Chapter 9: Data Fitting

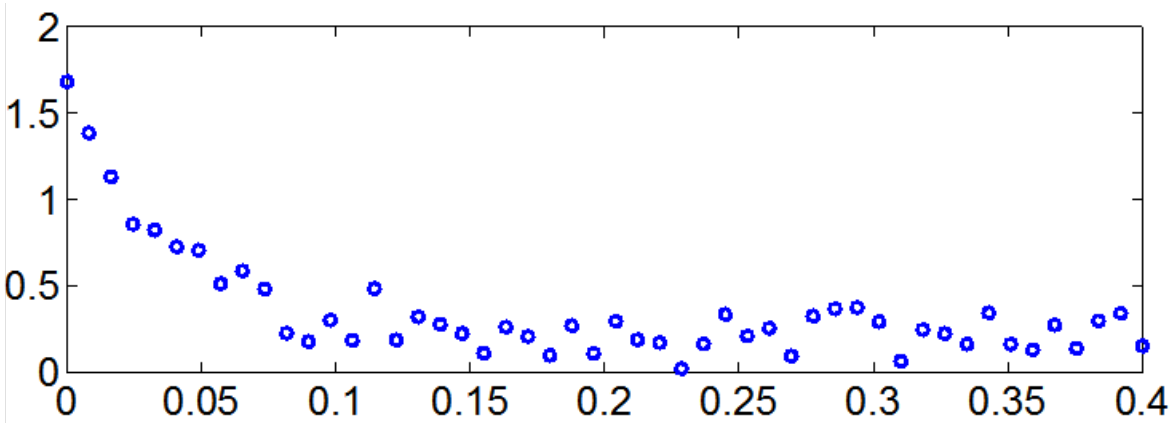
In Chapter 9, we mainly focus on the following contents:

- Introduction – read section 9.1 page 426–430.
- Basis functions – read section 9.1 page 430–432.
- Do **NOT** need read the rest part of Chapter 9.

More lectures on the data fitting referring to the course 02610 Optimization and Data Fitting.

Example: Parameter estimation

NMR techniques can be used to examine the molecular structure and distinguish different isotopes. Here is an NMR signal from frozen cod:



Data: Noisy measurements of the NMR signal from frozen cod at different time.

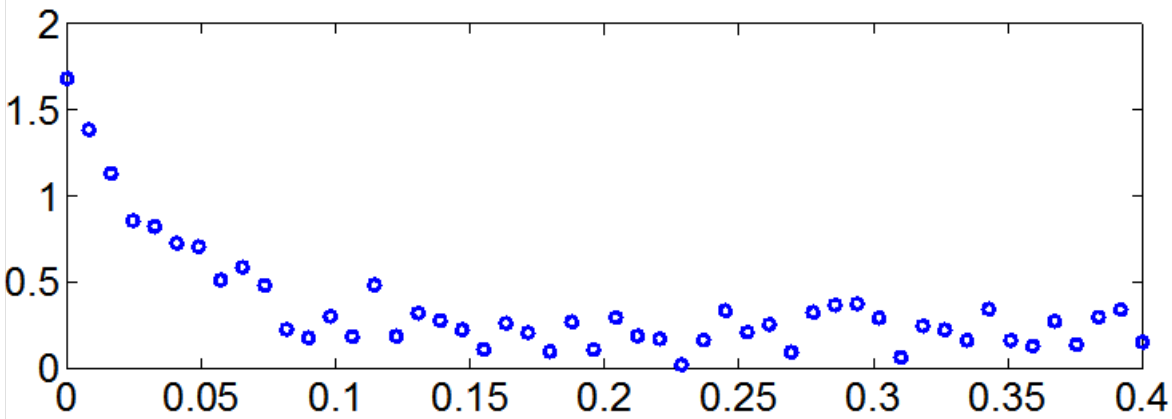
Model: The underlying mathematical model for the NMR signal is:

$$F(x) = c_0 + c_1 e^{-\lambda_1 x} + c_2 e^{-\lambda_2 x},$$

- x denotes the time and λ_1 and λ_2 are known.
- c_1 and c_2 are proportional to the amount of water containing the two kinds of protons.
- c_0 accounts for an undesired background (bias) in the measurements.

