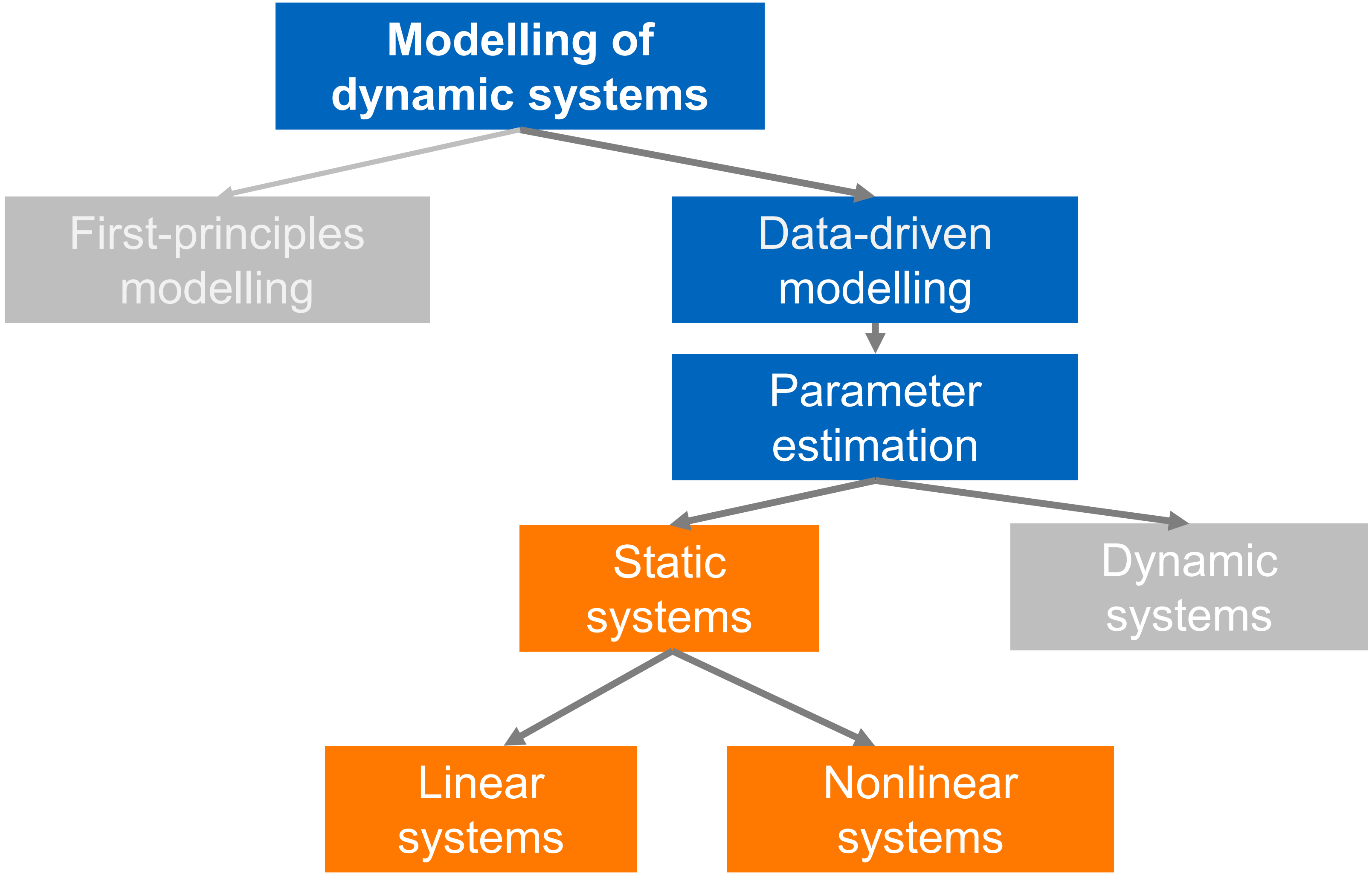
Aalto University
School of Electrical Engineering

ELEC-E8103 Modelling, Estimation and Dynamic Systems

Linear regression

Artur Kopitca,
Department of Electrical Engineering and Automation
Aalto University, School of Electrical Engineering
Email: artur.kopitca@aalto.fi

Overview



Modelling of dynamic systems
First-principles modelling
Data-driven modelling
Parameter estimation
Static systems
Dynamic systems
Linear systems
Nonlinear systems

Learning goals

Course Learning Outcomes

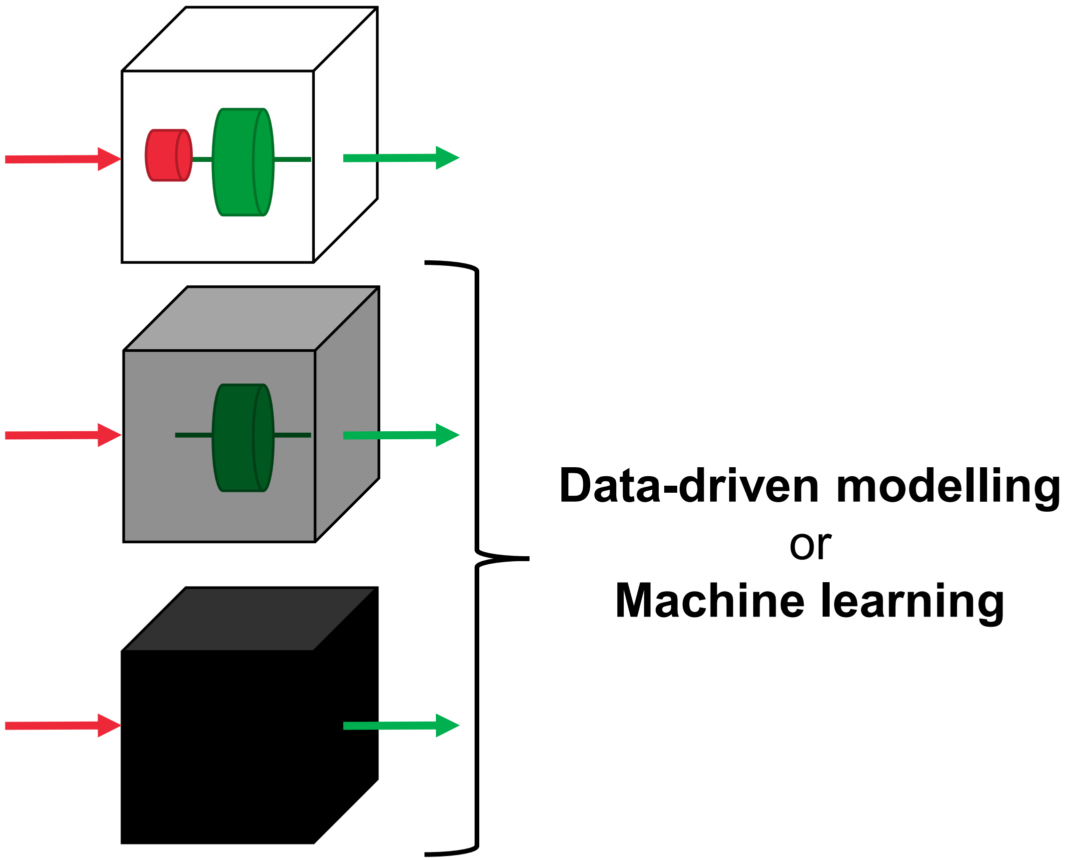
- Select a proper modelling approach for a specific practical problem,
- Formulate a mathematical model of a physical system,
- Construct models of systems using modelling tools, such as MATLAB and Simulink,
- Estimate the parameters of linear and nonlinear static systems from measurement data,
- Identify the models of linear dynamic systems from measurement data

Lecture Learning Outcomes

- Understand the principles of linear regression
- Apply the least-squares method in curve fitting
- Interpret the results of regression
- Deal with nonlinear relationships between inputs, outputs, and parameters
  - Curvilinear regression
  - Nonlinear regression

System models

- **White box**
  - Model based on theory
- **Grey box**
  - Model integrates partial theoretical structures of a system with empirical data to complete the model
- **Black box**
  - Model created completely based on data (input and output relationships)



Cantilever problem

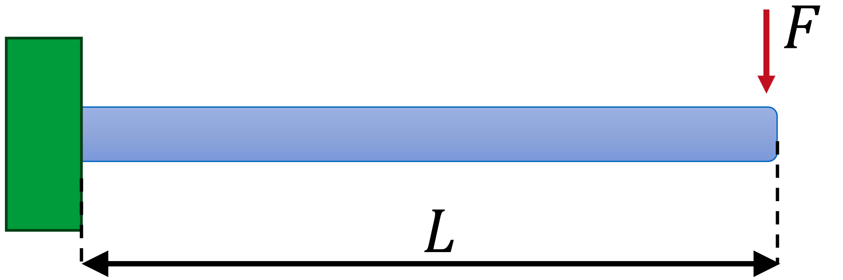
- For the cantilever problem we discussed on Lecture 1, we conducted experiments to estimate the Young's modulus (E)
- What we get:

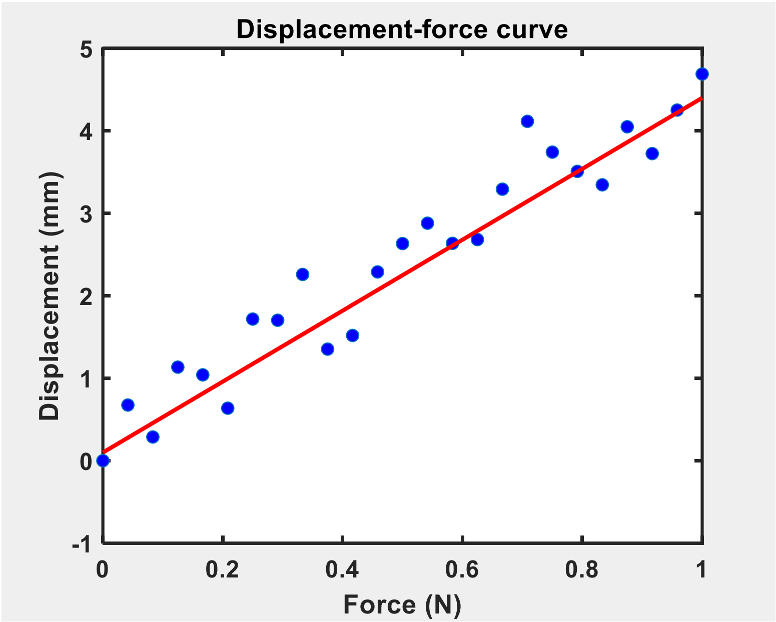
$$\begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}, \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{bmatrix}$$

- The goal is to find E such that

$$\delta = \frac{FL^3}{3EI} = \frac{L^3}{3EI}F = bF$$

- for all the measurement input/output pairs
  - where L, the length of the beam, and I, the momentum of inertia, are known





We don't know if the line passes through the origin, so we better write

$$\delta = a + bF$$

Problem formulation

- For the input/output pairs

$$X\left\{\begin{bmatrix}F_1\\F_2\\\vdots\\F_n\end{bmatrix}, \begin{bmatrix}\delta_1\\\delta_2\\\vdots\\\delta_n\end{bmatrix}\right\}Y$$

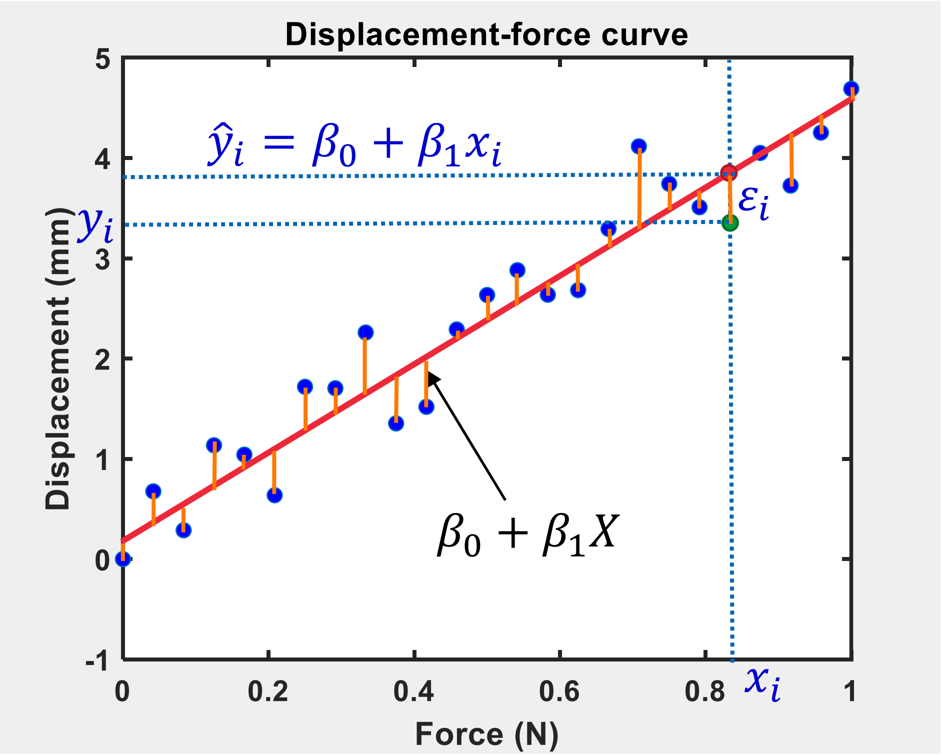
- Find a line

$$Y = \beta_0 + \beta_1 X \ ?$$

- where
  - β_0 is the intercept, or bias parameter
  - β_1 is the slope parameter
  - Both β_0 and β_1 are coefficients
  - X are input variables, or explanatory variables, here F
  - Y are measured variables, or dependent variables, here δ
- Including the error term, we have

$$Y = \beta_0 + \beta_1 X + \varepsilon = E(Y|X) + \varepsilon$$

  - ε is the error, $\varepsilon = [\varepsilon_1, \varepsilon_2, \dots \varepsilon_n]^T$
  - $E()$ is the expectation, or $\hat{Y}$



