DATA FITTING

Intensive Computation
2013/2014

Annalisa Massini

Data fitting

Data fitting concerns the problem of fitting discrete data to obtain intermediate estimates.

There are two general approaches two curve fitting:

- **Interpolation**
  - *Data is very precise.*
  - The strategy is to pass a curve (or a series of curves) through each of the points.
- **Approximation**
  - *Data exhibit a significant degree of scatter*.
  - The strategy is to derive a single curve that represents the general trend of the data.

Definition

Interpolation

- data points, obtained by sampling or experimentation, represent the values of a function $f(x)$for a limited number of values of the independent variable
- $f(x)$ is a function able to describe the relation between the set of values x and the set of values $y = f(x)$
- with interpolation we can compute new data points within the range of a discrete set of known data points

Definition

- Given a sequence of n real values x_k
- Given a value y_k for each x_k
- We want to find a function f - in a function family – such that

$$f(x_k)=y_k \qquad k=1, .., n$$

- (x_k, y_k) are said **data point**
- f is said the **interpolant** for the given point

Definition

There are many different interpolation methods

Some of the concerns to consider when choosing an algorithm are:

- How accurate is the method?
- How expensive is it?
- How smooth is the interpolant?
- How many data points are needed?

Interpolation methods

Simple interpolation methods are:

- **Linear interpolation**
- **Polynomial interpolation**
- **Spline interpolation**

Linear interpolation

- Linear interpolation considers pairs of data points
- Given (x_a,y_a) and (x_b,y_b) the interpolant is given by:

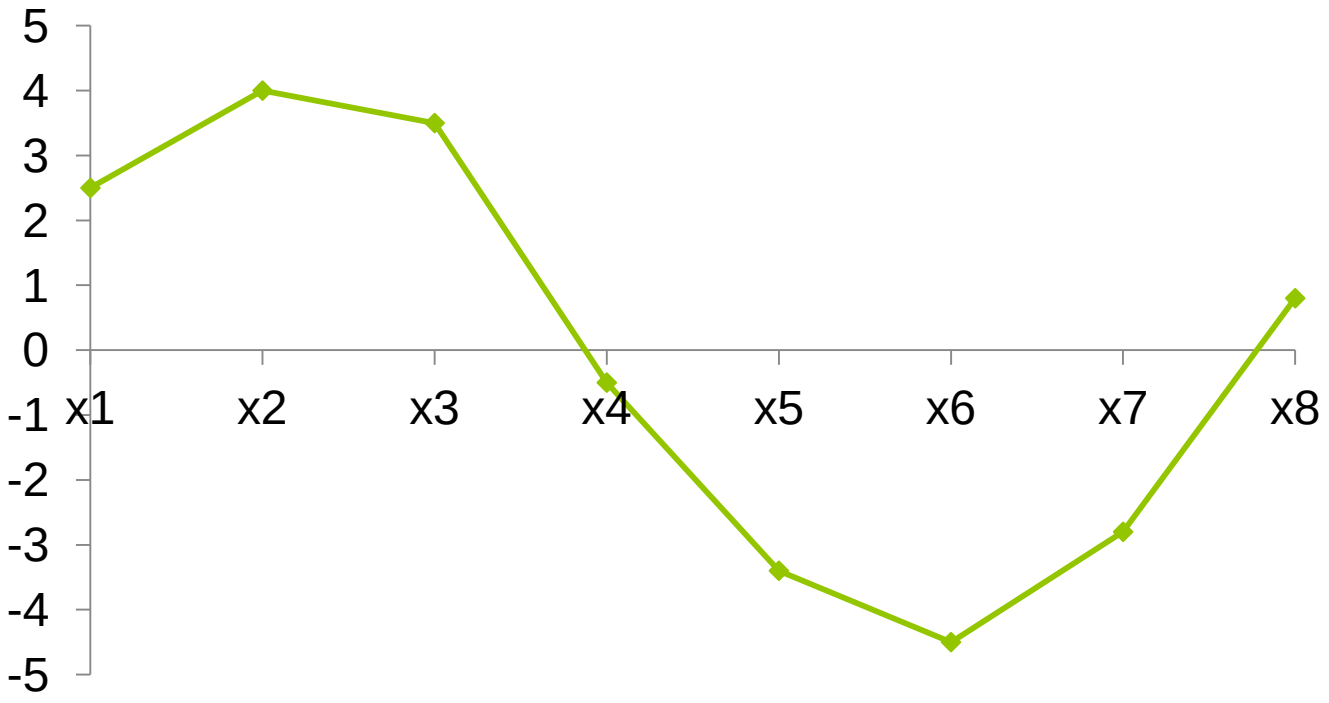
$$y = y_a + (y_b - y_a)\frac{x - x_a}{x_b - x_a}$$

