{k+1} a_{k+1}^T \cdot a_{k+1} \cdot \hat{\theta}_k$$
$$= \hat{\theta}_k + P_{k+1} a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

$$P_k^{-1} = P_{k+1}^{-1} - a_{k+1}^T \cdot a_{k+1}$$

We have found a (partially) **recursive expression** for $\hat{\theta}_{k+1}$!

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

We have the following partially recursive expression for $\hat{\theta}_{k+1}$:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + P_{k+1} \cdot a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

We are not satisfied with this result yet (why?)

Recall the earlier derived expression for the updated covariance matrix $P_{k+1}$:

$$P_{k+1} = \left( P_k^{-1} + a_{k+1}^T \cdot a_{k+1} \right)^{-1}$$

We use the matrix inversion lemma to reformulate this expression:

$$\left[ A + BCD \right]^{-1} = A^{-1} - A^{-1} B \left[ DA^{-1} B + C^{-1} \right]^{-1} DA^{-1}$$

$$A = P_k^{-1}, \quad B = a_{k+1}^T, \quad C = I, \quad D = a_{k+1}$$

This results in:

$$\left( P_k^{-1} + a_{k+1}^T \cdot a_{k+1} \right)^{-1} = P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I^{-1} \right)^{-1} \cdot a_{k+1} \cdot P_k$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

scalar inverse!

The new expression for the updated parameter covariance matrix is:

$$P_{k+1} = \left( P_k^{-1} + a_{k+1}^T \cdot a_{k+1} \right)^{-1} = P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1} \cdot a_{k+1} \cdot P_k$$

substitute this result in the parameter update equation:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

We have now found a **true recursive expression** for $\hat{\theta}_{k+1}$ that now only depends on the "old" covariance matrix $P_k$ and parameter estimate $\hat{\theta}_k$ and the new regressor $a_{k+1}$ and measurement $y_{k+1}$.

We can further simplify this expression. For this we introduce a new gain matrix as follows:

$$K_{k+1} = P_k \cdot a_{k+1}^T \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

Recall the Kalman gain matrix from the state estimation lectures:

$$K_{k+1} = P_k \cdot H_{k+1}^T \cdot \left( H_{k+1} \cdot P_k \cdot H_{k+1}^T + R_k \right)^{-1}$$

Compare this with our new RLS Kalman gain (vector):

$$K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1}$$

output noise covariance matrix!

Substitute the RLS Kalman gain in the parameter update equation:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

Can we simplify this expression even further? Yes we can!

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

An expression for the parameter update equation with Kalman gain is:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

Observe that the expression for the RLS Kalman gain

$$K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1}$$

can be reformulated as follows:

$$\begin{aligned} P_k \cdot a_{k+1}^T &= K_{k+1} \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right) \\ &= K_{k+1} \cdot a_{k+1} \cdot P_k \cdot a_{k+1}^T + K_{k+1} \end{aligned}$$

We then find an new alternative (implicit) expression for the Kalman gain:

$$\begin{aligned} K_{k+1} &= P_k \cdot a_{k+1}^T - K_{k+1} \cdot a_{k+1} \cdot P_k \cdot a_{k+1}^T \\ &= \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \end{aligned}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

Using the new expression for the Kalman gain:

$$K_{k+1} = \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T$$

we can further simplify the expression for the parameter update equation

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot \left( y_{k+1} - a_{k+1} \hat{\theta}_k \right)$$

into the final form:

$$\boxed{\hat{\theta}_{k+1} = \hat{\theta}_k + K_{k+1} \cdot \left( y_{k+1} - a_{k+1} \hat{\theta}_k \right)}$$

The only thing left to do now is to find a recursive expression for the updated parameter covariance matrix $P_{k+1}$ …

# Recursive Least Squares Estimation

## Derivation of the RLS estimator

The expression for the updated covariance matrix is:

$$P_{k+1} = P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1} \cdot a_{k+1} \cdot P_k$$