Example: Parameter estimation

NMR techniques can be used to examine the molecular structure and distinguish different isotopes. Here is an NMR signal from frozen cod:



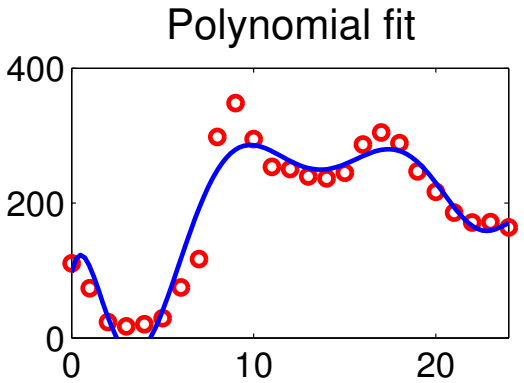
Data: Noisy measurements of the NMR signal from frozen cod at different time.

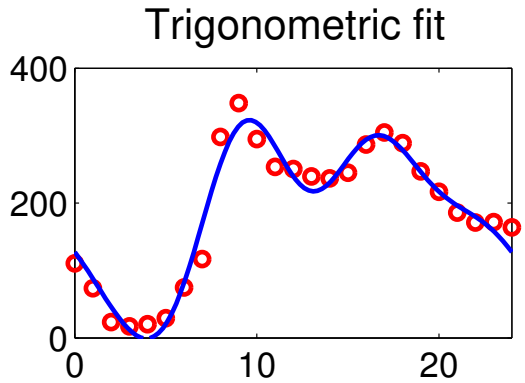
Model: The underlying mathematical model for the NMR signal is:

$$F(x) = c_0 + c_1 e^{-\lambda_1 x} + c_2 e^{-\lambda_2 x},$$

Goal: Estimate the unknown parameters c_0, c_1, c_2 in the model, and then compute the different kinds of water contents in the cod sample.

Example: Data approximation





Data: Measurements of air pollution, in the form of the NO concentration, over a period of 24 hours, at H. C. Andersens Boulevard on a Thursday.
Model: Polynomial and periodic models:

$$F(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$
$$F(x) = c_0 + c_1 \sin(\omega x) + c_2 \cos(\omega x) + c_3 \sin(2\omega x) + c_4 \cos(2\omega x) + \cdots$$

Goal: Fit a smooth curve to the measurements, so that we can compute the concentration at an arbitrary time between 0 and 24 hours.

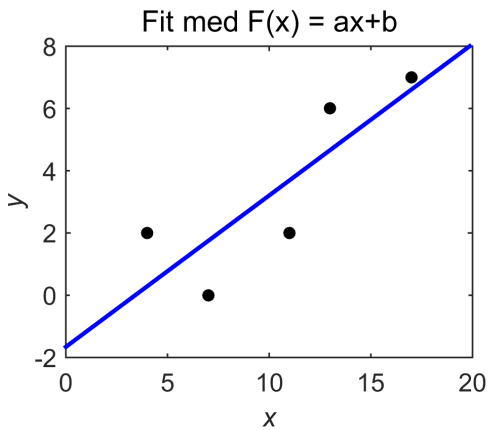
Introduction to data fitting

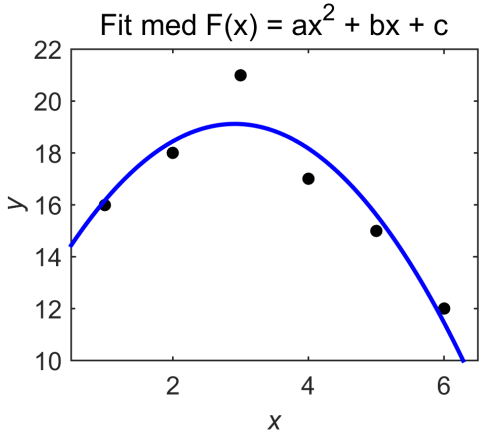
Ex. 1 page 429

x	4	7	11	13	17
y	2	0	2	6	7

Our exmaple

x	1	2	3	4	5	6
y	16	18	21	17	15	12





Given: data (x_k, y_k), $k = 0, 1, 2, \ldots, m$ with measurement errors.

We want: to fit a model – a function F – to these data.

Requirement: The model captures the "overall behavior" of the data without being too sensitive to the errors.