- Linear interpolation is quick and easy, but it is not very precise

- With linear interpolation the interpolant is not **differentiable** at the point x_k

Linear interpolation

- Example

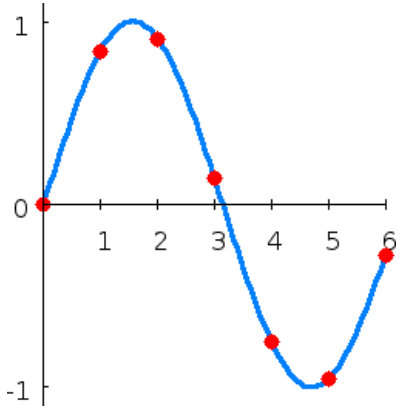


Polynomial interpolation

- Polynomial interpolation is a generalization of linear interpolation

- We now replace this interpolant with a polynomial of higher degree

- Given *n* data points, we look for the **polynomial** of **degree at most *n*−1** going through all the data points



Polynomial interpolation

Some questions are:

- Does the polynomial exist?
- If the polynomial exist, is it unique?
- If the polynomial exist and is unique, is it possible to compute it?
- Does the polynomial that interpolates $(x_k,f(x_k))$ approximate better than other polynomials?

Polynomial interpolation

n=1

- Given **two points** (x0,y0), (x1,y1) find the polynomial of degree one passing through the two points
- is equivalent to find the **line** passing through the two given points retta che passa per due punti assegnati

n=2

- Given **three points** (x0,y0), (x1,y1), (x2,y2) find the polynomial of degree two passing through the three points
- is equivalent to find the **parabola** $y = a1x^2 + a2x + a3$ passing through the three given points

Polynomial interpolation

Hence

- **n=3** given data points
- **m=3** conditions (data points belong to the parabola)
- **3 coefficients** to determine

In general:

Number of conditions

=

Number of coefficients to determine

⇩

Well posed problem

Polynomial interpolation

Example

Given the three data points (0, 1) (1, 2) (2, 9)
we want to comput the interpolant

$$P_2(x)=ax^2+bx+c$$

First approach – **Method of undertermined coefficients**
From the interpolation conditions we obtain the system

$$\begin{cases} c = 1 \\ a + b + c = 2 \\ 4a + 2b + c = 9 \end{cases} \Rightarrow \qquad \begin{cases} c = 1 \\ b = -2 \\ a = 3 \end{cases}$$

Polynomial interpolation

In general:

Given n points (x_k, y_k) $k=1, \ldots, n,$ find the **interpolant** as the **polynomial** of degree at most $n-1$

$$P_{n-1}(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_0 x^0$$

with the method of **undetermined coefficients** corresponds to solve the **linear equation system**

$$\begin{cases} c_{n-1}x_1^{n-1} + c_{n-2}x_1^{n-2} + \ldots + c_0 x_1^0 = y_1 \\ c_{n-1}x_2^{n-1} + c_{n-2}x_2^{n-2} + \ldots + c_0 x_2^0 = y_2 \\ \qquad\qquad \ldots \\ c_{n-1}x_n^{n-1} + c_{n-2}x_n^{n-2} + \ldots + c_0 x_n^0 = y_n \end{cases}$$

Polynomial interpolation

- In general the system could have no solution, that is the problem is not well posed (a problem is well posed if the coefficients **matrix not singular**, that is its determinant is not 0)