Again, we use the expression for the Kalman gain

$$K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + I \right)^{-1}$$

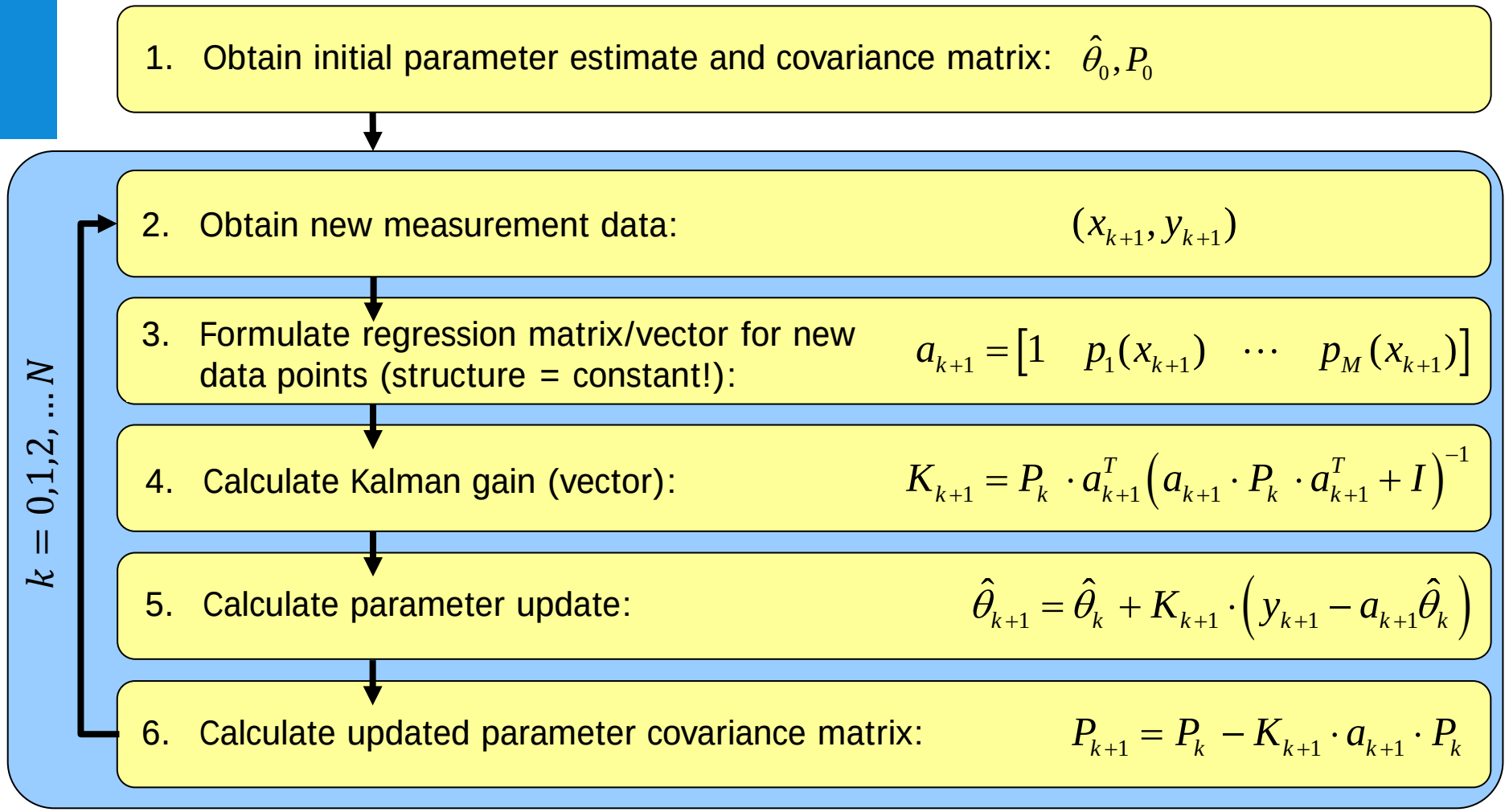
→ scalar inverse!

to simplify the covariance matrix update into its final form:

$$P_{k+1} = P_k - K_{k+1} \cdot a_{k+1} \cdot P_k$$

This completes the derivation of the **Recursive Least Squares** Estimator!