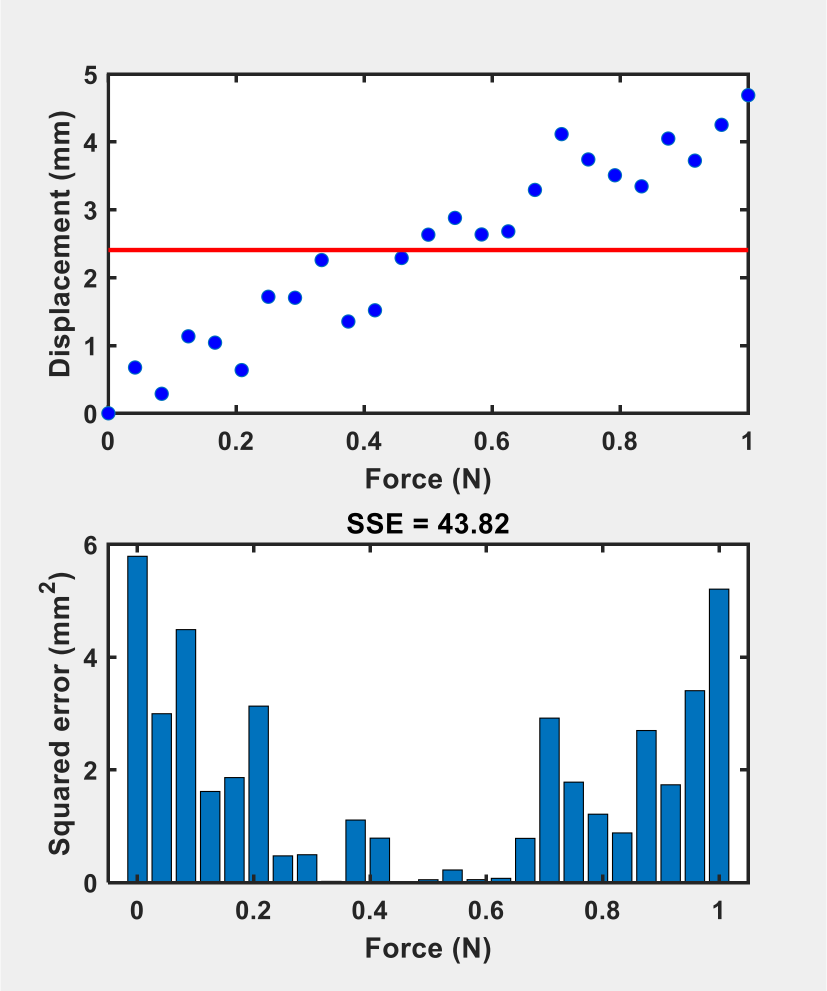
- How to model $\hat{Y}$, or how to determine β_0 and β_1?
  - We can use the residual or Sum of Squared Errors

$$SSE = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

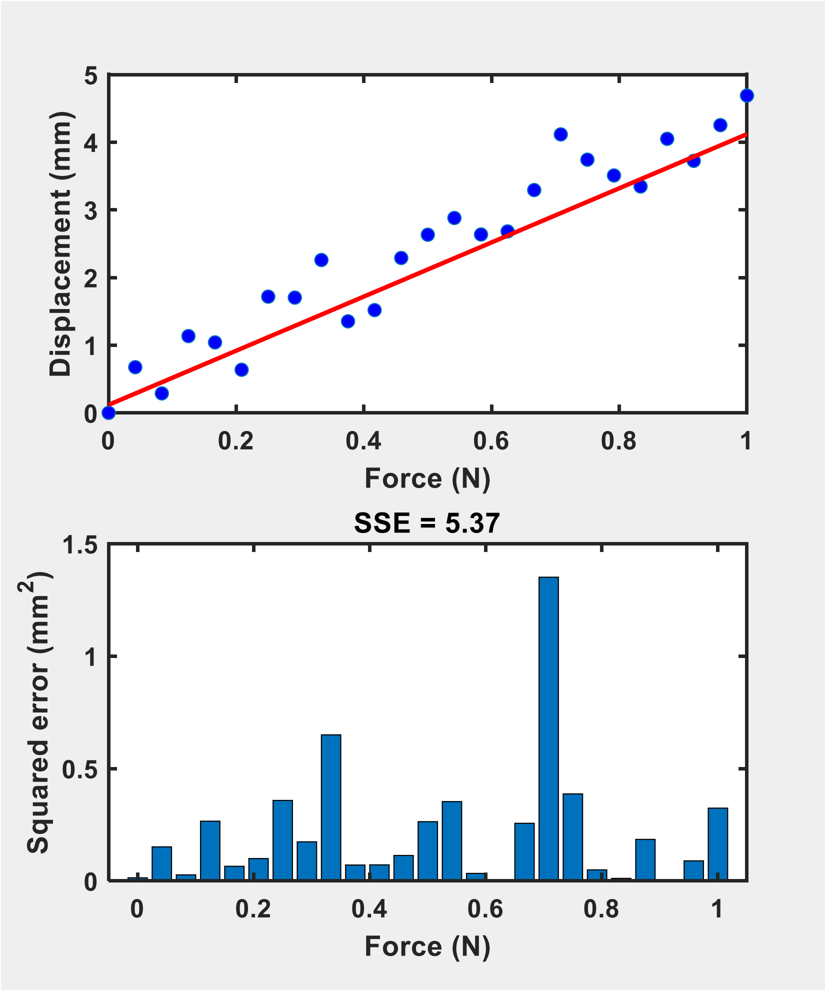
We want to find β_0 and β_1 such that SSE is minimum

Model examples

$$y = 2.41 + 0x$$



$$y = 0.12 + 4x$$



Aalto University
School of Electrical Engineering

The desired model

- For a problem,

$$Y = \beta_0 + \beta_1 X + \varepsilon = E(Y|X) + \varepsilon$$

- and the sum of squared residuals,

$$SSE = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

$$= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- The desired model is that SSE is minimized

Task: Find β_0, β_1 that minimize SSE

How to do that?

Find the minimum using calculus

$$SSE = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

$$SSE = \sum_{i=1}^{n} \left(y_i^2 - 2y_i(\beta_0 + \beta_1 x_i) + (\beta_0 + \beta_1 x_i)^2\right)$$

Reminder: Finding β_0, β_1

$$L_0 = \frac{\partial}{\partial \beta_0} SSE = \sum_{i=1}^{n} \left(2(\beta_0 + \beta_1 x_i) - 2y_i\right)$$

$$L_1 = \frac{\partial}{\partial \beta_1} SSE = \sum_{i=1}^{n} \left(2x_i(\beta_0 + \beta_1 x_i) - 2x_i y_i\right)$$

Let $L_0 = 0$ and $L_1 = 0$, we have

$$\sum_{i=1}^{n} y_i = n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} x_i y_i = \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2$$

So

$$\beta_0 = \frac{1}{n}\left(\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i\right)$$

$$\sum_{i=1}^{n} x_i y_i = \frac{1}{n}\left(\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i\right) \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2$$

$$n \sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i - \beta_1 \left(\sum_{i=1}^{n} x_i\right)^2 + n\beta_1 \sum_{i=1}^{n} x_i^2$$

$$\beta_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$\sum_{i=1}^{n} x_i = n\bar{x}$, $\sum_{i=1}^{n} \bar{x} = n\bar{x}$, $\widehat{\ }$ for estimation

So

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

or

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

We can calculate $\hat{\beta}_1$, $\hat{\beta}_0$ directly from data!

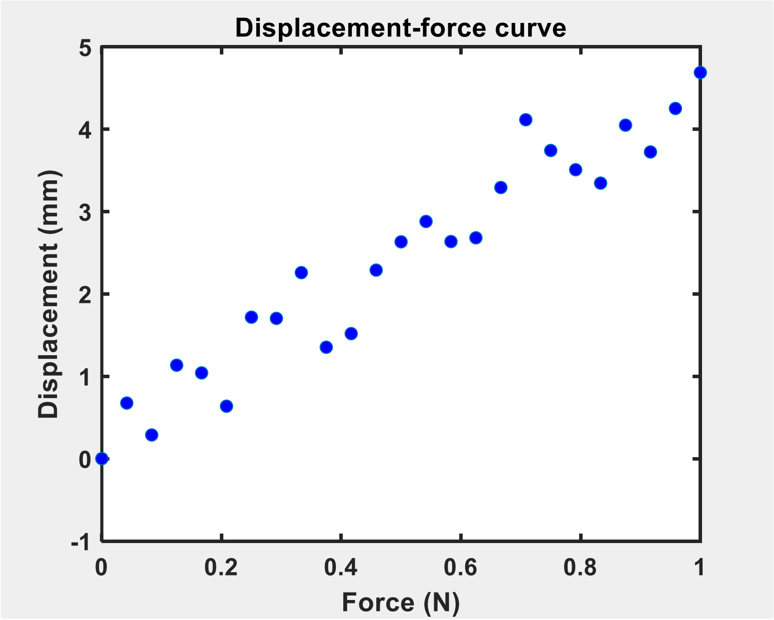
Code

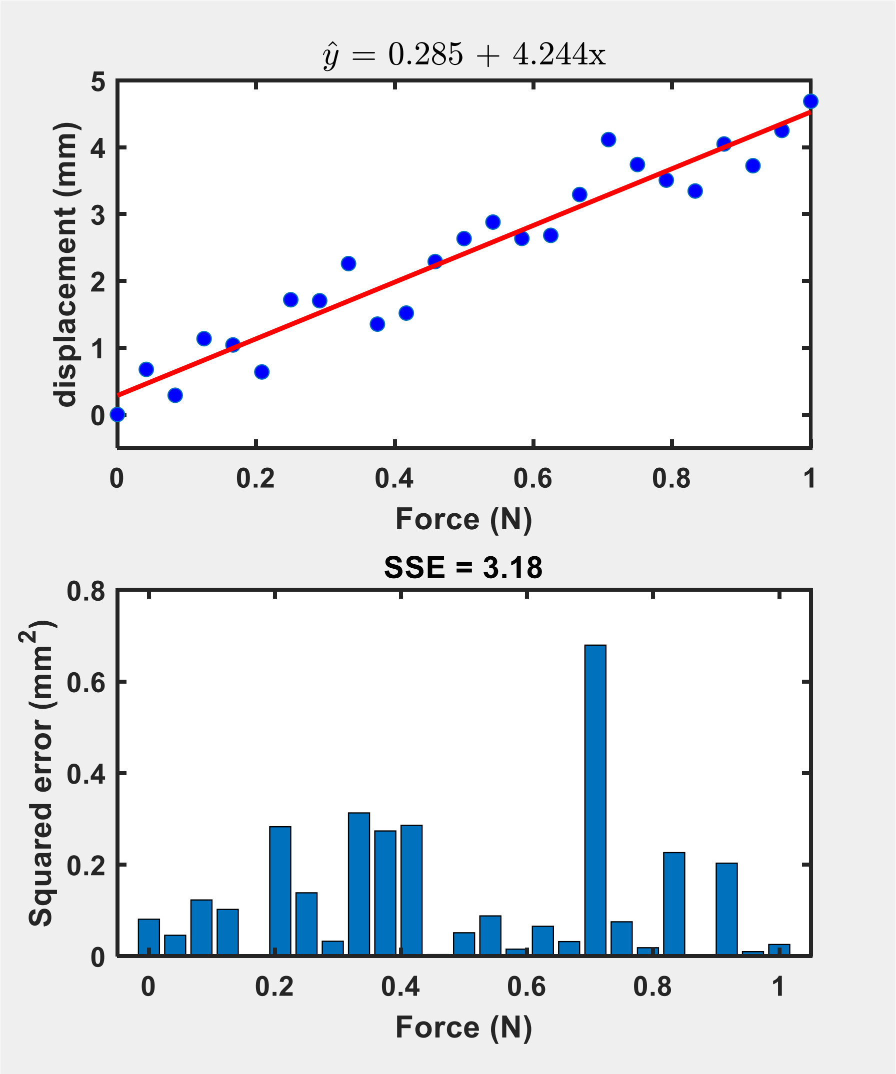
```
close all;

load lrdata
subplot(2,1,1); plot(x,y,'o','MarkerFaceColor', 'b');
xlabel('force (N)'); ylabel('displacement (mm)');
axis([0 1 -0.5 5]);
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
%% single variable linear least-square
n = length(x); p = 1;
xmean = mean(x); ymean = mean(y);
Sxy = sum(x.*y); Sxx = sum(x.^2);
beta1 = (Sxy-n*xmean*ymean)/(Sxx-n*xmean^2); disp(['beta1 = ',num2str(beta1)]);
beta0 = ymean-beta1*xmean; disp(['beta0 = ',num2str(beta0)]);

yhat = beta0 + beta1*x;
subplot(2,1,1); hold on, plot(x,yhat,'r', 'linewidth',2);
title(['$\hat{y}$ =' sprintf(' %1.3f + %1.3fx',beta0,beta1)],'Interpreter','latex','fontsize',14);
xlabel('Force (N)'); ylabel('displacement (mm)');
SE = (y-yhat).^2;
SSE = sum(SE);
MSE=SSE/(n-p-1);

subplot(2,1,2); bar(x,SE);
title(sprintf('SSE = %2.2f',SSE));
xlabel('Force (N)'); ylabel('Squared error (mm^2)');
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
set(gcf,'position',[300 300 500 600]);
```