We usually assume that the abscissas x_k appear in non-decreasing order, i.e., $x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_m$.

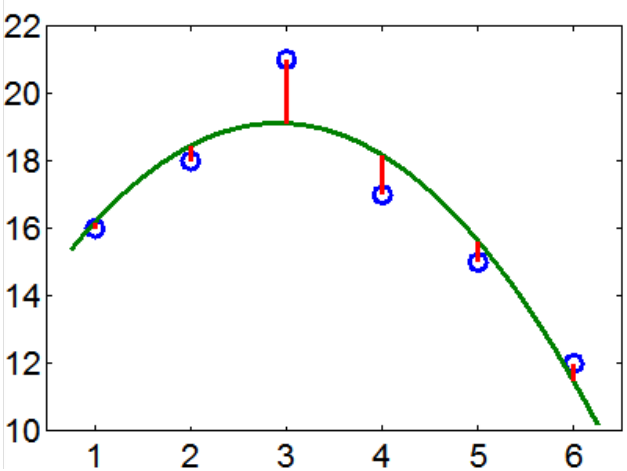
Formulation of data fitting problem

We assume that we have $m+1$ *data points*

- $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

which can be approximately described by the function F, i.e.

- $y_k \approx F(x_k), \qquad k = 0, 1, 2, \ldots, m.$



We introduce the *error* associated with the data points and the fit function F as:

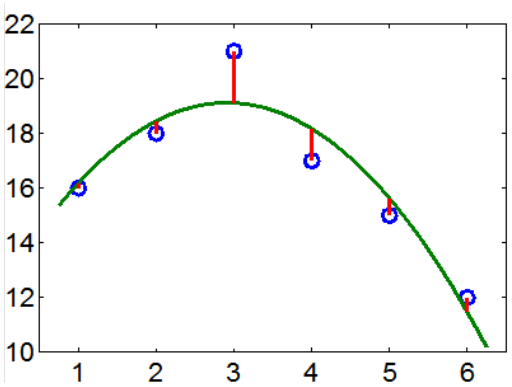
$$e_k = y_k - F(x_k), \qquad k = 0, 1, 2, \ldots, m$$

The absolute error $|e_k|$ is the vertical distance between the fit function and the data point.

We want to find the "best" *fit* function F to the data, which requires that the function F has a suitable number of degrees of freedom.

How can we find the best fit function?

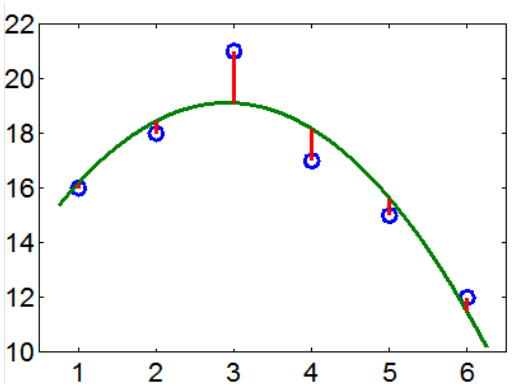
– ℓ_1 and ℓ_2 approximation (page 427)



Does it make sense to minimize the sum of the errors, i.e. min $\sum_{k=0}^{m} e_k$?

How can we find the best fit function?

– ℓ_1 and ℓ_2 approximation (page 427)

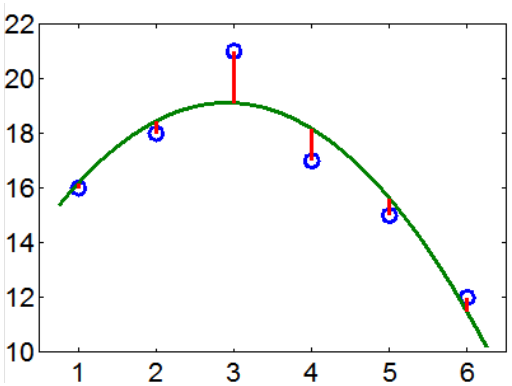


Does it make sense to minimize the sum of the errors, i.e. min $\sum_{k=0}^{m} e_k$? NO! The minimum is $-\infty$.

How about min$\left|\sum_{k=0}^{m} e_k\right|$?

How can we find the best fit function?