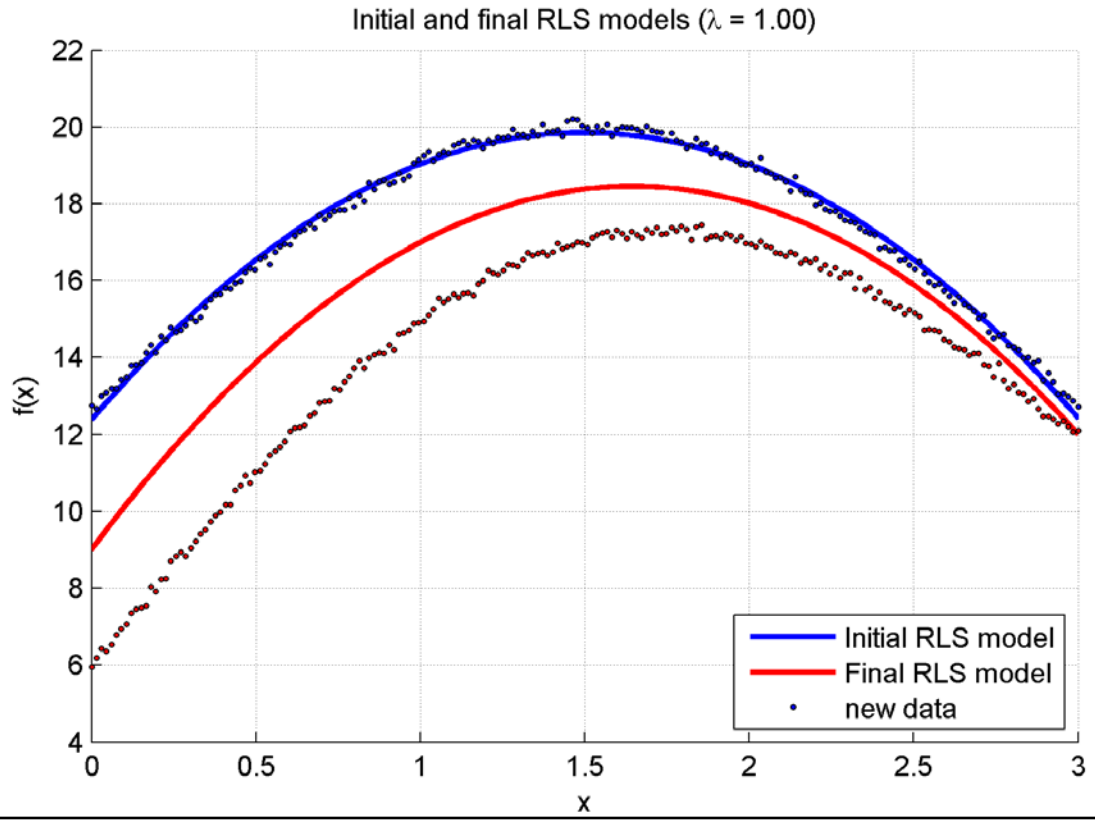
# Recursive Least Squares Estimation



# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

RLS estimator blends new data with old data, which may not be desirable...



# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

- One problem with the standard RLS estimator is that new data has the same weight as old data.
- One way to overcome this is to introduce a **forgetting factor** that acts as a weighting factor on new data.

For this, we include a weighting matrix $W_k^{-1}$ for the 'old' data, and a weighting matrix (vector) $w_{k+1}^{-1}$ for the new data in the quadratic cost function:

$$J(x_{k+1},\theta_{k+1}) = \left(Y_k - A_k \cdot \theta_{k+1}\right)^T \cdot W_k^{-1} \cdot \left(Y_k - A_k \cdot \theta_{k+1}\right) + \\ + \left(y_{k+1} - a_{k+1} \cdot \theta_{k+1}\right)^T \cdot w_{k+1}^{-1} \cdot \left(y_{k+1} - a_{k+1} \cdot \theta_{k+1}\right)$$

This leads to the following weighted parameter update equation:

$$\hat{\theta}_{k+1} = \left(A_k^T \cdot W_k^{-1} \cdot A_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1}\right)^{-1} \left(A_k^T \cdot W_k^{-1} \cdot Y_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot y_{k+1}\right)$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

We proceed with reformulating the weighted parameter update equation…

$$\hat{\theta}_{k+1} = \left( A_k^T \cdot W_k^{-1} \cdot A_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \right)^{-1} \left( A_k^T \cdot W_k^{-1} \cdot Y_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot y_{k+1} \right)$$

The weighted update covariance matrix is:

$$P_{k+1} = \left( A_k^T \cdot W_k^{-1} \cdot A_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \right)^{-1} = \left( P_k^{-1} + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \right)^{-1}$$

The weighted parameter update equation can now be reformulated in partial recurrent form:

$$\hat{\theta}_{k+1} = P_{k+1} \cdot \left( A_k^T \cdot W_k^{-1} \cdot Y_k + a_{k+1}^T \cdot y_{k+1} \right)$$

$$= P_{k+1} \cdot \left( P_k^{-1} \hat{\theta}_k + a_{k+1}^T \cdot y_{k+1} \right)$$

$$= P_{k+1} \cdot \left( P_{k+1}^{-1} \cdot \hat{\theta}_k - a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \cdot \hat{\theta}_k + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot y_{k+1} \right)$$

$$= \hat{\theta}_k + P_{k+1} \cdot a_{k+1}^T \cdot w_{k+1}^{-1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

$$A_k^T \cdot W_k^{-1} \cdot Y_k = P_k^{-1} \hat{\theta}_k$$

$$P_k^{-1} = P_{k+1}^{-1} - a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1}$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

We have the following partially recursive expression for $\hat{\theta}_{k+1}$:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + P_{k+1} \cdot a_{k+1}^T \cdot w_{k+1}^{-1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

Again, we use the matrix inversion lemma on the expression for $P_{k+1}$:

$$P_{k+1} = \left( P_k^{-1} + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \right)^{-1}$$

We now have for the matrix inversion lemma:

$$\left[A + BCD\right]^{-1} = A^{-1} - A^{-1}B\left[DA^{-1}B + C^{-1}\right]^{-1}DA^{-1}$$

$$A = P_k^{-1}, \quad B = a_{k+1}^T,$$
$$C = w_k^{-1}, \quad D = a_{k+1}$$

This results in a new expression for $P_{k+1}$:

$$P_{k+1} = \left( P_k^{-1} + a_{k+1}^T \cdot w_{k+1}^{-1} \cdot a_{k+1} \right)^{-1} = P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + w_{k+1} \right)^{-1} \cdot a_{k+1} \cdot P_k$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

Using the new expression for $P_{k+1}$ we find for $\hat{\theta}_{k+1}$:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + w_{k+1} \right)^{-1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot w_{k+1}^{-1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

We define the new Kalman gain as follows:

$$\boxed{K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + w_{k+1} \right)^{-1}}$$