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

$$\hat{\beta}_0 = \bar{y} - \beta_1 \bar{x}$$

Example





- $\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} = 4.244$
- $\hat{\beta}_0 = \bar{y} - \beta_1 \bar{x} = 0.285$

Better than our guessed line: $y = 0.12 + 4x$, where SSE = 5.37

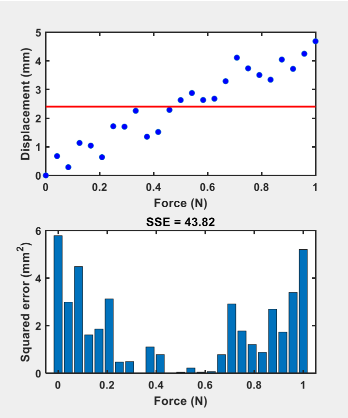
Goodness of fit

- SSE:

实际 预测

$$SSE(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

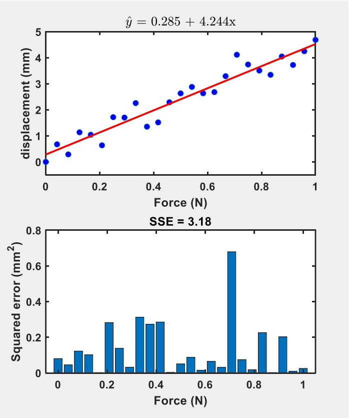
- SSE without regression, also called Total Sum of Squares



实际 均值

$$SST = \sum_{i=1}^{n} (y_i - \bar{Y})^2$$

$$= \left(\sum_{i=1}^{n} y_i^2\right) - n\bar{Y}^2$$



- The difference between SST and SSE is called Explained Sum of Squares or Regression Sum of Squares

$$SSR = SST - SSE = \sum_{i=1}^{n} (\hat{y}_i - \bar{Y})^2$$

预测-均值

- The coefficient of determination

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

- Shows how well data fits a statistical model
  - $R^2 = 1$: Perfect fit
  - $R^2 = 0$: No fit

Aalto University
School of Electrical Engineering

Calculating R^2

```
close all;

load lrdata
subplot(2,1,1); plot(x,y,'o','MarkerFaceColor', 'b');
xlabel('force (N)'); ylabel('displacement (mm)');
axis([0 1 -0.5 5]);
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
%% single variable linear least-square
n = length(x); p = 1;
xmean = mean(x); ymean = mean(y);
Sxy = sum(x.*y); Sxx = sum(x.^2);
beta1 = (Sxy-n*xmean*ymean)/(Sxx-n*xmean^2); disp(['beta1 = ',num2str(beta1)]);
beta0 = ymean-beta1*xmean; disp(['beta0 = ',num2str(beta0)]);

yhat = beta0 + beta1*x;
subplot(2,1,1); hold on, plot(x,yhat,'r', 'linewidth',2);
title(['$\hat{y}$ =' sprintf(' %1.3f + %1.3fx',beta0,beta1)],'Interpreter','latex','fontsize',14);
xlabel('Force (N)'); ylabel('displacement (mm)');
SE = (y-yhat).^2;
SSE = sum(SE);
MSE=SSE/(n-p-1);

subplot(2,1,2); bar(x,SE);
title(sprintf('SSE = %2.2f',SSE));
xlabel('Force (N)'); ylabel('Squared error (mm^2)');
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
set(gcf,'position',[300 300 500 600]);
%% R2
SST = sum((y-ymean).^2);
R2 = 1-SSE/SST;
disp(['R2 = ',num2str(R2)]);
```

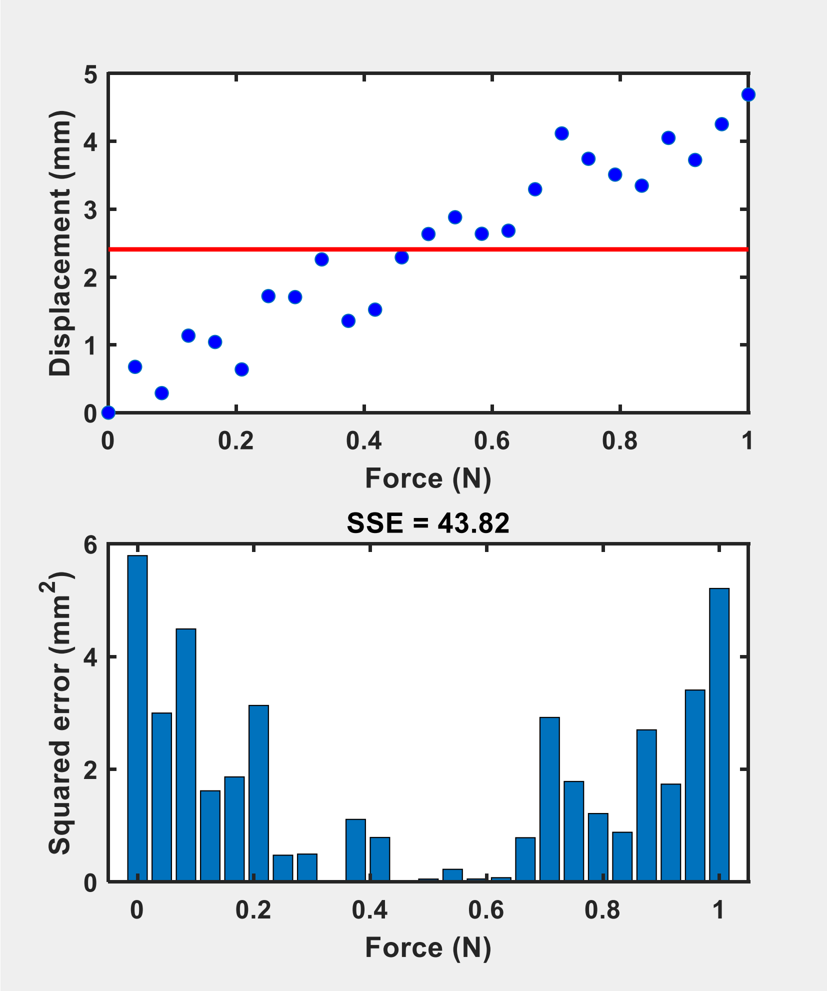
$$SST = \sum_{i=1}^{n} (y_i - \bar{Y})^2$$

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

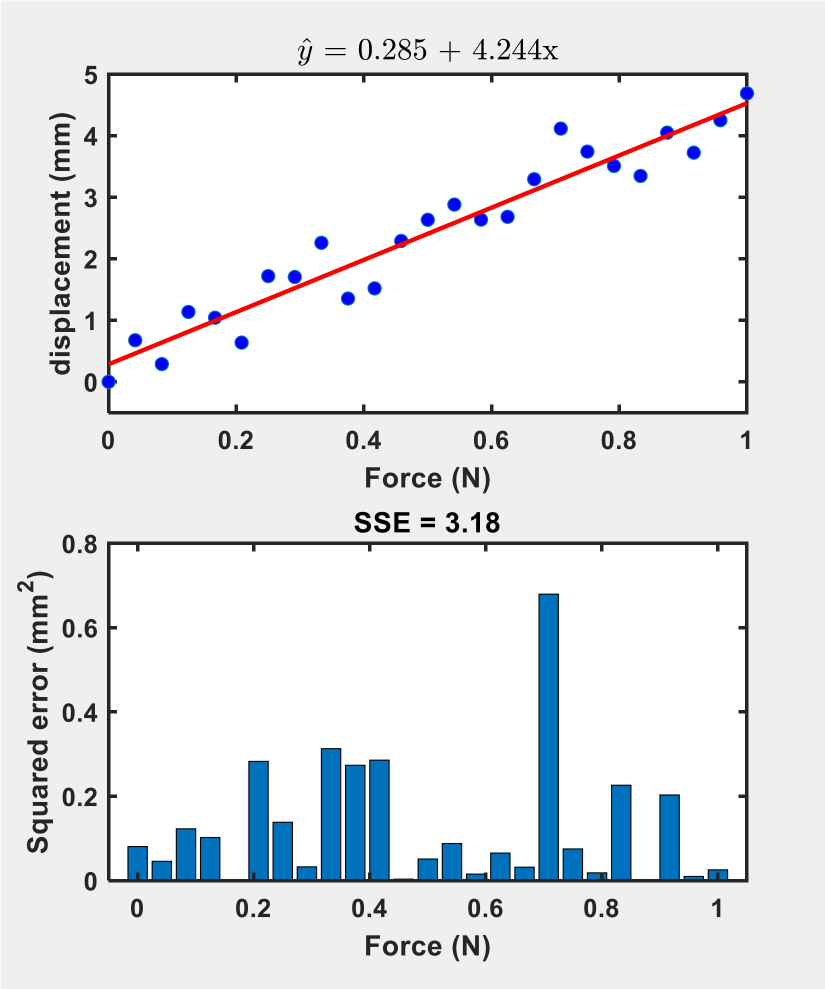
Aalto University
School of Electrical
Engineering

$R^2 = 0$

$R^2 = 0.9275$

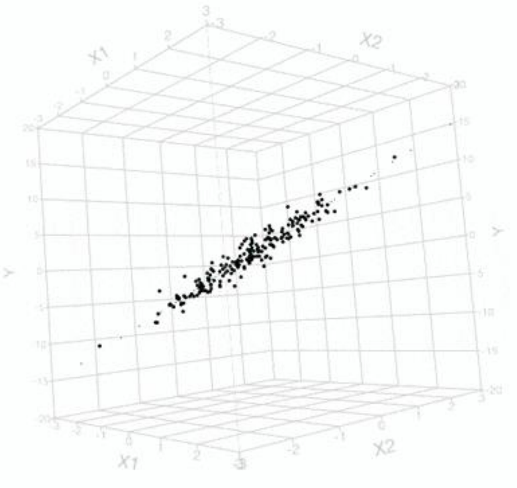


Displacement (mm)
Force (N)
SSE = 43.82
Squared error (mm^2)
Force (N)



$\hat{y} = 0.285 + 4.244x$
displacement (mm)
Force (N)
SSE = 3.18
Squared error (mm^2)
Force (N)

Multiple-input cases



For a problem with p inputs and n data points

$$\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$$

The relationship is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

$$\varepsilon_i = y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)$$

SSE can be represented as:

$$SSE = \sum_{i=1}^{n} \varepsilon_i^2$$

Let X be a $n \times (p+1)$ matrix

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

So, we have

$$Y - X\beta = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \varepsilon$$

We can write:

$$SSE = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon^T \varepsilon = (Y - X\beta)^T (Y - X\beta)$$

$$= Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta$$

and

$$\frac{\partial SSE}{\partial \beta} = -2X^T (Y - X\beta)$$

If columns of X are linearly independent, let

$$X^T (Y - X\beta) = 0$$

We have the normal equation:

$$(X^T X)\beta = X^T Y$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

So

$$\hat{Y} = X\beta = \overbrace{X(X^T X)^{-1} X^T}^{\text{Hat matrix: } H} Y$$

What is the potential challenge?