– ℓ_1 and ℓ_2 approximation (page 427)

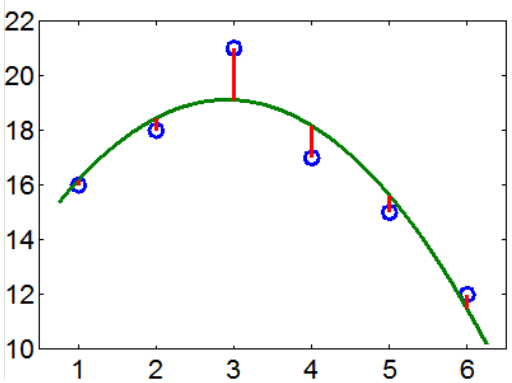


Does it make sense to minimize the sum of the errors, i.e. min $\sum_{k=0}^{m} e_k$? NO! The minimum is $-\infty$.

How about min$\left|\sum_{k=0}^{m} e_k\right|$? NO! The errors with different signs would cancel each other.

How can we find the best fit function?

– ℓ_1 and ℓ_2 approximation (page 427)



Does it make sense to minimize the sum of the errors, i.e. min $\sum_{k=0}^{m} e_k$? NO! The minimum is $-\infty$.

How about min$\left|\sum_{k=0}^{m} e_k\right|$? NO! The errors with different signs would cancel each other.

ℓ_1 approximation. Minimize the sum of the absolute errors:

$$\min \sum_{k=0}^{m} |e_k| = \min \sum_{k=0}^{m} |F(x_k) - y_k|.$$

Good idea, but the problem is difficult and expensive to solve (linear programming).

How can we find the best fit function?

– ℓ_1 and ℓ_2 approximation (page 427)

ℓ_2 approximation = **least-squares fit**.
Minimize the sum of squares of the errors:

$$\min \sum_{k=0}^{m} {e_k}^2 = \min \sum_{k=0}^{m} (F(x_k) - y_k)^2 \quad .$$

- **Easy to solve**.
- If the errors follow *a normal probability distribution*, then the minimization produces a best estimate of F.

Example: fit with a linear function (linear regression)

Find the coefficients a and b in the linear function $F(x) = a\,x + b$, which give the linear least-squares fit:

$$\min_{a,b} \varphi(a,b), \qquad \varphi(a,b) = \sum_{k=0}^{m} (a\,x_k + b - y_k)^2.$$

Necessary conditions at the minimum: $\frac{\partial \varphi}{\partial a} = 0$ and $\frac{\partial \varphi}{\partial b} = 0$, which lead to

$$\sum_{k=0}^{m} 2\,(a\,x_k + b - y_k)x_k = 0\ , \qquad \sum_{k=0}^{m} 2\,(a\,x_k + b - y_k) = 0.$$

The coefficient pair (a, b) is the solution of the *normal equations:*

$$\begin{bmatrix} s & p \\ p & m{+}1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ q \end{bmatrix} \qquad \begin{cases} p = \sum_{k=0}^{m} x_k & r = \sum_{k=0}^{m} x_k y_k \\ q = \sum_{k=0}^{m} y_k & s = \sum_{k=0}^{m} {x_k}^2 \end{cases}$$

Matrix-vector notation

Define

$$\boldsymbol{A} = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} a \\ b \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

The normal equations can be written as:

$$(\boldsymbol{A}^T \boldsymbol{A})\, \boldsymbol{c} = \boldsymbol{A}^T \boldsymbol{y}$$

It can be easily solved in Python.

```
>>> x = np.array([4, 7, 11, 13, 17])
>>> y = np.array([2, 0, 2, 6, 7])
>>> A = np.array([x,np.ones(5)]).T
>>> c = np.linalg.solve(A.T@A,A.T@y)
>>> xx = np.array([0,20])
>>> yy = c[0]*xx+c[1]
```

Generalization of data fitting problems (page 431–432)

Suppose that the fit function F is a *linear combination* of a set known *basis functions*:

$$F(x) = \sum_{j=0}^{n} c_j \, g_j(x).$$

The coefficients $c_0, c_1, \ldots, c_n$ are to be determined in order to provide the least-squares fit.

