Least-Squares method

- **Task**: There are a set of “experimental” points $(x_1,\ y_1)$, $(x_2,\ y_2)$,..,$(x_n,\ y_n)$, $x_1 < x_2 <...< x_n$ (a total of n points):

x_1	x_2	x_3	$\ldots$	x_n
y_1	y_2	y_3	$\ldots$	y_n

On the other hand, there is an analytical expression that the data should satisfy. In general, the expression is represented by a function of x with parameters $A_0, .., A_m$:

$$y = f(x;\ A_0,..,A_m)$$

In general, $m + 1 < n \Rightarrow$ no guarantee we can find the parameters such that the functions passes exactly through all the data points:

$$f(x_i;\ A_0,..,A_m) = y_i$$

However, we can formulate the *deviation* of the calculated values from the “experimental” ones $f(x_i;\ A_0,.., A_m) - y_i$.

$$\varepsilon(A_0,.., A_m) = \sum_{i=1}^{n} \left(f(x_i; A_0,.., A_m) - y_i\right)^2$$

$$\varepsilon(A_0,\dots,A_m)=\sum_{i=1}^{n}\left(f(x_i;A_0,\dots,A_m)-y_i\right)^2$$

The summation runs over all the "experimental" points $x_1,\ x_2,\dots,x_n$ and the function $\varepsilon(A_0,\dots,A_m)$ depends on the parameters $A_0,\dots,A_m$, not on the argument x. $\varepsilon(A_0,\dots,A_m)$ is a non-negative function and can be zero if and only if $f(x_i;\ A_0,\dots,A_m)=y_i$.

- The optimal approximation can be obtained by *minimizing* ε as a function of $A_0,\dots,A_m$.

The partial derivative of ε with respect to each of the parameters $A_0,\dots,A_m$:

$$\frac{\partial}{\partial A_k}\varepsilon(A_0,\dots,A_m)=\sum_{i=1}^{n}\frac{\partial}{\partial A_k}\left(f(x_i;A_0,\dots,A_m)-y_i\right)^2$$
$$=\sum_{i=1}^{n}2\left((f(x_i;A_0,\dots,A_m)-y_i)\cdot\frac{\partial}{\partial A_k}f(x_i;A_0,\dots,A_m)\right)$$

The optimal parameters are obtained by solving a system of $m+1$ equations with $m+1$ variables:

$$\sum_{i=0}^{n}\left((f(x_i;A_0,\dots,A_m)-y_i)\cdot\frac{\partial}{\partial A_k}f(x_i;A_0,\dots,A_m)\right)=0,\quad k=0,\dots,m$$

- In the case when $f(x_i;\ A_0,..,A_m)$ is a polynomial function of parameters $A_0,..,A_m$:

$$\boxed{f(x; A_0,..,A_m) = \sum_{j=0}^{m} A_j x^j}$$

$$\varepsilon(A_0,..,A_m) = \sum_{i=1}^{n}\left(f(x_i; A_0,..,A_m) - y_i\right)^2 = \sum_{i=1}^{n}\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)^2$$

$$\frac{\partial}{\partial A_k}\varepsilon(A_0,..,A_m) = \frac{\partial}{\partial A_k}\left(\sum_{i=1}^{n}\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)^2\right) = \sum_{i=1}^{n}\frac{\partial}{\partial A_k}\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)^2$$

$$\frac{\partial}{\partial A_k}\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)^2 = 2\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)\cdot\underbrace{\frac{\partial}{\partial A_k}\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)}_{x_i^k} =$$

$$= 2\left(\sum_{j=0}^{m} A_j x_i^j - y_i\right)\cdot x_i^k = 2\left(\sum_{j=0}^{m} A_j x_i^j \cdot x_i^k - y_i x_i^k\right)$$

$$\frac{\partial}{\partial A_k}\varepsilon(A_0,..,A_m) = \sum_{i=1}^{n} 2\left(\sum_{j=0}^{m} A_j x_i^{j+k} - y_i x_i^k\right) = 2\sum_{i=1}^{n}\sum_{j=0}^{m} A_j x_i^{j+k} - 2\sum_{i=1}^{n} y_i x_i^k$$