Aalto University
School of Electrical Engineering

Solving linear regression by search

- We can treat the residual as a cost function of an optimization problem

$$J(\beta) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j x_j\right)\right)^2}_{\text{MSE}}$$

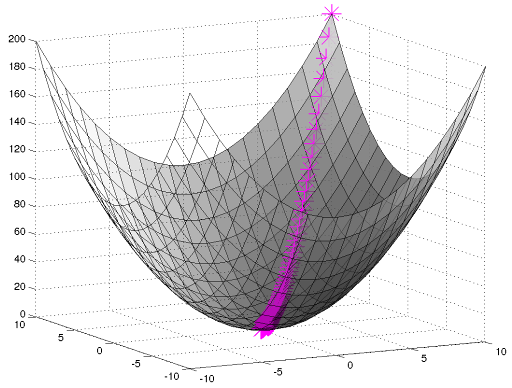
- And solve

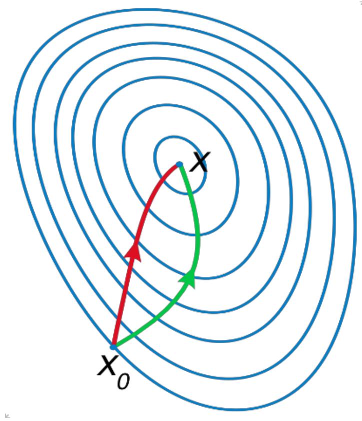
$$\arg\min_{\beta} J(\beta)$$

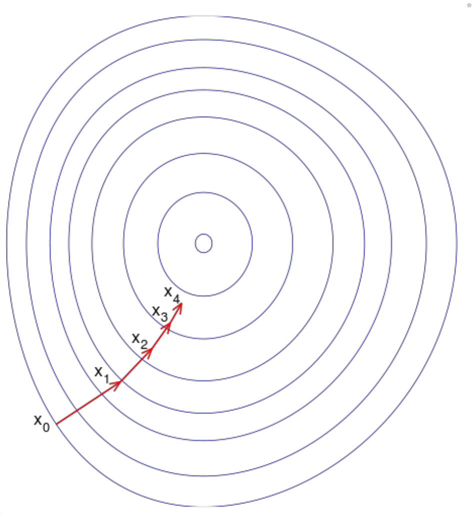
- Using search algorithm such as gradient descent or Newton's method

```
While not converged
```

$$\beta(k+1) = \beta(k) - \gamma \nabla J\big(\beta(k)\big)$$







Search is cheaper when the problem is large

Aalto University
School of Electrical Engineering

```matlab
close all;

load lrdata
subplot(2,1,1); plot(x,y,'o','MarkerFaceColor', 'b');
xlabel('force (N)'); ylabel('displacement (mm)');
axis([0 1 -0.5 5]);
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
%% single variable linear least-square
n = length(x); p = 1;
xmean = mean(x); ymean = mean(y);
Sxy = sum(x.*y); Sxx = sum(x.^2);
beta1 = (Sxy-n*xmean*ymean)/(Sxx-n*xmean^2); disp(['beta1 = ',num2str(beta1)]);
beta0 = ymean-beta1*xmean; disp(['beta0 = ',num2str(beta0)]);

yhat = beta0 + beta1*x;
subplot(2,1,1); hold on, plot(x,yhat,'r', 'linewidth',2);
title(['$\hat{y}$ =' sprintf(' %1.3f + %1.3fx',beta0,beta1)],'Interpreter','latex','fontsize',14);
xlabel('Force (N)'); ylabel('displacement (mm)');
SE = (y-yhat).^2;
SSE = sum(SE);
MSE=SSE/(n-p-1);

subplot(2,1,2); bar(x,SE);
title(sprintf('SSE = %2.2f',SSE));
xlabel('Force (N)'); ylabel('Squared error (mm^2)');
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
set(gcf,'position',[300 300 500 600]);
%% R2
SST = sum((y-ymean).^2);
R2 = 1-SSE/SST;
disp(['R2 = ',num2str(R2)]);

%% using general formulation of linear least square
X = [ones(length(x),1) x'];
Y = y';
beta = inv(X'*X)*X'*Y;
disp(['beta = ' num2str(beta(1)) '; ' num2str(beta(2))]);
```

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Modelling the error

$$\hat{Y} = X\beta$$
$$\hat{\beta} = X(X^TX)^{-1}X^TY$$

- Is SSE a good measure of modelling error?
- The variance, or the mean squared error, of the regression is usually calculated by
$$\hat{\sigma}^2 = MSE = \frac{SSE}{n-p-1}$$
$$= \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
- $n-p-1$ instead of n is due to the number of regression parameters
- The standard deviation of the error:
$$\hat{\sigma} = \sqrt{MSE} = \sqrt{\frac{SSE}{n-p-1}}$$

- From the normal equation, we have the covariance matrix of $\hat{\beta}$:
$$Var(\hat{\beta}) = (X^TX)^{-1}\hat{\sigma}^2$$
- The variance of jth coefficient
$$Var(\hat{\beta}_j) = (X^TX)^{-1}_{jj}\hat{\sigma}^2$$
- The standard error of coefficients can be estimated with:
$$s.e.(\hat{\beta}_j) = \sqrt{(X^TX)^{-1}_{jj}\hat{\sigma}^2}$$
- For simple regression, the *s.e.* of coefficients are:
$$s.e.(\hat{\beta}_0) = \hat{\sigma}\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}}$$
$$s.e.(\hat{\beta}_1) = \frac{\hat{\sigma}}{\sqrt{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}}$$

Confidence interval

- How confident can we be in our model when it is applied to new, unseen data:
  - For example, what is the range of values that we are 95% confident $\hat{\beta}_j$ will fall into based on our observed data?
  - The $100(1-\alpha)\%$ confidence interval of coefficient $\hat{\beta}_j$ is:

$$\hat{\beta}_j \pm t_{(n-1-p,\alpha/2)} s.e.(\hat{\beta}_j)$$

where $t_{(n-1-p,\alpha/2)}$ is a t-distribution with degree of freedom of $n-1-p$

$n-p-1$ $\alpha/2$

Table entry for p and C is the critical t* value with probability p lying to its right and probability C lying between –t* and t*.

Upper Tail Probability p → / Degrees of freedom ↓	0.25	0.20	0.15	0.10	0.05	0.025	0.02	0.01	0.005	0.0025	0.001	0.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	0.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	0.765	0.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	0.741	0.941	1.190	$\boxed{1.533}$	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	0.727	0.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.894	6.869
6	0.718	0.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	0.711	0.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	0.706	0.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	0.703	0.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	0.700	0.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	0.697	0.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	0.695	0.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318
13	0.694	0.870	1.079	1.350	1.771	2.160	2.282	2.650	3.012	3.372	3.852	4.221
14	0.692	0.868	1.076	1.345	1.761	2.145	2.264	2.624	2.977	3.326	3.787	4.140
15	0.691	0.866	1.074	1.341	1.753	2.131	2.249	2.602	2.947	3.286	3.733	4.073
16	0.690	0.865	1.071	1.337	1.746	2.120	2.235	2.583	2.921	3.252	3.686	4.015
17	0.689	0.863	1.069	1.333	1.740	2.110	2.224	2.567	2.898	3.222	3.646	3.965
18	0.688	0.862	1.067	1.330	1.734	2.101	2.214	2.552	2.878	3.197	3.610	3.922
19	0.688	0.861	1.066	1.328	1.729	2.093	2.205	2.539	2.861	3.174	3.579	3.883
20	0.687	0.860	1.064	1.325	1.725	2.086	2.197	2.528	2.845	3.153	3.552	3.850
21	0.686	0.859	1.063	1.323	1.721	2.080	2.189	2.518	2.831	3.135	3.527	3.819
22	0.686	0.858	1.061	1.321	1.717	2.074	2.183	2.508	2.819	3.119	3.505	3.792
23	0.685	0.858	1.060	1.319	1.714	2.069	2.177	2.500	2.807	3.104	3.485	3.768
24	0.685	0.857	1.059	1.318	1.711	2.064	2.172	2.492	2.797	3.091	3.467	3.745
25	0.684	0.856	1.058	1.316	1.708	2.060	2.167	2.485	2.787	3.078	3.450	3.725
26	0.684	0.856	1.058	1.315	1.706	2.056	2.162	2.479	2.779	3.067	3.435	3.707
27	0.684	0.855	1.057	1.314	1.703	2.052	2.158	2.473	2.771	3.057	3.421	3.689
28	0.683	0.855	1.056	1.313	1.701	2.048	2.154	2.467	2.763	3.047	3.408	3.674
29	0.683	0.854	1.055	1.311	1.699	2.045	2.150	2.462	2.756	3.038	3.396	3.660
30	0.683	0.854	1.055	1.310	1.697	2.042	2.147	2.457	2.750	3.030	3.385	3.646
40	0.681	0.851	1.050	1.303	1.684	2.021	2.123	2.423	2.704	2.971	3.307	3.551
50	0.679	0.849	1.047	1.299	1.676	2.009	2.109	2.403	2.678	2.937	3.261	3.496
60	0.679	0.848	1.045	1.296	1.671	2.000	2.099	2.390	2.660	2.915	3.232	3.460
80	0.678	0.846	1.043	1.292	1.664	1.990	2.088	2.374	2.639	2.887	3.195	3.416
100	0.677	0.845	1.042	1.290	1.660	1.984	2.081	2.364	2.626	2.871	3.174	3.390
1000	0.675	0.842	1.037	1.282	1.646	1.962	2.056	2.330	2.581	2.813	3.098	3.300
z*	0.674	0.841	1.036	1.282	1.645	1.960	2.054	2.326	2.576	2.807	3.091	3.291
Confidence level C = 1- 2p →	50%	60%	70%	80%	90%	95%	96%	98%	99%	99.5%	99.8%	99.9%

80% confidence

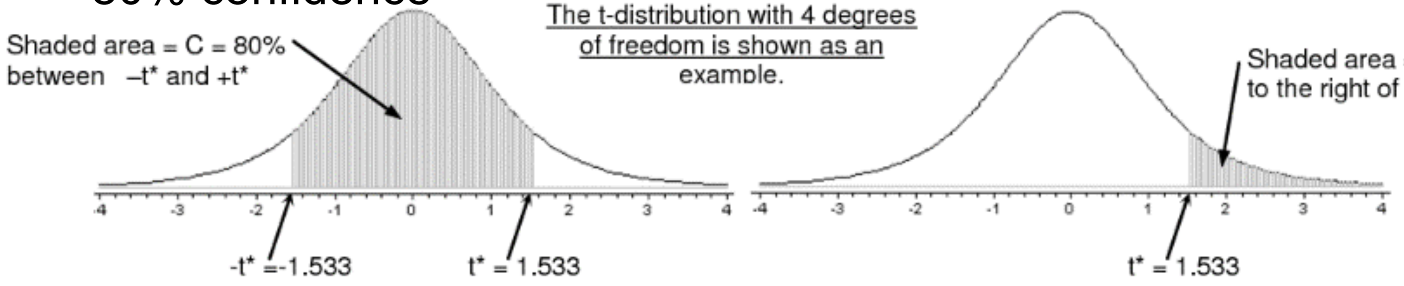


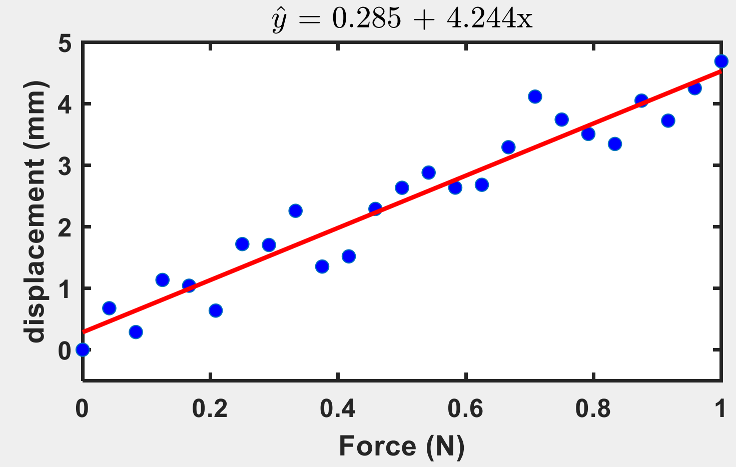
Table entry for p and C is the critical t* value with probability p lying to its right and probability C lying between –t* and t*.