Substitution in the weighted parameter update results in:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot w_{k+1}^{-1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

Note that we can (again) reformulate the expression for the Kalman gain:

$$K_{k+1} \cdot w_{k+1} + K_{k+1} \cdot a_{k+1} \cdot P_k \cdot a_{k+1}^T = P_k \cdot a_{k+1}^T$$

# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

The implicit relationship for the Kalman gain can be written as:

$$K_{k+1} = \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot w_{k+1}^{-1}$$

Using this result in the weighted parameter update equation

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left( P_k - K_{k+1} \cdot a_{k+1} \cdot P_k \right) \cdot a_{k+1}^T \cdot w_{k+1}^{-1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)$$

results in the final weighted RLS parameter estimator:

$$\boxed{\hat{\theta}_{k+1} = \hat{\theta}_k + K_{k+1} \cdot \left( y_{k+1} - a_{k+1} \cdot \hat{\theta}_k \right)}$$

Finally, the covariance matrix update is:

$$\boxed{P_{k+1} = P_k - K_{k+1} \cdot a_{k+1} \cdot P_k}$$

Weighted RLS differs from ordinary RLS only in the definition of the Kalman gain!

# Recursive Least Squares Estimation

## Derivation of the RLS estimator with Forgetting Factor

The new expression for the Kalman gain only contains the weight for the new data $(x_{k+1}, y_{k+1})$ in the form of $w_{k+1}$ :

$$K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + w_{k+1} \right)^{-1}$$

If we assume that $x_{k+1}$ and $y_{k+1}$ are scalars, then $a_{k+1} \cdot P_k \cdot a_{k+1}^T$ is also a scalar!

In that case $w_{k+1}$ must be a scalar, which we now call the **forgetting factor** indicated as $\lambda$, with $\lambda = 0$ being the maximum forget factor (infinite weight).

With the **forgetting factor** $\lambda$, the expression for the Kalman gain becomes:

$$K_{k+1} = P_k \cdot a_{k+1}^T \cdot \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + \lambda \right)^{-1}$$

# Recursive Least Squares Estimation

1. Obtain initial parameter estimate and covariance matrix, and determine the forgetting factor: $\hat{\theta}_0, P_0, \lambda > 0$

$k = 0,1,2,\ldots N$

2. Obtain new measurement data: $(x_{k+1}, y_{k+1})$

3. Formulate regression matrix/vector for new data points: $a_{k+1} = \begin{bmatrix} 1 & p_1(x_{k+1}) & \cdots & p_M(x_{k+1}) \end{bmatrix}$

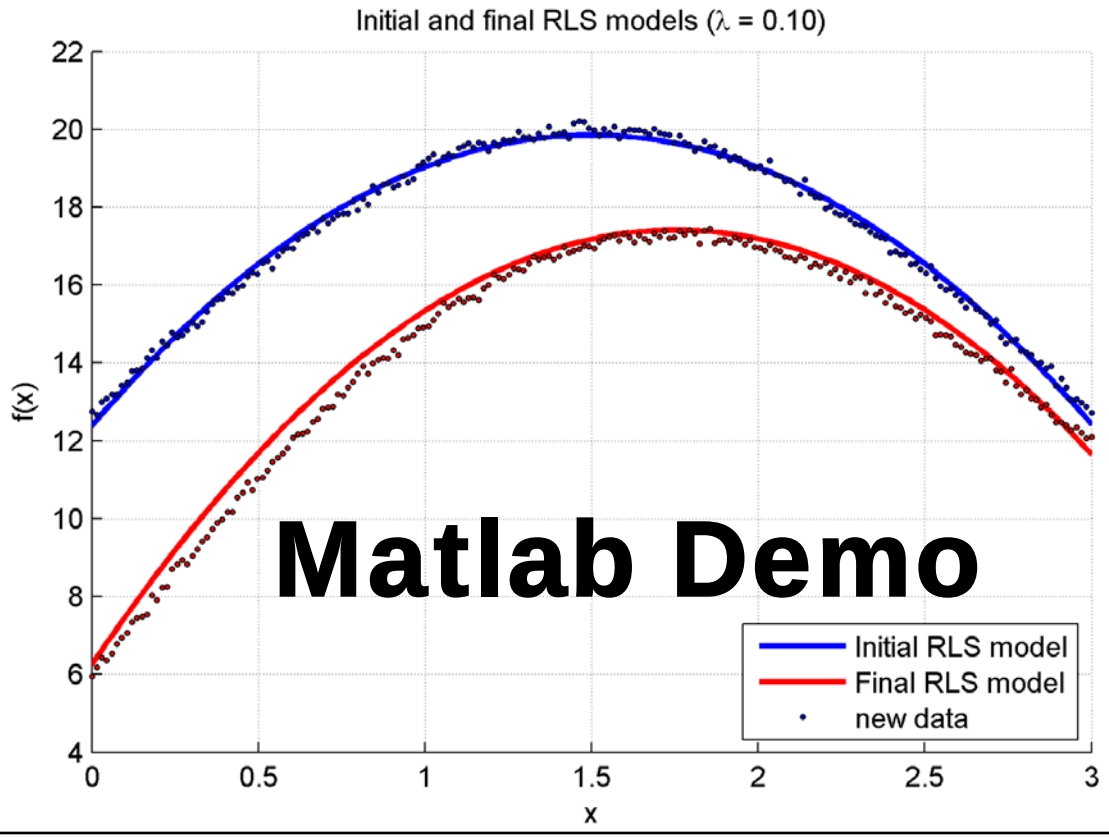
4. Calculate Kalman gain: $K_{k+1} = P_k \cdot a_{k+1}^T \left( a_{k+1} \cdot P_k \cdot a_{k+1}^T + \lambda \right)^{-1}$