- When the linear system is not *easy* to solve we can use another approach: **Lagrange interpolation**

Polynomial interpolation

Second approach - **Lagrange**

The interpolant is the polynomial:

$$P_{n_1}(x) = y_{n_1} l_{n_1}(x) + y_{n_2} l_{n_2}(x) + \ldots + y_0 l_0(x)$$

that is the polynomial is a
linear combination of the $l_i(x)$ polynomial
that are **Lagrange basis polynomials**
with y_i as coefficients

Polynomial interpolation

In the polynomial:

$$P_{n_1}(x) = y_{n_1} l_{n_1}(x) + y_{n_2} l_{n_2}(x) + \ldots + y_0 l_0(x)$$

The **Lagrange basis polynomials** are given by:

$$l_j(x) = \frac{(x - x_{n-1})\ldots(x - x_{j+1})(x - x_{j-1})\ldots(x - x_0)}{(x_j - x_{n-1})\ldots(x_j - x_{j+1})(x_j - x_{j-1})\ldots(x_j - x_0)}$$

$l_j(x)$ is a *n-1* degree polynomial and has the property:

$$l_j(x_k) = \begin{cases} 1 & se\, k = j \\ 0 & se\, k \neq j \end{cases}$$

Polynomial interpolation

Example

- Compute the Lagrange interpolant interpolating the data points

(0, 1) (1, 2) (2, 9)

- We have to compute the **two degree polynomial**:

$$P_2(x) = \sum_{0}^{2} y_i l_i(x) = y_2 l_2(x) + y_1 l_1(x) + y_0 l_0(x)$$

Polynomial interpolation

- The **Lagrange basis polynomials** are:

$$l_0(x) = \frac{(x-x_2)(x-x_1)}{(x_0-x_2)(x_0-x_1)} = \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{1}{2}(x-1)(x-2)$$

$$l_1(x) = \frac{(x-x_2)(x-x_0)}{(x_1-x_2)(x_1-x_0)} = \frac{(x-2)(x-0)}{(1-2)(1-0)} = -x(x-2)$$

$$l_2(x) = \frac{(x-x_1)(x-x_0)}{(x_2-x_1)(x_2-x_0)} = \frac{(x-1)(x-0)}{(2-1)(2-0)} = \frac{1}{2}x(x-1)$$

Polynomial interpolation

By substituting we obtain:

$$P_2(x) = y_2 l_2(x) + y_1 l_1(x) + y_0 l_0(x) =$$
$$= 9\frac{1}{2}x(x-1) - 2x(x-2) + \frac{1}{2}(x-1)(x-2) =$$
$$= 3x^2 - 2x + 1$$

Observe that the polynomial $P_2(x)$ obtained by the **Lagrange method** is the same obtained with the **undetermined cofficients method**

Polynomial interpolation

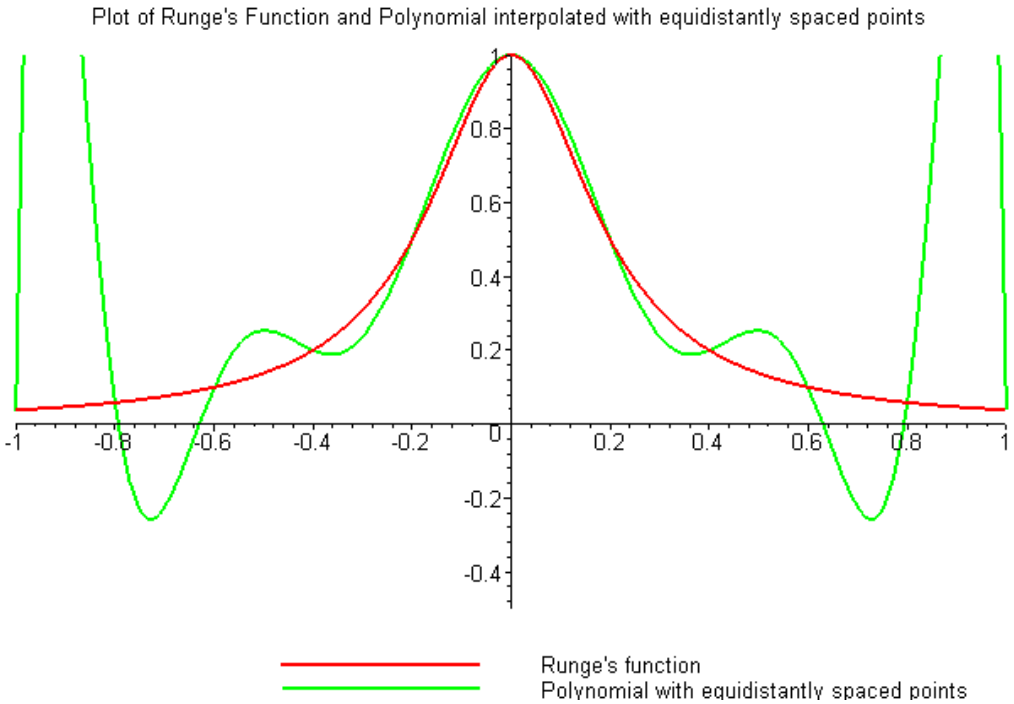
- The interpolant obtained by polynomial interpolation is clearly differentiable