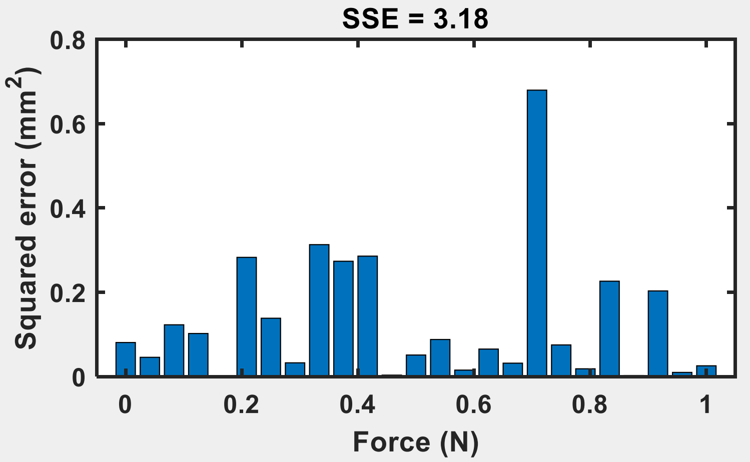
Example

- For our model:
  - $n = 25$
  - $p = 1$
  - Degree of freedom $n - p - 1 = 23$
  - If we want to check 95% confidence interval, $\alpha = 0.05$
  - $t_{(n-1-p,\alpha/2)} = 2.069$
- Using the least-squares formulation, we get
  - $\hat{\beta}_0 = 0.285$
  - $\hat{\beta}_1 = 4.244$
  - $s.e.(\hat{\beta}_0) = 0.1443$
  - $s.e.(\hat{\beta}_1) = 0.2474$
- Confidence interval at 95% is
  - For $\hat{\beta}_0$: [0.047865   0.52292]
  - For $\hat{\beta}_1$: [3.8365   4.6509]
- $R^2 = 0.9275$, quite good

Table entry for p and C is the critical t* value with probability p lying to its right and probability C lying between –t* and t*.

Upper Tail Probability p →	0.25	0.20	0.15	0.10	0.05	0.025	0.02	0.01	0.005	0.0025	0.001	0.0005
Degrees of freedom ↓												
21	0.686	0.859	1.063	1.323	1.721	2.080	2.189	2.518	2.831	3.135	3.527	3.819
22	0.686	0.858	1.061	1.321	1.717	2.074	2.183	2.508	2.819	3.119	3.505	3.792
23	0.685	0.858	1.060	1.319	1.714	2.069	2.177	2.500	2.807	3.104	3.485	3.768
24	0.685	0.857	1.059	1.318	1.711	2.064	2.172	2.492	2.797	3.091	3.467	3.745
25	0.684	0.856	1.058	1.316	1.708	2.060	2.167	2.485	2.787	3.078	3.450	3.725
Confidence level C = 1- 2p →	50%	60%	70%	80%	90%	95%	96%	98%	99%	99.5%	99.8%	99.9%





Prediction interval

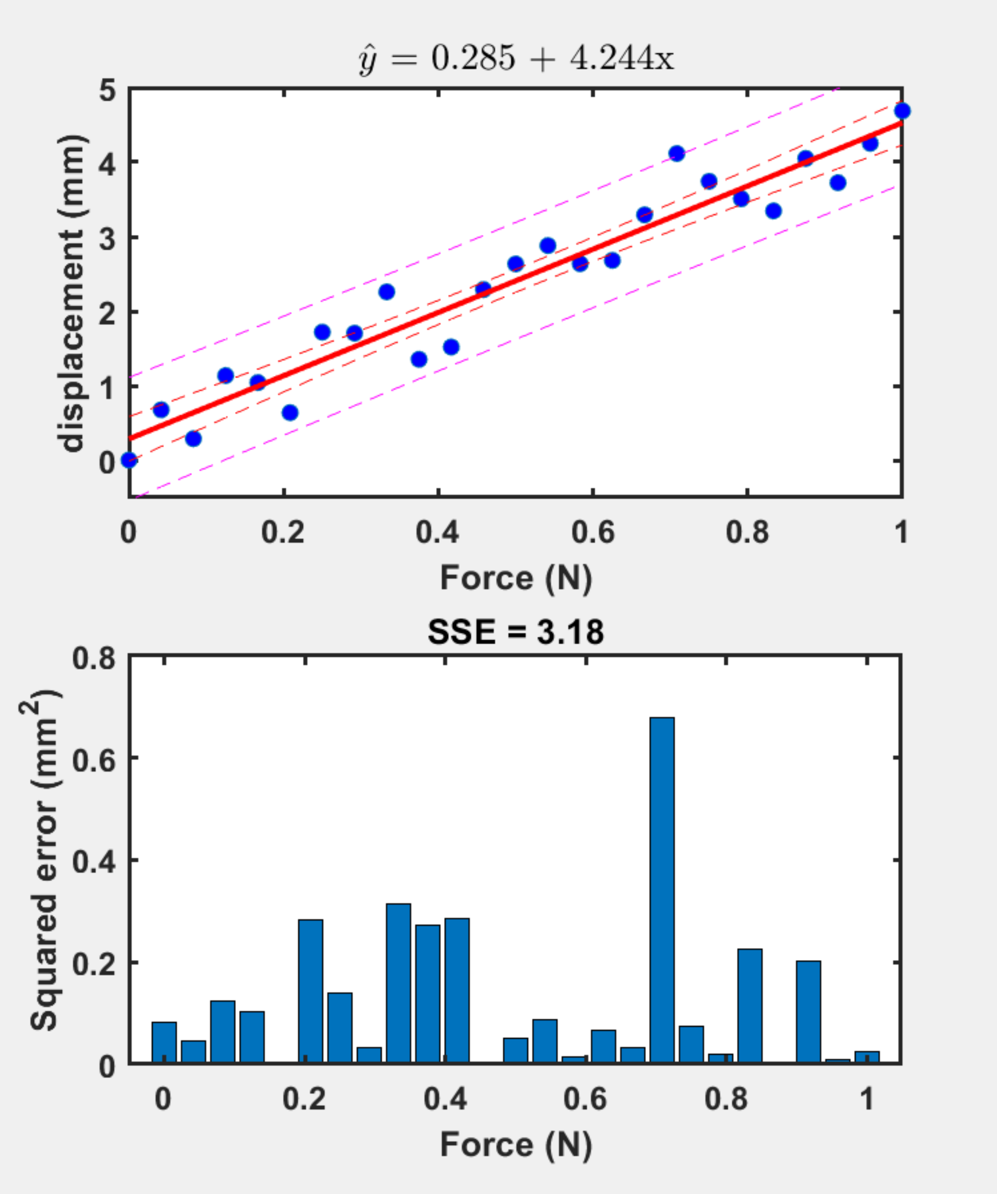
- Where the future measurement will be:
  - Or: What is the range of values that we are 95% confident the future measurement will fall into based on our observed data?
  - The $100(1-\alpha)\%$ prediction interval of a new predicted output $\hat{y}_{n+1}$ with input x_{n+1} is:

$$\hat{y}_{new} \pm t_{(n-1-p,\alpha/2)} s.e.(\hat{y}_{new})$$

  - where $t_{(n-1-p,\alpha/2)}$ is a t-distribution with degree of freedom of $n-1-p$

$$s.e.(\hat{y}_{new}) = \sqrt{MSE\left(1+\frac{1}{n}+\frac{(x_{new}-\bar{x})^2}{\sum(x_i-\bar{x})^2}\right)}$$

$$= \sqrt{MSE\left(1+\frac{1}{n}+\frac{(x_{new}-\bar{x})^2}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}\right)}$$



```matlab
close all;

load lrdata
subplot(2,1,1); plot(x,y,'o','MarkerFaceColor', 'b');
xlabel('force (N)'); ylabel('displacement (mm)');
axis([0 1 -0.5 5]);
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
%% single variable linear least-square
n = length(x); p = 1;
xmean = mean(x); ymean = mean(y);
Sxy = sum(x.*y); Sxx = sum(x.^2);
beta1 = (Sxy-n*xmean*ymean)/(Sxx-n*xmean^2); disp(['beta1 = ',num2str(beta1)]);
beta0 = ymean-beta1*xmean; disp(['beta0 = ',num2str(beta0)]);

yhat = beta0 + beta1*x;
subplot(2,1,1); hold on, plot(x,yhat,'r', 'linewidth',2);
title(['$\hat{y}$ =' sprintf(' %1.3f + %1.3fx',beta0,beta1)],'Interpreter','latex','fontsize',14);
xlabel('Force (N)'); ylabel('displacement (mm)');
SE = (y-yhat).^2;
SSE = sum(SE);
MSE=SSE/(n-p-1);

subplot(2,1,2); bar(x,SE);
title(sprintf('SSE = %2.2f',SSE));
xlabel('Force (N)'); ylabel('Squared error (mm^2)');
set(gca,'linewidth',1.5,'fontweight','bold','fontsize',12)
set(gcf,'position',[300 300 500 600]);
%% R2
SST = sum((y-ymean).^2);
R2 = 1-SSE/SST;
disp(['R2 = ',num2str(R2)]);

%% using general formulation of linear least square
X = [ones(length(x),1) x'];
Y = y';
beta = inv(X'*X)*X'*Y;
disp(['beta = ' num2str(beta(1)) '; ' num2str(beta(2))]);

%% variance and standard error of the coefficients
s = sqrt(MSE);
var = inv(X'*X)*s^2
seB = sqrt(diag(var))
seB0 = s*sqrt(1/n+xmean^2/(Sxx-n*xmean^2))
seB1 = s/sqrt(Sxx-n*xmean^2)

%% confidnece interval of regress parameters at 95%
intervalbeta0 = [beta0-2.069*seB0 beta0+2.069*seB0];
intervalbeta1 = [beta1-2.069*seB1 beta1+2.069*seB1];

disp(['Confidence interval of ' num2str(char(946)) '0 is [', num2str(intervalbeta0) ']']);
disp(['Confidence interval of ' num2str(char(946)) '1 is [', num2str(intervalbeta1) ']']);

%% prediction interval at 95%
sePredict = @(v) sqrt(MSE*(1+1/n+(v-xmean).^2/(Sxx-n*xmean^2)));
subplot(2,1,1), plot(x,yhat+2.069*sePredict(x),'m--',x,yhat-2.069*sePredict(x),'m--');

%% confidence interval at 95%
seConfident = @(v) sqrt(MSE*(1/n+(v-xmean).^2/(Sxx-n*xmean^2)));
plot(x,yhat+2.069*seConfident(x),'r--',x,yhat-2.069*seConfident(x),'r--');
```

$$\hat{\sigma} = \sqrt{MSE} = \sqrt{\frac{SSE}{n-p-1}}$$

$$Var(\hat{\beta}) = (X^T X)^{-1}\hat{\sigma}^2$$

$$s.e.(\hat{\beta}_j) = \sqrt{(X^T X)^{-1}_{jj}\hat{\sigma}^2}$$

$$s.e.(\hat{\beta}_0) = \hat{\sigma}\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n x_i^2 - n\bar{x}^2}}$$

$$s.e.(\hat{\beta}_1) = \frac{\hat{\sigma}}{\sqrt{\sum_{i=1}^n x_i^2 - n\bar{x}^2}}$$

$$\hat{\beta}_j \pm t_{(n-1-p,\alpha/2)} s.e.(\hat{\beta}_j)$$

$$s.e.(\hat{y}_{new}) = \sqrt{MSE\left(1 + \frac{1}{n} + \frac{(x_{new}-\bar{x})^2}{\sum_{i=1}^n x_i^2 - n\bar{x}^2}\right)}$$

$$\hat{y}_{new} \pm t_{\left(n-1-p,\frac{\alpha}{2}\right)} s.e.(\hat{y}_{new})$$

Summary of Linear Least Squares Model

- The linear regression model predicting the real-valued output Y has the form:

$$\hat{Y}(X) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

- The least-squares solution that minimizes the residual

$$SSE(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- is

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

- where

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$$

- Coefficient of determination

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n} (\hat{y}_i - \bar{Y})^2}{\sum_{i=1}^{n} y_i^2 - n\bar{Y}^2}$$