The function F should minimize

$$\varphi(c_0, c_1, \ldots, c_n) = \sum_{k=0}^{m} \big(F(x_k) - y_k\big)^2 = \sum_{k=0}^{m} \left(\sum_{j=0}^{n} c_j \, g_j(x_k) - y_k \right)^2 .$$

The necessary conditions for the minimum are $\frac{\partial \varphi}{\partial c_j} = 0$ for $j = 0, 1, \ldots, n$.

Matrix-vector notation

Define

$$\boldsymbol{A} = \begin{bmatrix} g_0(x_0) & g_1(x_0) & \cdots & g_n(x_0) \\ g_0(x_1) & g_1(x_1) & \cdots & g_n(x_1) \\ g_0(x_2) & g_1(x_2) & \cdots & g_n(x_2) \\ g_0(x_3) & g_1(x_3) & \cdots & g_n(x_3) \\ \vdots & \vdots & & \vdots \\ g_0(x_m) & g_1(x_m) & \cdots & g_n(x_m) \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}$$

where $\boldsymbol{A}$ is an $(m+1) \times (n+1)$ matrix, $\boldsymbol{c}$ is an $n+1$ vector, and $\boldsymbol{y}$ is an $m+1$ vector.

Matrix-vector notation

Define the error vector, which depends on $\boldsymbol{c}$,

$$\boldsymbol{r}(\boldsymbol{c}) = \begin{bmatrix} \sum_{j=0}^{n} c_j\, g_j(x_0) - y_0 \\ \sum_{j=0}^{n} c_j\, g_j(x_1) - y_1 \\ \vdots \\ \sum_{j=0}^{n} c_j\, g_j(x_m) - y_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}(1,:)\boldsymbol{c} - \boldsymbol{y}_1 \\ \boldsymbol{A}(2,:)\boldsymbol{c} - \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{A}(m+1,:)\boldsymbol{c} - \boldsymbol{y}_{m+1} \end{bmatrix} = \boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}$$

Matrix-vector notation

Define the error vector, which depends on $\boldsymbol{c}$,

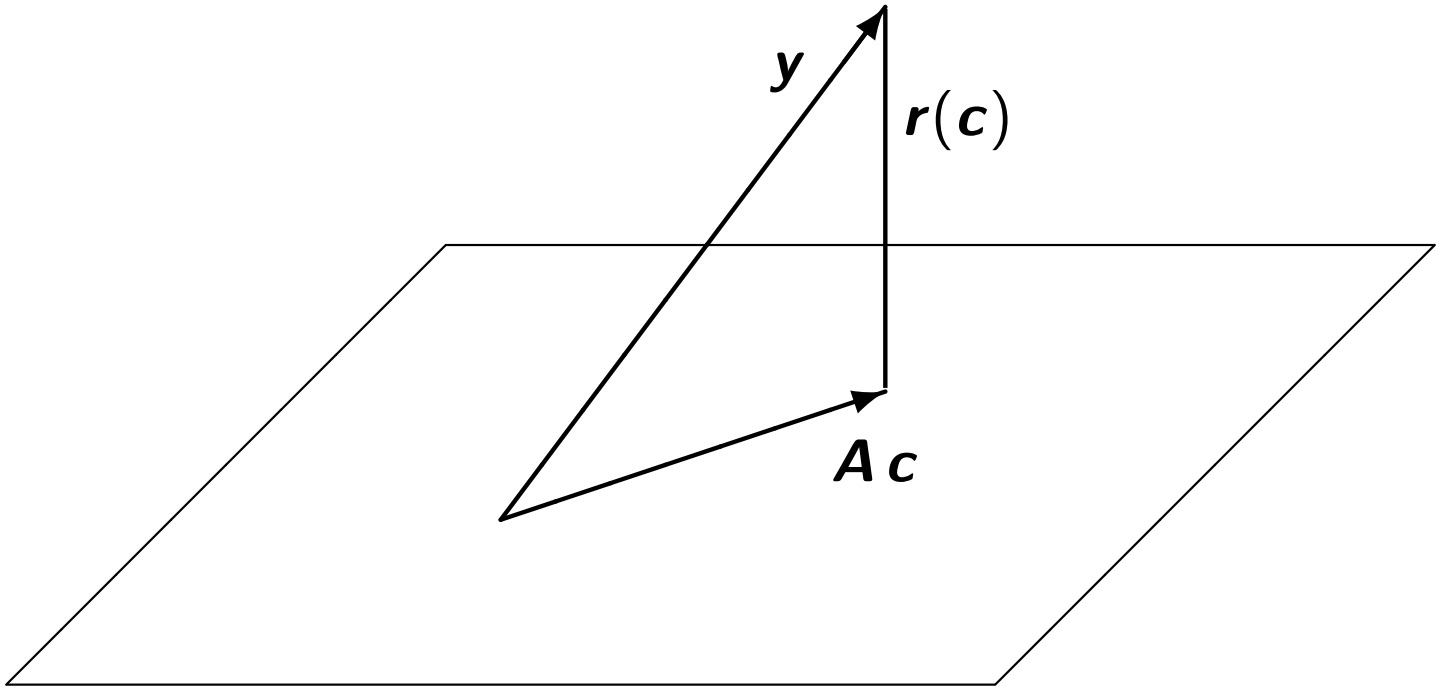
$$\boldsymbol{r}(\boldsymbol{c}) = \begin{bmatrix} \sum_{j=0}^{n} c_j\, g_j(x_0) - y_0 \\ \sum_{j=0}^{n} c_j\, g_j(x_1) - y_1 \\ \vdots \\ \sum_{j=0}^{n} c_j\, g_j(x_m) - y_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}(1,:)\boldsymbol{c} - \boldsymbol{y}_1 \\ \boldsymbol{A}(2,:)\boldsymbol{c} - \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{A}(m+1,:)\boldsymbol{c} - \boldsymbol{y}_{m+1} \end{bmatrix} = \boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}$$