$$2\sum_{i=1}^{n}\sum_{j=0}^{m} A_j x_i^{j+k} - 2\sum_{i=1}^{n} y_i x_i^k = 0$$

Changing the summation order, we obtain a system of **linear** equations with unknown A_j:

$$\sum_{j=0}^{m} \underbrace{\left(\sum_{i=1}^{n} x_i^{k+j}\right)}_{\substack{\text{coefficient of}\\ k\text{-th row,}\\ j\text{-th column}}} A_j = \underbrace{\sum_{i=1}^{n} y_i x_i^k}_{\substack{\text{free term}\\ \text{of } k\text{-th row}}} \quad , \quad k = 0, \ldots, m$$

- <u>Example</u>: linear "regression"

$$f(x; A_0, A_1) = A_0 + A_1 x$$

$$\begin{pmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \cdot \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

Quadratic:

$$f(x; A_0, A_1, A_2) = A_0 + A_1 x + A_2 x^2$$

$$\underbrace{\begin{pmatrix} n & \sum_i x_i & \sum_i x_i^2 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \end{pmatrix}}_{\mathbf{R}} \cdot \underbrace{\begin{pmatrix} A_0 \\ A_1 \\ A_2 \end{pmatrix}}_{\mathbf{A}} = \underbrace{\begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \sum_i x_i^2 y_i \end{pmatrix}}_{\mathbf{Z}}$$

- Note that all the matrix elements R_{kl} are equal if the sum of indices $k+l$ is equal. Thus, not $(m+1)^2$, but only $2m$ sums must be calculated!
- ***Hint***: An efficient way to program it would be to calculate **R** and **Z** would be to run a loop over i, then over all necessary powers p to compute x_i^p, then to update the sums. Finally, the sums should be copied to the corresponding matrix elements.
- ***Hint***: Do not use explicit **`x(i)**j`** – it will be highly inefficient.

$$\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}\begin{pmatrix} \\ \\ \\ \end{pmatrix} \longrightarrow \begin{pmatrix} n & \sum_i x_i & \sum_i x_i^2 \\ & & \\ & & \end{pmatrix} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \sum_i x_i^2 y_i \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} n & \sum_i x_i & \sum_i x_i^2 \\ & & \sum_i x_i^3 \\ & & \sum_i x_i^4 \end{pmatrix} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \sum_i x_i^2 y_i \end{pmatrix} \rightarrow \begin{pmatrix} n & \sum_i x_i & \sum_i x_i^2 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \end{pmatrix} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \sum_i x_i^2 y_i \end{pmatrix}$$

Programming task

- ***Input***:
  - Degree of the fitting polynomial m;
  - The number of "experimental" points n;
  - The number points themselves (X's and Y's by pairs).
- ***Output***:
  - Values of the coefficients $A_0, .., A_m$ obtained
  - Value I of the definite integral of the fitted polynomial in the interval from the minimum value of all X's to the maximum value of all X's.

$$I = \int_{\min\{x_i\}}^{\max\{x_i\}} \sum_{k=0}^{m} A_k x^k dx$$

- ***Note***:
  - The program must be general, i.e, work for any m and n.

- ***Hint***: The above matrix **R** is defined as `R(0:m,0:m)`
  - However, linear equation solvers usually use matrices from 1 to *n*:

```
subroutine GAUSS(A,B,X,m)
integer :: m
double precision, dimension(1:m,1:m) :: A
double precision, dimension(1:m) :: B, X
```

  - The following call serves to overcome this problem:

```
call GAUSS(R,Z,A,n+1)
```

  where **n+1** is the *size* of the matrix (and vector). No modification of the subroutine is needed is this case.

- ***Hint***: *Calculation of a polynomial*.
  - A polynomial $P(x) = \sum_{k=0}^{N} b_k x^k$ is *never* calculated according to this formula directly since it requires about $n^2/2$ multiplications.
  - The easiest way to implement it efficiently corresponds to the following formula:

$$P(x) = \left(\left(b_N x + b_{N-1}\right) x + b_{N-2}\right) x \ldots + b_0$$