- Variance of coefficient

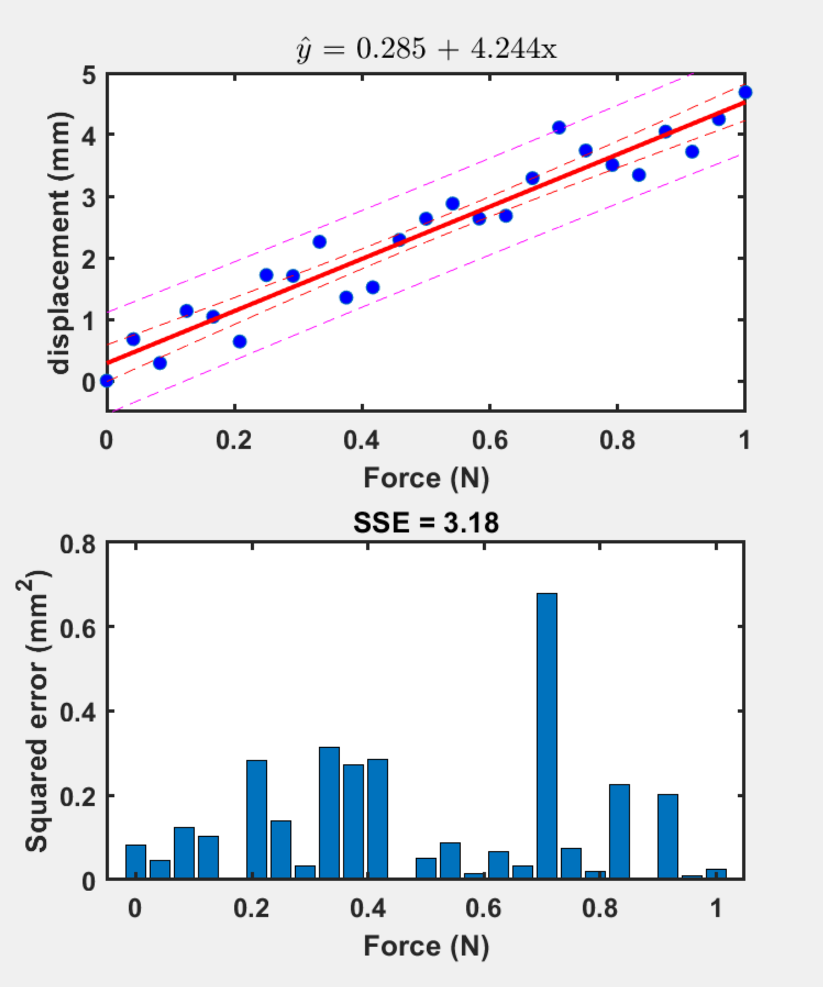
$$Var(\hat{\beta}_j) = (X^T X)_{jj}^{-1} \hat{\sigma}^2$$

- Confidence interval of the coefficient

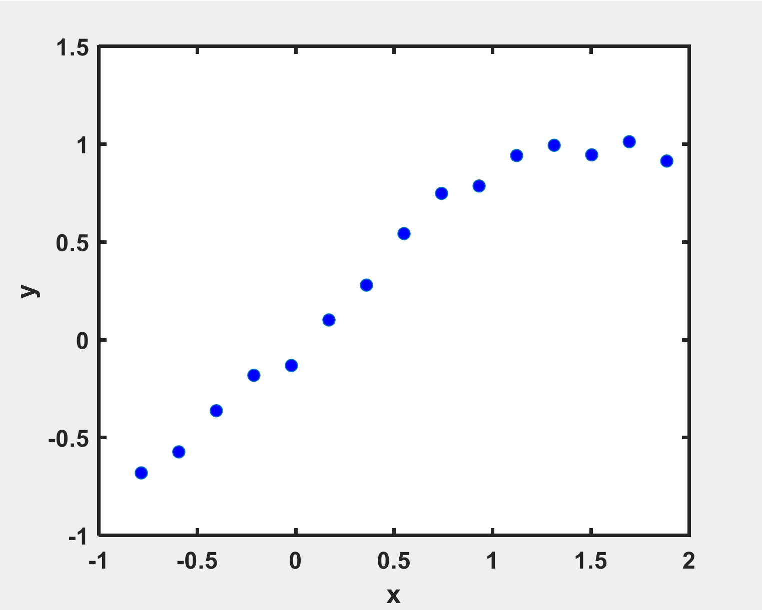
$$\hat{\beta}_j \pm t_{(n-1-p, \alpha/2)} s.e.(\hat{\beta}_j)$$

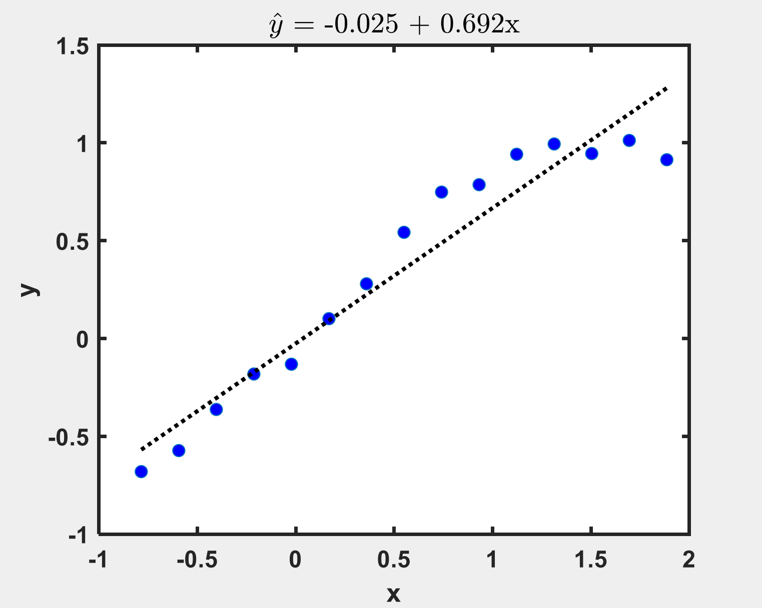
- Predication interval

$$\hat{y}_{new} \pm t_{(n-1-p, \alpha/2)} s.e.(\hat{y}_{new})$$

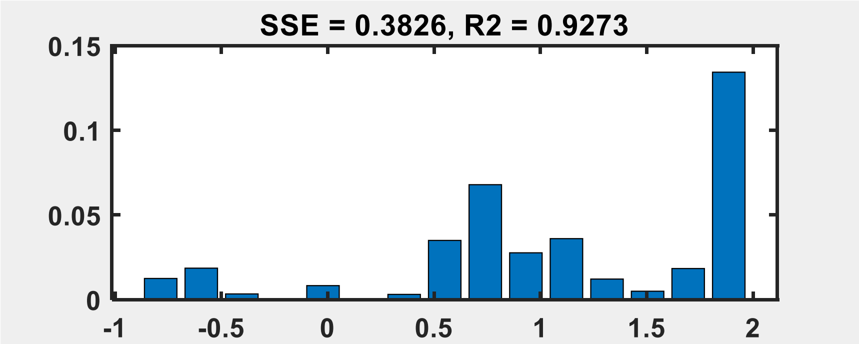


Not so linear problems





- We can approximate it directly using linear least square regression
- Results:
  - $y = 0.025 + 0.692x$
  - $R^2 = 0.9273$
- This could be acceptable



Curvilinear regression

- We could also use the curvilinear approach
  - where we transform a non-linear problem to a linear problem
- The plot looks like a sinusoidal function, so we can use

$$y = \beta_0 + \beta_1 \sin(x)$$

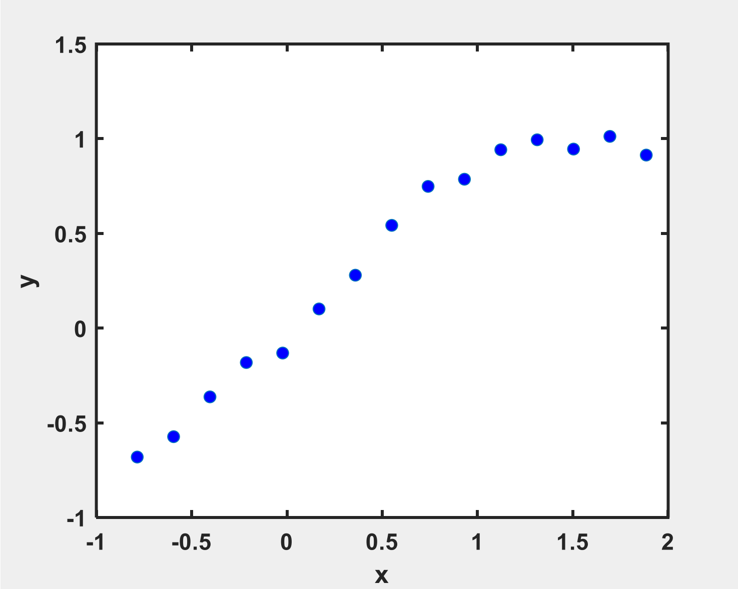
- So

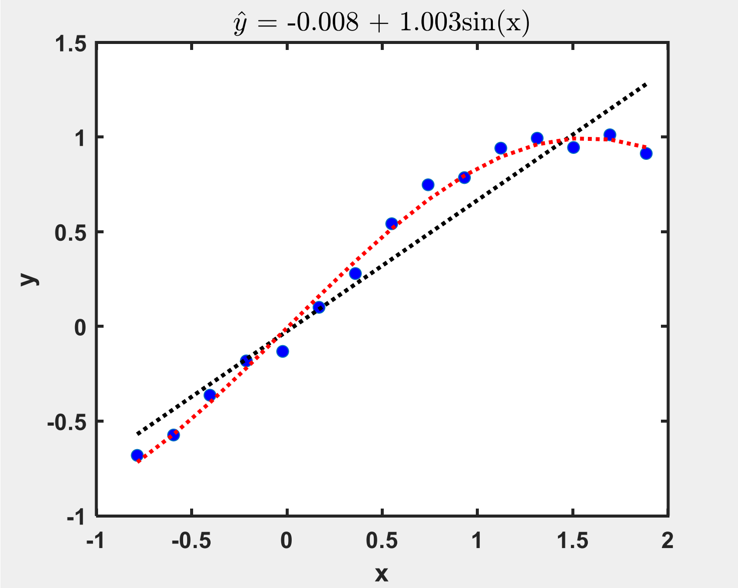
$$X = \begin{bmatrix} 1 & \sin(x_1) \\ \vdots & \vdots \\ 1 & \sin(x_n) \end{bmatrix}$$

- So, we have:

$$y = -0.008 + 1.003\sin(x)$$
$$R^2 = 0.993$$

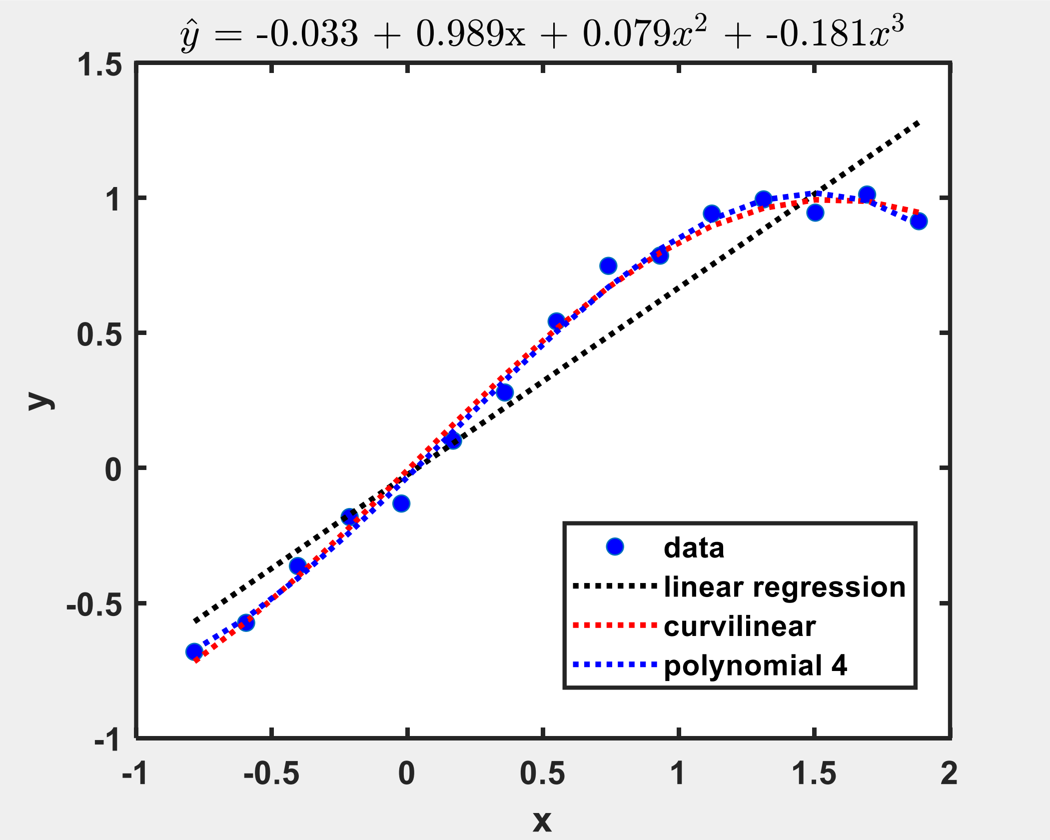
(vs. 0.927 for a line)





Polynomial regression

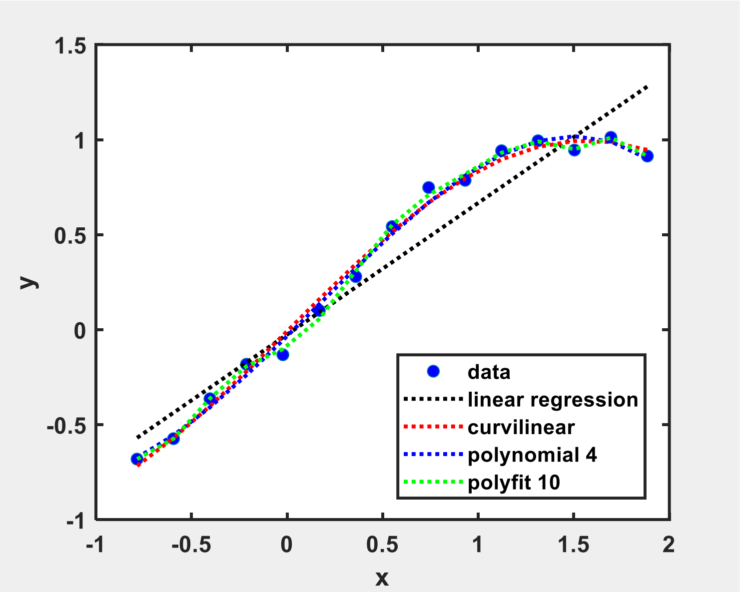
- For the previous example, if we do not know it is sinusoidal, we can use a more generic polynomial:
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$
- So, we have:
$$y = -0.033 + 0.989x + 0.079x^2 - 0.181x^3$$
- $R^2 = 0.994$, not bad!