Polynomial interpolation also has some ***disadvantages***:

- Calculating the interpolating polynomial is **computationally expensive** compared to linear interpolation.
- the **shape** of the resulting curve, especially for very high or low values of x, may be contrary to commonsense
- The interpolation **error** is proportional to the distance between the data points to the power *n*

Polynomial interpolation

Polynomial interpolation may exhibit oscillatory artifacts, especially at the end points - **Runge's phenomenon**



Spline interpolation

Spline interpolation:

- is a piecewice interpolation
- uses polynomials in each interval $[x_k, x_{k+1}]$
- uses low-degree polynomials
- incurs a smaller error than linear interpolation and the interpolant is smoother
- the interpolant is easier to evaluate than a high-degree polynomial such as the interpolant
- does not suffer from Runge's phenomenon.

Spline interpolation

- In a **cubic** spline each piece is a third-degree polynomial
- It is easy to compute

Each piece of the spline is a polynomial of the form

$$p(x) = ax^3 + bx^2 + cx + d$$

Since we have to determine the **four coefficients** *a*, *b*, *c* and *d,* we need **four equations**

Spline interpolation

- Two conditions are obtained by imposing the points belong to the polynomial, that is:

$$y_i = ax_i^3 + bx_i^2 + cx_i + d$$
$$y_{i+1} = ax_{i+1}^3 + bx_{i+1}^2 + cx_{i+1} + d$$

- The other two conditions are obtained by using the **derivatives**
- For example we can impose
  - the first derivatives match at the points
  - the second derivatives match at the points

- Note that the derivative of the polynomial is very simple and is:

$$p'(x) = 3ax^2 + 2bx + c$$

Approximation

If available data are:

- a large set
- affected by errors

it is possible capturing the trend in the data by assigning a single function across the entire range.