5. Calculate parameter update: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_{k+1} \cdot \left( y_{k+1} - a_{k+1} \hat{\theta}_k \right)$

6. Calculate updated parameter covariance matrix: $P_{k+1} = P_k - K_{k+1} \cdot a_{k+1} \cdot P_k$

# Recursive Least Squares Estimation

Derivation of the RLS estimator with Forgetting Factor



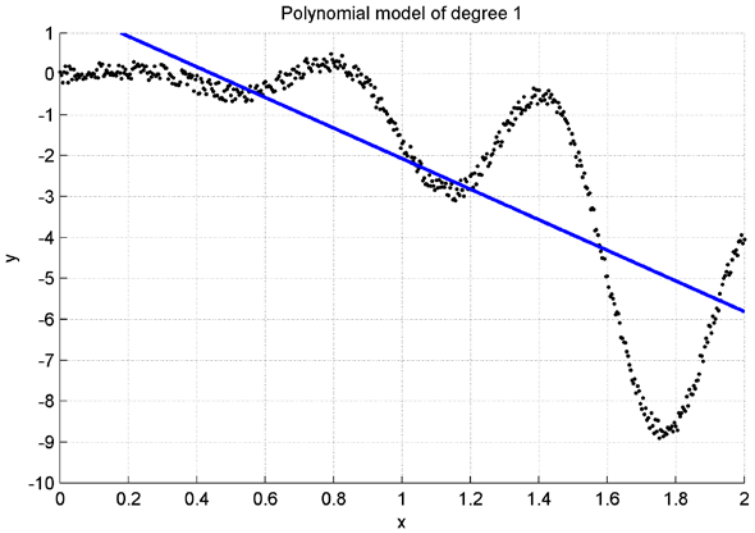
# Model Structure Selection

## A new concept: Approximation Power

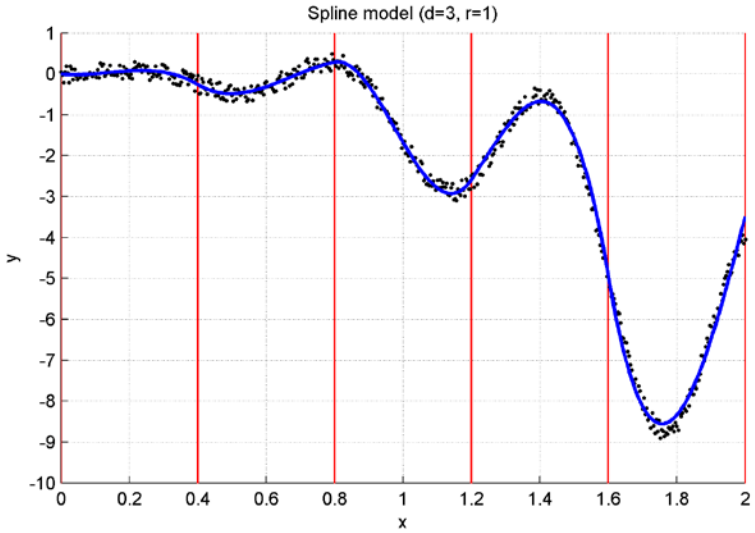
**Definition 4.4: Approximation Power**

Approximation power is the ability of a model $p(x, \theta)$ to approximate a function $f(x)$ and it derivatives $\frac{\partial^m f(x)}{\partial x^m}$ on a preset part of the domain of $f(x)$.

**Approximation power is a property of model structure!**




linear polynomial of the form $y = ax + b$




cubic spline of the form $y = B^d(x)c$

# Model Structure Selection

## Computational complexity

**Definition 4.5: Computational Complexity**

Computational complexity is an abstract, hardware independent measure for the amount of basic computational operations (i.e. $+, -, \times, \div$) necessary to compute a mathematical operation.

**Definition 4.6: Big-O notation**

Computational complexity can be expressed in "Big-O" notation: $O(n)$ means that an operations requires **approximately** $n$ basic computational operations.

**NOTE:**

Computational complexity is not the same as "compute time", which is a **hardware dependent** measure. However, the compute time of an operation can be estimated by dividing the Big-O number with the number of operations/second that a particular CPU can perform.

# Model Structure Selection

## Computational complexity

**Example 4.9:** Computational complexity of some basic operations:

Basic atomic operations on scalars with $n$ digits.