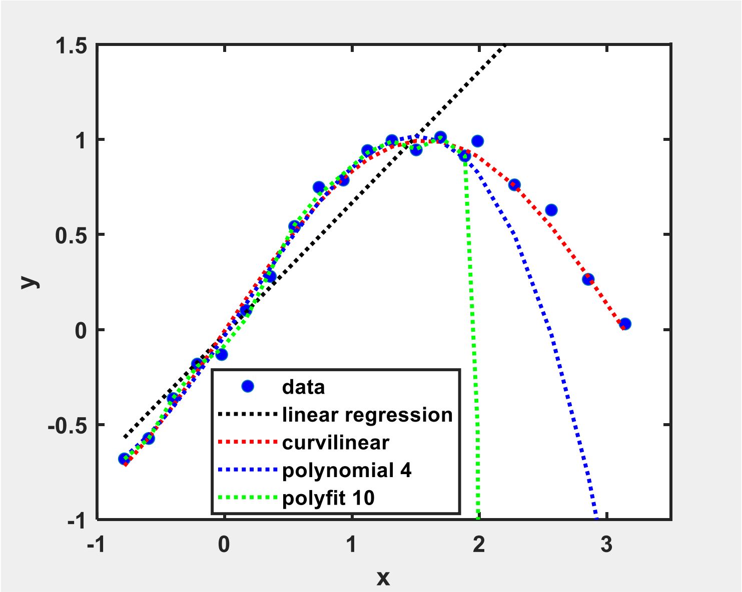


More on least-squares polynomial regression

最小二乘解

- MATLAB [mldivide] operator, \, gives the least-squares solution ($Ax = B \rightarrow x = A\backslash B$)
- [Polyfit] does the least-squares polynomial fitting
  - Polyfit of 10th order: R^2 = 0.9987!
  - Be careful of the limitation of polynomial fitting
  - Overfit does more harm than good
- Extrapolation is not guaranteed



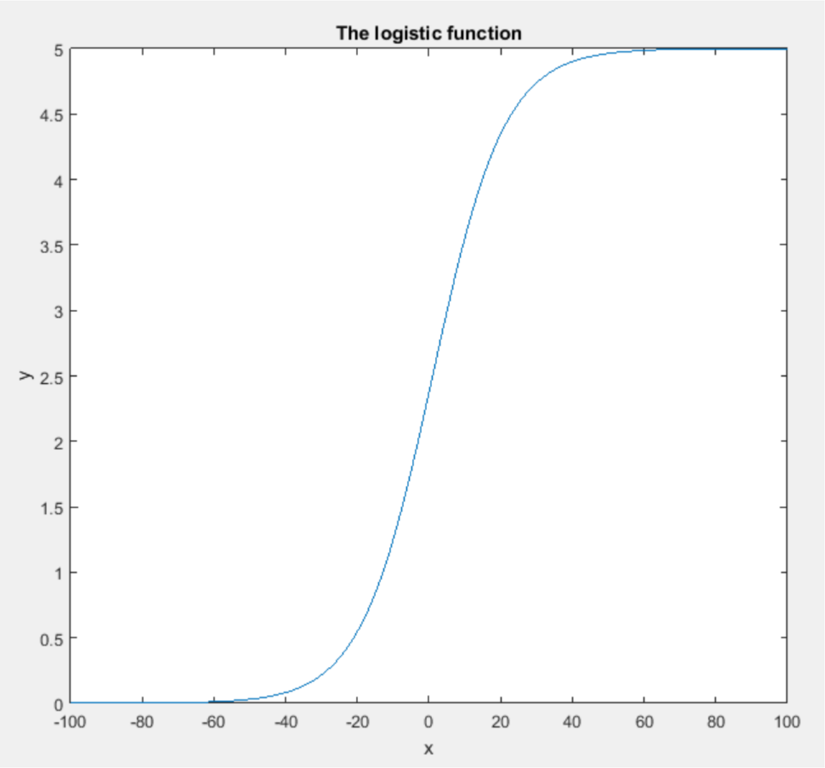


Other nonlinear problems

- Some problems are hard to linearize, or we prefer not to linearize, e.g., logistic function or logistic curve:

$$y = \frac{L}{1 + e^{-k(x - x_0)}}$$

- Where
  - x_0 the x-value of the Sigmoid's mid-point
  - L the curve's maximum value
  - k the steepness of the curve



Nonlinear regression

- A nonlinear regression model can be written as:
$$Y = f(X, \theta) + \varepsilon$$
- Where
  - Y is the measured variable or dependent variable
  - X is the input variables or regressor variables or explanatory variables
  - f is the expectation function
  - θ is the coefficients
  - ε is the noise or disturbance
  - n is the number of training data points
  - p is the number of parameters

- We want to reduce the residual (SSE)
$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2$$
- Because $f(X, \theta)$ is a nonlinear function, so a closed form solution does not exist
- We should use an iterative algorithm to solve the problem
$$\arg\min_{\theta} SSE(\theta)$$

Minimizing the cost function…

- We can use Taylor series to iteratively evaluate

$$f(x_i, \theta) \approx f\left(x_i, \theta(k)\right) + u_i\left(\theta(k)\right)^T\left(\theta - \theta(k)\right)$$

- Where $u\left(\theta(k)\right)$ is the score vector (or Jacobian)
  - If θ has p elements, then u has also p elements
  - Its jth element is given by $\partial f(x, \theta)/(\partial \theta_j)$ evaluated at $\theta = \theta(k)$
- So, our residuals become:

$$\begin{aligned} SSE(\theta) &= \sum_{i=1}^{n}\left(y_i - f(x_i, \theta)\right)^2 \\ &\approx \sum_{i=1}^{n}\left(y_i - f\left(x_i, \theta(k)\right) - u_i\left(\theta(k)\right)^T\left(\theta - \theta(k)\right)\right)^2 \\ &= \sum_{i=1}^{n}\left(\hat{e}_i(k) - u_i\left(\theta(k)\right)^T\left(\theta - \theta(k)\right)\right)^2 \end{aligned}$$

- Where $\hat{e}_i(k) = y_i - f\left(x_i, \theta(k)\right)$ is ith working residual and it depends on the current guess $\theta(k)$

Minimizing the cost function…

$\hat{e}(k) \mapsto Y,\ U(\theta(k)) \mapsto X,\ (\theta - \theta(k)) \mapsto \beta$

- For the SSE formulation we just got

$$SSE(\theta) = \sum_{i=1}^{n} \left(\hat{e}_i(k) - u_i(\theta(k))^T (\theta - \theta(k))\right)^2$$

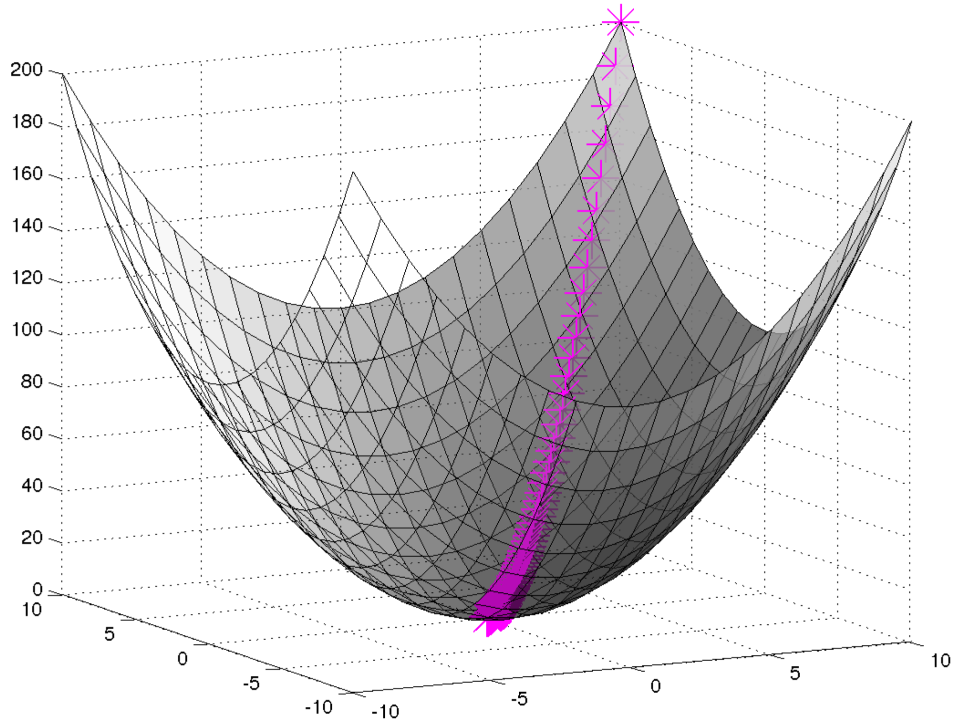
- We use matrix notation, let
  - $U(\theta(k))$ be a $n \times p$ matrix, with ith row $u_i(\theta(k))^T$
  - $\hat{e}(k) = (\hat{e}_1(k), \dots, \hat{e}_n(k))^T$
- The least squares estimate is

$$\Delta\hat{\theta} = \widehat{(\theta - \theta(k))}$$
$$= \left[U(\theta\,(k))^T U(\theta(k))\right]^{-1} U(\theta(k))^T \hat{e}(k)$$
$$\hat{\theta} = \theta(k) + \left[U(\theta\,(k))^T U(\theta(k))\right]^{-1} U(\theta(k))^T \hat{e}(k)$$

- We can use it as the basis for estimating and inferencing the next θ

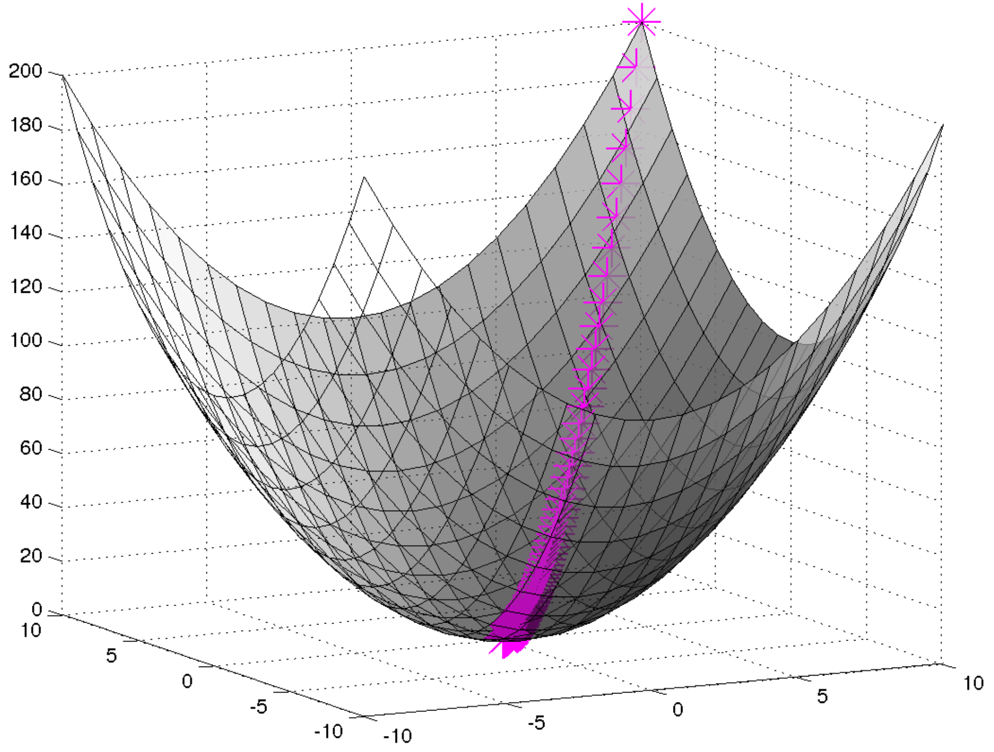
Linear form:

$$SSE(\beta) = \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)\right)^2$$
$$\hat{\beta} = (X^T X)^{-1} X^T Y$$