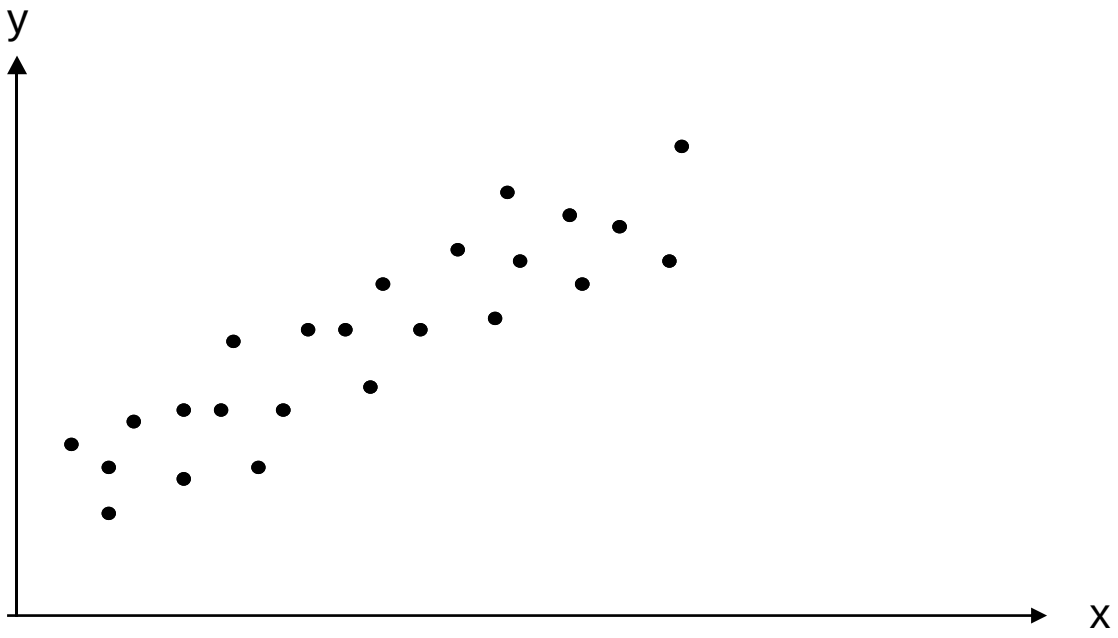
In the process we exploit the information given by data, taking into account that data are affected by errors

We need:

1. To choose the type of approximating function
2. To choose a measure of the difference between data and approximating function

Linear approximation

Suppose that there are n pairs of data, $\{(x_i, y_i)\}$, $i=1, 2,.. , n$ and a plot of these data appears as



What is a plausible mathematical model describing *x* and *y* relation?

Linear approximation

- Suppose variable y is related to variable x and their relationships can be expressed, in general, as

$$y = g(x)$$

where g is the general expression for a function;

y is the dependent (or response) variable;

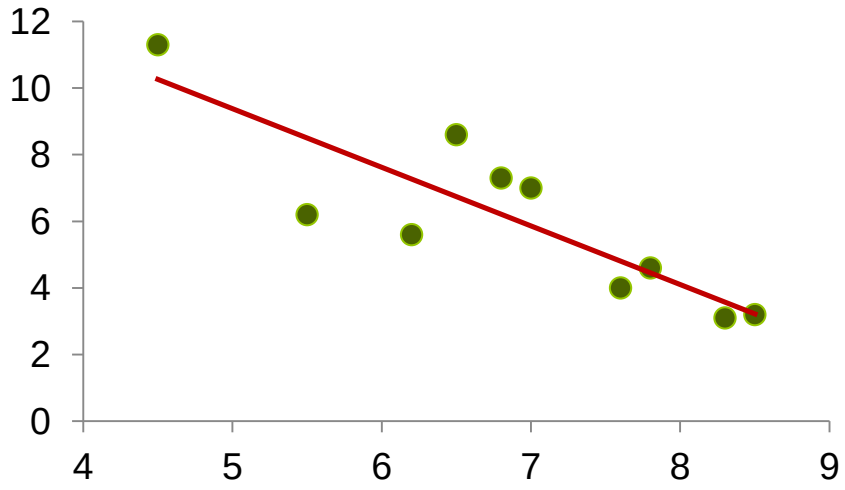
x is the independent (or explanatory) variables.

- If available data show a **linear dependence** it is possible to consider a **regression line**
- For each point we have:

$$y_i = a\,x_i + b + \text{residual}_i$$

that is

$$y_i = a\,x_i + b + \varepsilon_i$$



Linear approximation

The questions are:

- **Which line is the most representative?**
- **How can we chhose the coefficients of the line?**

We aim to minimise the residuals (or errors):

- Residuals should be ***small*** both positive and negative residuals
- We need a function of residuals independent of the sign, such as the absolute value or the square