- Basic addition: $x + y \rightarrow O(n)$
- Basic multiply: $x * y \rightarrow O(n^2)$ (naïve), $O(n^{1.585})$ (Karatsuba)
- Basic division: $x/y \rightarrow O(n^2)$
- Power functions: $x^k \rightarrow O(kn^2)$ (naïve), $O(kn^{1.585})$ (Karatsuba)
- Trigonometric functions: $\sin x \rightarrow O(n^2 \log(n))$ (naïve), $O(n^{1.585} \log(n))$ (Karatsuba)

Basic floating point operations (FLOPS) on matrices and vectors

- Vector-Vector multiplication (VVM): $x \cdot x \rightarrow O(N), \quad x \in \mathbb{R}^N$
- Matrix-Vector multiplication (MVM): $A \cdot x \rightarrow O(N^2), \quad A \in \mathbb{R}^{N \times N}$
- Matrix-Matrix multiplication (MMM): $A \cdot A \rightarrow O(N^3)(\text{naive}), O(N^{2.3728639})$ (Le Gall), $A \in \mathbb{R}^{N \times N}$
- Matrix-Inverse: $A^{-1} \rightarrow O(N^3)(\text{naive}), O(N^{2.3728639})$ (Le Gall), $A \in \mathbb{R}^{N \times N}$

# Model Structure Selection

## Computational complexity

**Example 4.10:** Computational complexity of equation solvers

Let $N$ be the total number of parameters of a model. The following approximate computational complexity numbers can then be given for the most widely used equation solvers:

- Direct linear solver: $O(N^3)$
- Recursive linear solver: $O(N^2)$
- Gradient Descent solver (iterative): $O(k \cdot N^2), k \geq 2$
- Conjugate Gradient Descent solver (iterative): $O(k \cdot N^2), k \geq 2$
- Back-propagation (iterative): $O(N^c), c \geq 3$
- Branch & Bound (iterative): $O(c^N), c \geq 2$