Gauss-Newton algorithms

- $\hat{\theta} = \theta(k) + \left[U(\theta\,(k))^T U(\theta(k))\right]^{-1} U(\theta(k))^T \hat{e}(k)$
- Algorithm
  - Select an initial guess $\theta(0)$, and compute $SSE(\theta(0))$
  - Set the initial counter $k = 0$
  - Compute $U(\theta(k))$ and $\hat{e}(k)$ with ith element $\hat{e}_i(k) = y_i - f(x_i, \theta(k))$, we get new estimator $\theta(k+1)$
  - Calculate the residual $SSE(\theta(k+1))$
  - If $SSE(\theta(k)) - SSE(\theta(k+1))$ is sufficiently small, STOP
  - Else: $k = k + 1$;
    - If k is too large, STOP
    - Otherwise, go to step 3.
- Many implementations will use a modification of the basic form



Levenberg-Marquardt algorithm

- The formulation is similar to the previous formulation
- Instead of

$$\Delta\hat{\theta} = \left[U\big(\theta\ (k)\big)^T U\big(\theta(k)\big)\right]^{-1} U\big(\theta(k)\big)^T \hat{e}(k)$$

- We have

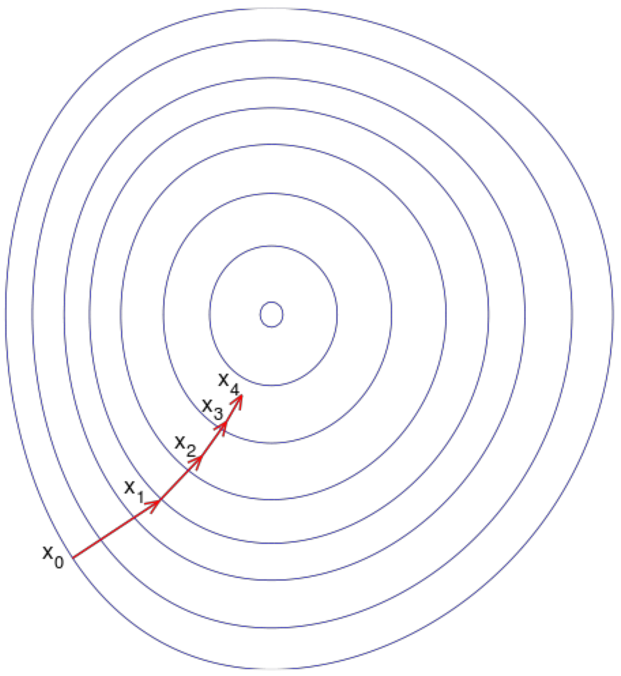
$$\Delta\hat{\theta} = \left[U\big(\theta\ (k)\big)^T U\big(\theta(k)\big) + \lambda\,\mathrm{diag}\Big(U\big(\theta\ (k)\big)^T U\big(\theta(k)\big)\Big)\right]^{-1} U\big(\theta(k)\big)^T \hat{e}(k)$$

It is a "damped version", where $\lambda \geq 0$, is adjusted at each iteration.

- If reduction of SSE is rapid, a smaller λ can be used, so the algorithm is close to Gauss-Newton
- If reduction of SSE is slow, λ can be increased, giving steps closer to the gradient descent direction
- The $\mathrm{diag}\Big(U\big(\theta\ (k)\big)^T U\big(\theta(k)\big)\Big)$ is to scale the gradient step based on the curvature when steps tend to be large

Remarks on nonlinear least squares

- Need to supply an initial guess
  - Can be trapped in local minima if the initial guess is not good
  - Multiple initial guesses can be tried to obtain the best results
  - Plotting SSE to get a visual understanding if possible
- Other methods
  - Decomposition
  - Gradient methods
  - Direct search
  - …



Example

- Logistic function,

$$y = \frac{L}{1 + e^{-k(x-x_0)}}$$

with only partial data

- We get

$$y = \frac{5.006}{1 + e^{-0.092(x-2.23)}}$$

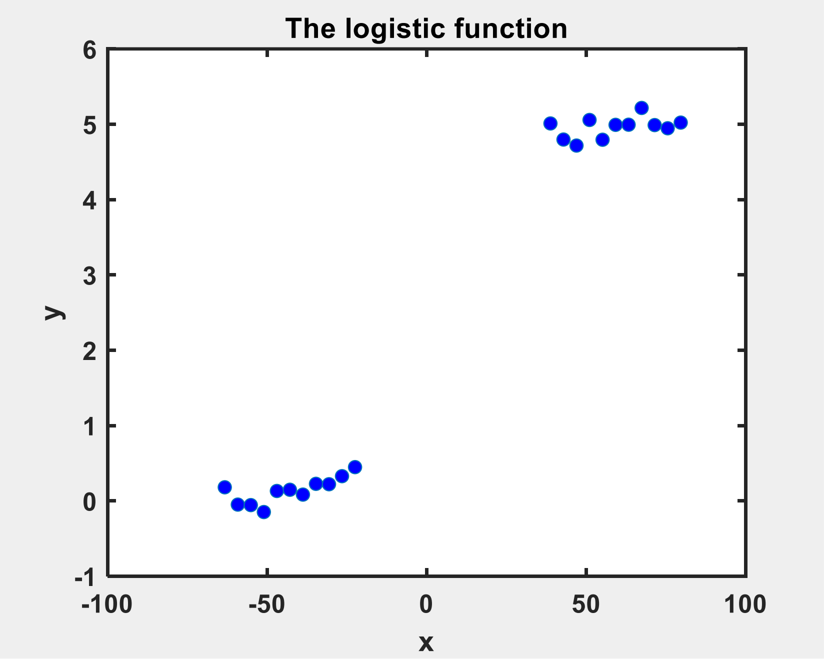
and

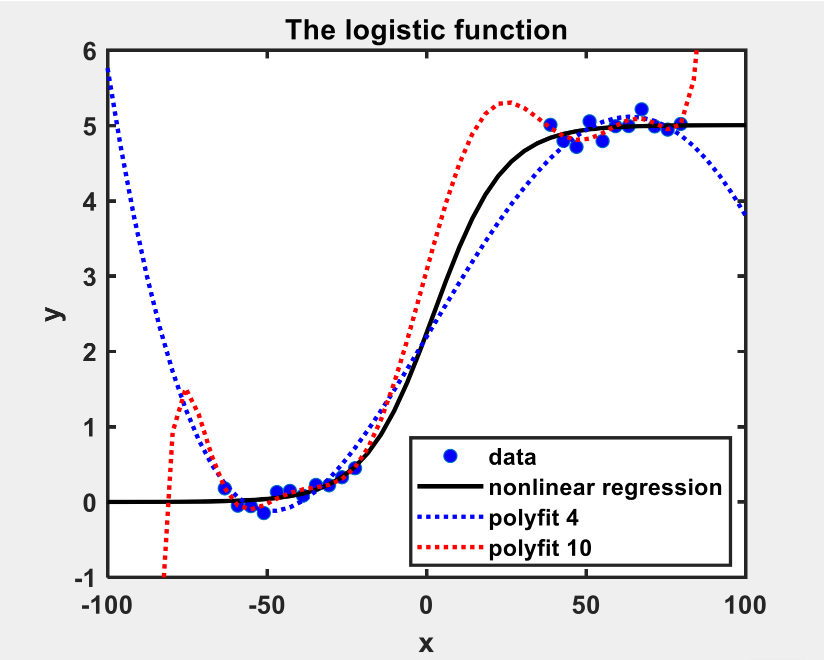
$$R^2 = 0.998$$

- Close to the true value

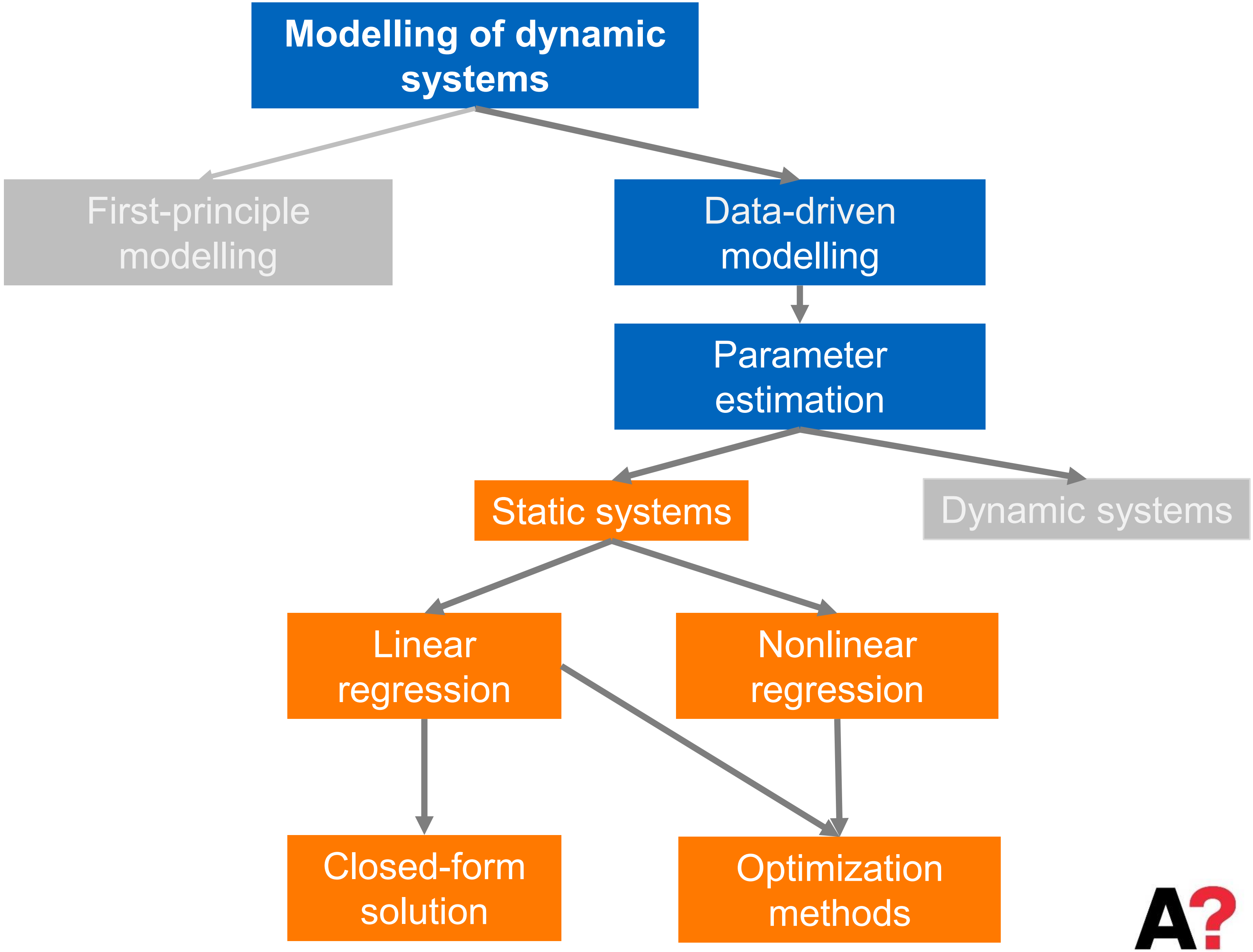
$$y = \frac{5}{1 + e^{-0.1(x-1)}}$$

- For comparison:
  - Fit with polynomial





What we have learnt



Summary

- The principles of linear regression
- Least-squares method in data fitting
- R^2, confidence interval, prediction interval
- Curvilinear regression
- Nonlinear regression

Readings

- Ch. 9, Howard J. Seltman, Experimental Design and Analysis, online book, 2015.
- Ch. 2-3, Weisberg, Applied Linear Regression, 2005.
- Ch. 6,11, Weisberg, Applied Linear Regression, 2005
- Wikipedia