The least-squares fit function F should minimize

$$\varphi(c_0, c_1, \ldots, c_n) = \sum_{k=0}^{m} \left(\sum_{j=0}^{n} c_j\, g_j(x_k) - y_k \right)^2 = \sum_{k=0}^{m} \boldsymbol{r}(\boldsymbol{c})_{k+1}^2,$$

which shows the distance between $\boldsymbol{A}\boldsymbol{c}$ and $\boldsymbol{y}$.

Normal equations



When $\boldsymbol{r}(\boldsymbol{c}) \perp \boldsymbol{A}\,\boldsymbol{c}$, we obtain the shortest $\boldsymbol{r}(\boldsymbol{c}) = \boldsymbol{A}\,\boldsymbol{c} - \boldsymbol{y}$. That is, $\boldsymbol{r}(\boldsymbol{c}) = \boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}$ must be orthogonal to all columns of $\boldsymbol{A}$:

$$\boldsymbol{A}^T \boldsymbol{r}(\boldsymbol{c}) = 0 \quad \Leftrightarrow \quad \boldsymbol{A}^T \boldsymbol{A}\,\boldsymbol{c} - \boldsymbol{A}^T \boldsymbol{y} = 0 \quad \Leftrightarrow \quad \boldsymbol{A}^T \boldsymbol{A}\,\boldsymbol{c} = \boldsymbol{A}^T \boldsymbol{y},$$

which is the *normal equations*.

Basis functions

Requirements for the basis functions g_j.

- The behavior of the fit function F strongly depends on the basis functions.
- They should be appropriate to describe the data.
  - Monotonically decreasing data $\rightarrow$ exponential functions.
  - Periodic data $\rightarrow$ sine and cosine.
- They must be linear independent, i.e.

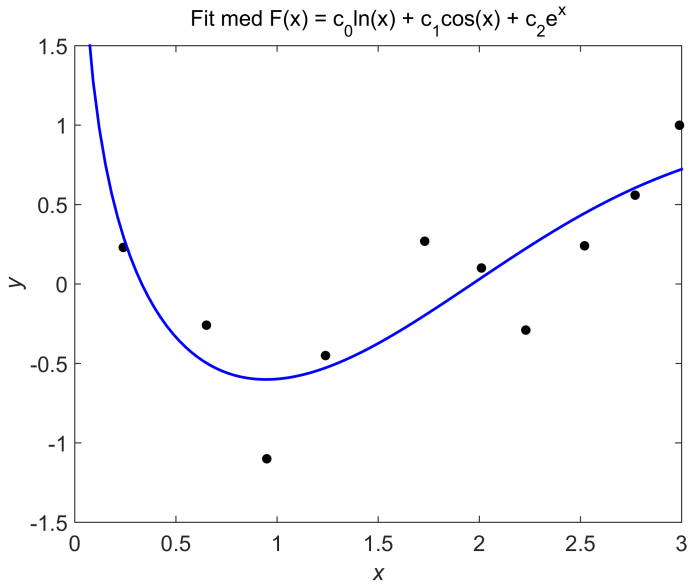
$$\alpha_0 g_0(x) + \alpha_1 g_1(x) + \alpha_2 g_2(x) + \cdots + \alpha_n g_n(x) = 0$$