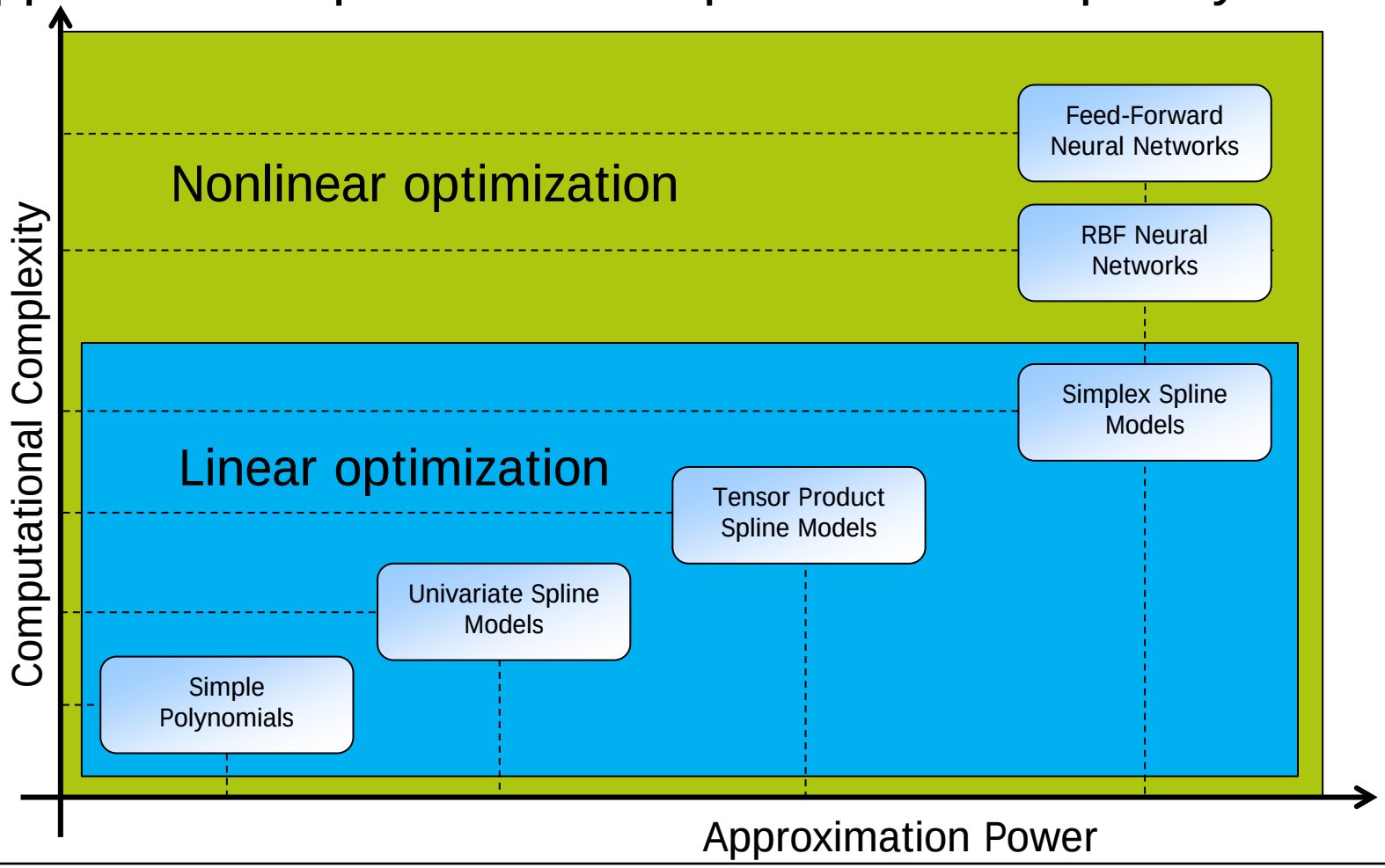
# Model Structure Selection

## Computational complexity

- The computational complexity of a parameter estimation method is strongly affected by the choice of equation solver.
- The choice of equation solver in turn depends on the chosen model structure. Some solvers cannot be used with some model structures. For example, direct linear solvers cannot be used with nonlinear-in-the-parameter models structures.
- In general, the computational complexity of parameter estimators using linear-in-the-parameter model structures is (much) lower than that of parameter estimators using nonlinear-in-the-parameter model structures.
- Computational complexity is a driving property of parameter estimators used in online applications (adaptive control!).

# Model Structure Selection

## Approximation power vs. Computational Complexity



# Summary Parameter Estimation

## Linear Estimators

Linear regression model:

$$Y = A(x) \cdot \theta + \varepsilon$$

Least squares estimator

$$\hat{\theta}_{OLS} = \left( A^T(x) A(x) \right)^{-1} A^T(x) Y \Rightarrow E(\varepsilon) = 0,\ E\left\{ \varepsilon \varepsilon^T \right\} = \sigma I$$

Weighted least squares estimator

$$\hat{\theta}_{WLS} = \left( A^T(x) W^{-1} A(x) \right)^{-1} A^T(x) W^{-1} Y \Rightarrow E(\varepsilon) = 0,\ E\left\{ \varepsilon \varepsilon^T \right\} = W$$

Generalized least squares estimator

$$\hat{\theta}_{GLS} = \left( A^T(x) \Sigma^{-1} A(x) \right)^{-1} A^T(x) \Sigma^{-1} Y \Rightarrow E(\varepsilon) = 0,\ E\left\{ \varepsilon \varepsilon^T \right\} = \Sigma$$

# Summary Parameter Estimation

## Constrained Linear Estimators

The equality constrained OLS optimization problem is formulated as:

$$\hat{\theta}_{ECOLS} = \arg\min_{\theta} \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T \left(Y - A(x)\cdot\theta\right), \quad \text{subject to } H\cdot\theta = b$$

Using Lagrange multipliers, this can be reformulated into the Karush-Kuhn-Tucker (KKT) matrix:

$$\begin{bmatrix} A^T(x)\cdot A(x) & H^T \\ H & 0 \end{bmatrix} \cdot \begin{bmatrix} \hat{\theta}_{ECOLS} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

The equality constrained OLS estimator then is:

$$\begin{bmatrix} \hat{\theta}_{ECOLS} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} A^T(x)\cdot A(x) & H^T \\ H & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} A^T(x)\cdot Y \\ b \end{bmatrix}$$

# Summary Parameter Estimation

## Nonlinear Estimators

Nonlinear regression model with correlated residuals:

$$Y = p(x,\theta) + \varepsilon(\theta)$$

Maximum Likelihood estimator

$$\hat{\theta}_{ML} = \arg\min_{\theta} L(x,\theta) \Rightarrow E(\varepsilon(\theta)) = 0,\ E\left\{\varepsilon(\theta)\varepsilon^T(\theta)\right\} = \Sigma(\theta)$$

With likelihood function $L(x,\theta)$:

$$L(x,\theta) = \prod_{i=1}^{N} f_i(Y_i \mid \theta)$$

Maximum likelihood estimator for normally distributed observations:

$$\hat{\theta}_{ML} = \arg\max_{\theta} \ln(2\pi)^{N/2}\left|\Sigma(\theta)\right|^{1/2} + \frac{1}{2}\left(Y - A(x)\cdot\theta\right)^T \Sigma(\theta)^{-1}\left(Y - A(x)\cdot\theta\right)$$

# Goals of this Lecture

***Questions that were answered during this lecture:***

1. *What is parameter estimation, and why do we need it?*
   - *Parameter estimation is concerned with estimating the parameters of a predetermined model structure given a set of data. Without parameter estimation, a model will remain an abstract model structure (class vs. class instance).*
2. *What is an estimator?*
   - *An estimator, indicated as $\hat{\theta}$, is the vector of parameters that minimize a given cost function.*
3. *What types of parameter estimators do we have and when do we use which type?*
   - *There are many different types of parameter estimators. The choice of estimator depends on the chosen model structure, the noise characteristics, and the intended application of the model.*