Least square method

The least squares (LS) criterion requires that

the **sum of the squares** of
residuals (or errors, deviations)
is **minimum**

that is

$$\varepsilon(a,b) = \sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

Least square method

- Hence we want determine $\tilde{a}$ and $\tilde{b}$ such that the quantity

$$\varepsilon(\tilde{a},\tilde{b}) = \min_{a,b} \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

is minimum

that is we want to solve the **system of normal equations**:

$$\begin{cases} \dfrac{\partial \varepsilon}{\partial a} = 0 \\ \dfrac{\partial \varepsilon}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} 2\sum_{i=1}^{n} [y_i - (ax_i + b)](-x_i) = 0 \\ 2\sum_{i=1}^{n} [y_i - (ax_i + b)](-1) = 0 \end{cases}$$

Least square method

This results in a *system* with **two equations** and **two unknowns**

$$\begin{cases} (\sum_{i=1}^{n} x_i^2)a + (\sum_{i=1}^{n} x_i)b = \sum_{i=1}^{n} x_i y_i \\ (\sum_{i=1}^{n} x_i)a + mb = \sum_{i=1}^{n} y_i \end{cases}$$

This minimization problem has a unique solution

The solution of the system gives the
coefficients of the line realizing the **best data fitting**
in the **least square** sense

Least square method

- Data can be approximated with a **polynomial** instead of a line
- The **polynomial of second degree** giving the best fit in the least square sense is a polynomial in the form:

$$p(x) = \tilde{a}x^2 + \tilde{b}x + \tilde{c}$$

where the coefficients are determined by imposing:

$$\varepsilon(\tilde{a}, \tilde{b}, \tilde{c}) = \min_{a,b,c} \sum_{i=1}^{n} [y_i - p(x_i)]^2 =$$

$$= \min_{a,b,c} \sum_{i=1}^{n} [y_i - (ax_i^2 + bx_i + c)]^2$$

Least square method

- This results in the following **system of normal equations**:

$$\begin{cases} \dfrac{\partial \varepsilon}{\partial a} = 0 \\ \dfrac{\partial \varepsilon}{\partial b} = 0 \\ \dfrac{\partial \varepsilon}{\partial c} = 0 \end{cases} \Rightarrow \begin{cases} (\sum_{i=1}^{n} x_i^4)a + (\sum_{i=1}^{n} x_i^3)b + (\sum_{i=1}^{n} x_i^2)c = \sum_{i=1}^{n} x_i^2 y_i \\ (\sum_{i=1}^{n} x_i^3)a + (\sum_{i=1}^{n} x_i^2)b + (\sum_{i=1}^{n} x_i)c = \sum_{i=1}^{n} x_i y_i \\ (\sum_{i=1}^{n} x_i^2)a + (\sum_{i=1}^{n} x_i)b + nc = \sum_{i=1}^{n} y_i \end{cases}$$

The solution of the system gives the

coefficients of the parabola realizing the **best data fitting**

in the **least square** sense

Least square method

If the relationship between the independent variable *x* and the dependent variable *y* is modelled as an *n*th order polynomial

$$p(x) = \tilde{a}_k x^k + \tilde{a}_{k-1} x^{k-1} + \cdots + \tilde{a}_1 x + \tilde{a}_0$$

then the **system of normal equations** is:

$$\begin{cases} (\sum_{i=1}^{n} x_i^{2k}) a_k + \cdots + (\sum_{i=1}^{n} x_i^{k+1}) a_1 + (\sum_{i=1}^{n} x_i^{k}) a_0 = \sum_{i=1}^{n} x_i^k y_i \\ \cdots \\ (\sum_{i=1}^{n} x_i^{k+1}) a_k + \cdots + (\sum_{i=1}^{n} x_i^{2}) a_1 + (\sum_{i=1}^{n} x_i) a_0 = \sum_{i=1}^{n} x_i y_i \\ (\sum_{i=1}^{n} x_i^{k}) a_k + \cdots + (\sum_{i=1}^{n} x_i) a_1 + n a_0 = \sum_{i=1}^{n} y_i \end{cases}$$

- **Polynomial regression** fits a nonlinear relationship between the value of *x* and the corresponding y

Least square method

The polynomial regression model can be expressed in **matrix form**

- Considering the matrix *A*

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ . & . & . & . & . \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{pmatrix}$$