  if and only if $\alpha_0 = \alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$.
- Because of linear independent, the matrix $\boldsymbol{A}^T \boldsymbol{A}$ in the normal equations has full rank, so the least-squares solution exists and is unique.
- If the condition of linear independent is not satisfied, the matrix $\boldsymbol{A}^T \boldsymbol{A}$ is singular, we cannot solve the normal equations.

Example 2, page 430–431

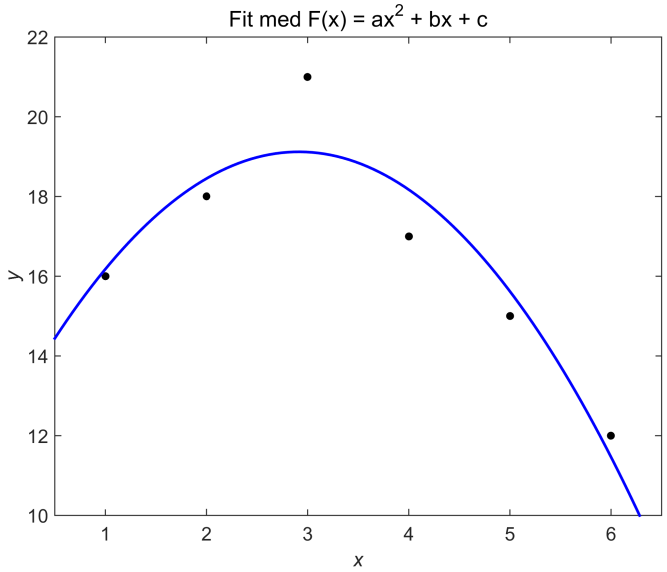
Fit the given data x and y with the basis functions $g_0(x) = \ln x$, $g_1(x) = \cos x$ and $g_2(x) = e^x$.

```
>>> A = np.array([np.log(x), np.cos(x), np.exp(x)]).T
>>> c = np.linalg.solve(A.T@A,A.T@y)
>>> xx = np.linspace(0,3);
>>> yy = c[0]*np.log(xx) + c[1]*np.cos(xx) + c[2]*np.exp(xx);
>>> plt.plot(x,y,'o',xx,yy,'-b')
```



Fit with polynomial: polyfit

```
>>> m = 2   #Set the degree of the polynomial.
>>> x = np.arange(1,7)
>>> y = np.array([16, 18, 21, 17, 15, 12])
>>> c = np.polyfit(x,y,m);
>>> xx = np.linspace(0.5,6.5);
>>> yy = np.polyval(c,xx);
>>> plt.plot(x,y,'.k',xx,yy,'-b')
>>> plt.show()
```



Summary

- Since computers can only accept the finite format, the float-point representation would lead to the *roundoff error*.
- Subtraction $(a - (a + \varepsilon))$ can lead to loss of significant digits.
- Different ways to calculate mathematical expressions can bring
  - different costs on operations,
  - different loss of significant digits.
- Errors in the data and during computation can lead to large perturbation on the solution.
- **Linear least-squares fit** to a *linear combination* of a set known linear independent *basis functions* can be obtained by solving *normal equations*

$$(\boldsymbol{A}^T \boldsymbol{A})\, \boldsymbol{c} = \boldsymbol{A}^T \boldsymbol{y}.$$

  Sometimes, normal equations can be difficult to solve numerically. (We will get back to this topic in the last part of the course.)

Exercises for the first 3 weeks

Time:	10am-12pm
Locations:	Building 308, Databar 001 & 009 & 017 & 127.
TAs:	3 TAs each time

If you have questions on the exercises, you can write your name on the blackboard.

All exercises are designed based on Python, but adaptable to any other programming languages.

Next week we will have lectures and exercises, so we will meet at 8am in Building 208.