# Goals of this Lecture

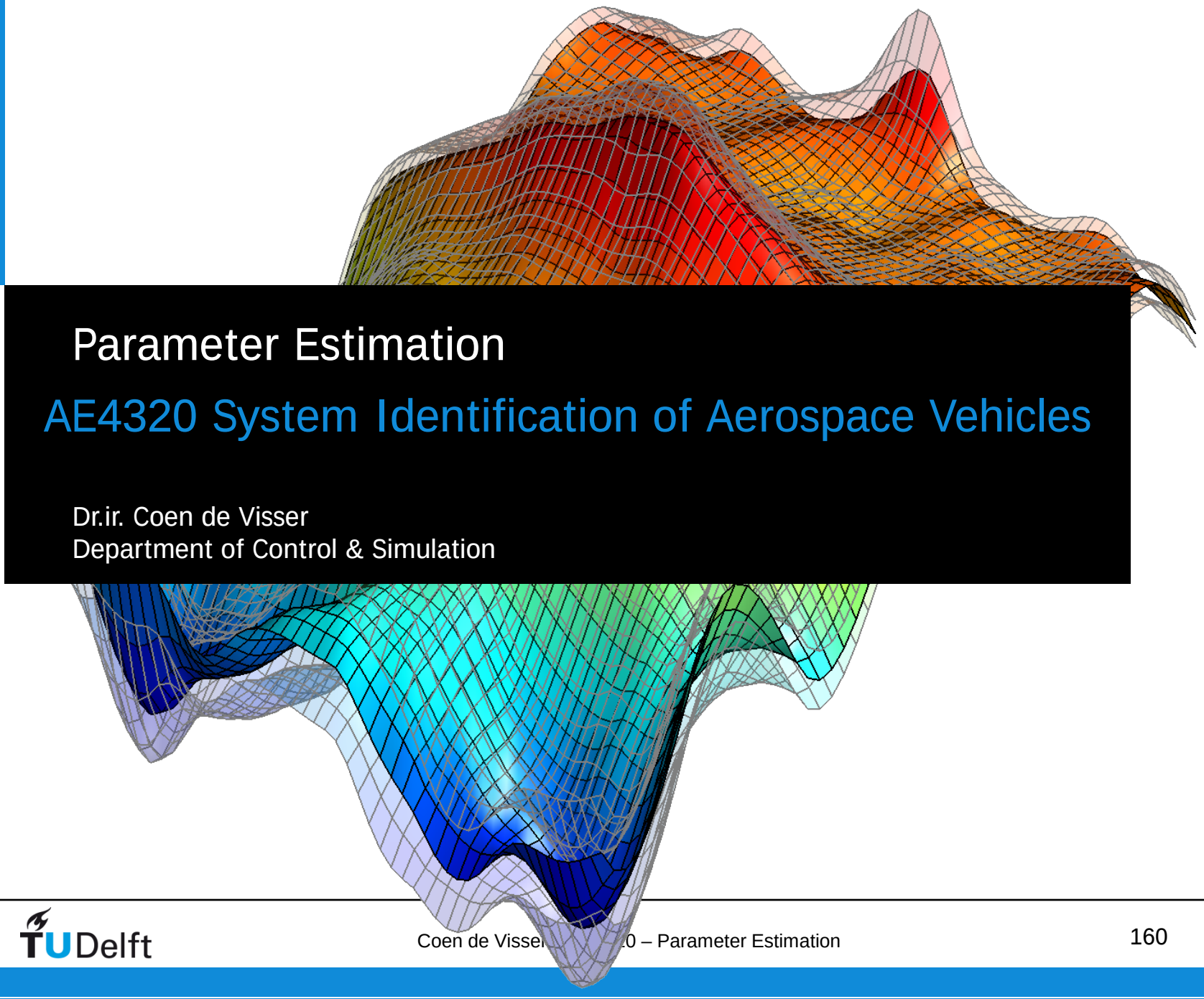
***Questions that were answered during this lecture:***

4. *What is linear regression?*
   - *Linear regression is fitting, in some optimal way, a linear-in-the-parameters model to a given set of data.*
5. *What are the assumptions made for the OLS, WLS, GLS, TLS, RLS, and MLE parameter estimators?*
   - **OLS**: no process noise, constant variance uncorrelated white residuals
   - **WLS**: no process noise, varying variance uncorrelated white residuals
   - **GLS**: no process noise, varying variance correlated white residuals
   - **TLS**: linear process noise, varying variance correlated white residuals
   - **MLE**: nonlinear process noise, varying variance correlated white residuals
   - **RLS**: recursive (online) estimator, no process noise, varying variance uncorrelated white residuals.

# Goals of this Lecture

***Questions that were answered during this lecture:***

6. *How do we obtain the optimal set of parameters given a particular optimization problem?*
   - *We solve an optimization problem in terms of the unknown parameters. There are many different types of solvers; single step (convex) solvers, gradient descent, back-propagation, branch & bound, dual ascent, etc. The choice of solver depends on the optimisation problem type and the intended application of the model.*



# Parameter Estimation

## AE4320 System Identification of Aerospace Vehicles

Dr.ir. Coen de Visser
Department of Control & Simulation