The **coefficient matrix** of the **system of normal equations** is given by:

$$A^T * A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ . & . & . & . \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ . & . & . & . & . \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{pmatrix}$$

Least square method

- The **vector** on the right hand side is given by:

$$A^T y = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ . & . & . & . \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

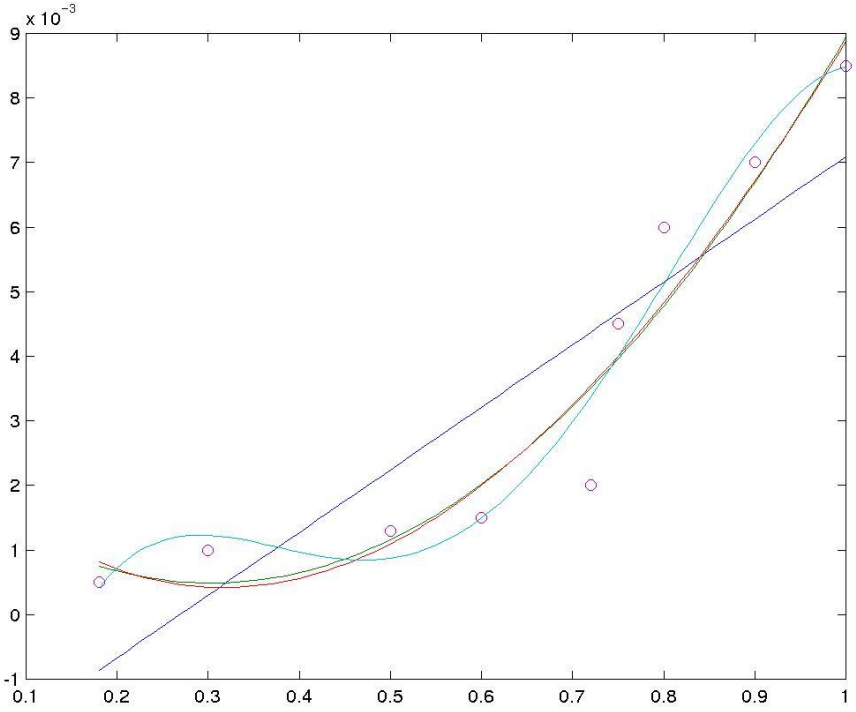
- Then matrix form of the system of normal equations is

$$\boldsymbol{A^T * A\ a = A^t y}$$

- To avoid that $A^T * A$ gives an ill-conditioned system, several methods of factorization are used.

Least square method

- Example of **linear regression** and **polynomial regression** with degree n=2, 3, 4.



Least square method

- A polynomial with a higher degree does not necessarily provide a better approximation.

- Another approach consider an **approximation function**:

$$f(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \cdots + a_k\varphi_k(x)$$

that uses functions φ_i and the matrix

$$A = \begin{pmatrix} \varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_k(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_k(x_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_k(x_n) \end{pmatrix}$$

Least square method

- As an example we can consider the trigonometric polynomial ***T(x)*** giving **the best fit** in the least square sense:

$$T(x) = a_0 + a_1 \sin(x) + a_2 \sin(2x) + \cdots + a_k \sin(kx)$$

In this case the matrix *A* is:

$$A = \begin{pmatrix} 1 & \sin(x_1) & \sin(2x_1) & \cdots & \sin(kx_1) \\ 1 & \sin(x_2) & \sin(2x_2) & \cdots & \sin(kx_2) \\ 1 & \sin(x_3) & \sin(2x_3) & \cdots & \sin(kx_3) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \sin(x_n) & \sin(2x_n) & \cdots & \sin(kx_n) \end{pmatrix}$$

Matlab

- The Matlab function **`polyfit`** gives both
  - the **interpolant polynomial**
  - the **least square sense approximation polynomial**

The syntax of **`polyfit`** is **`polyfit(x,y,n)`**

where

- **`x`** and **`y`** are the coordinate vectors and
- **`n`** is the degree of the polynomial

- If n = length(x)-1, then we obtain the **interpolant polynomial**
- If n < length(x)-1, then we obtain the **least square sense approximation polynomial**