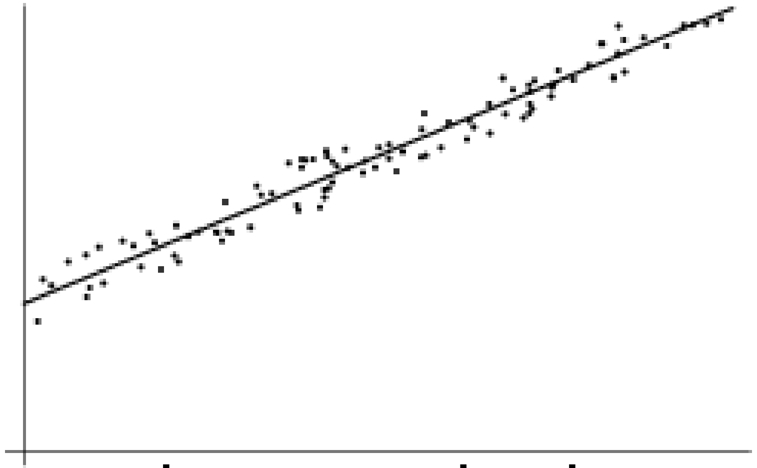
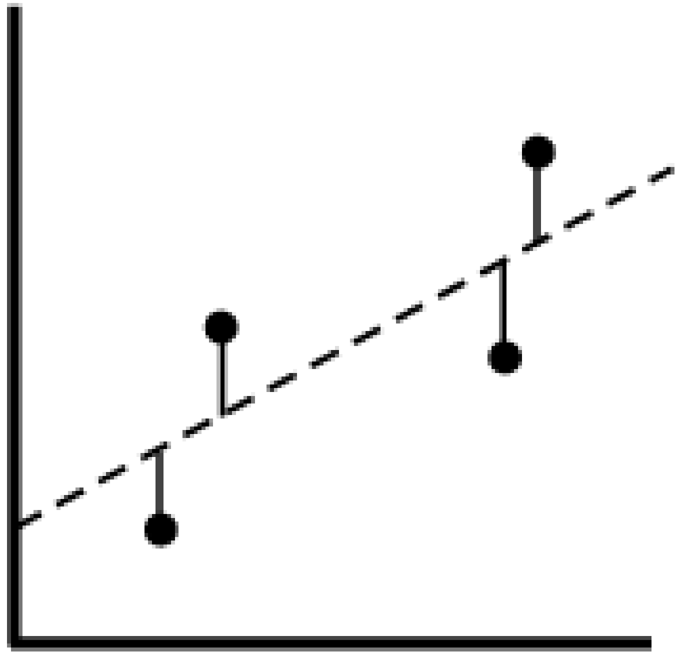
Data manipulation II: Least Squares Fitting

- http://mathworld.wolfram.com/LeastSquaresFitting.htm
- You have measured a set of data points, {xi,yi}, i = 1, 2, ..., N, and you know that they should approximately lie on a straight line of the form $y = a\,x + b$ if the yi's are plotted against xi's.



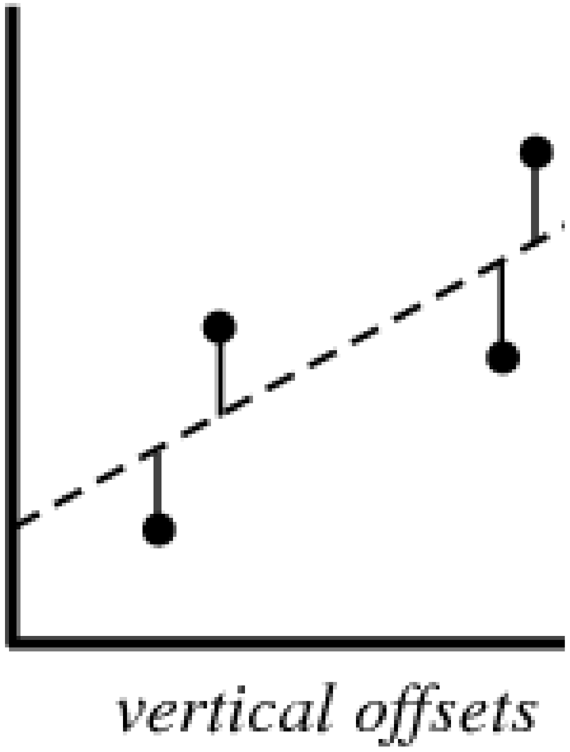
- We wish to know what are the best values for *a* and *b* that make the best fit for the data set.

Vertical offset



- Let *f*(xi,*a*,*b*) = b xi + a, in which we are looking for the best values for m and c.
- Vertical least squares fitting proceeds by minimizing the sum of the *squares* of the *vertical* deviations *R*2 of a set of *n* data points

Brute force minimisation



The values of m and c at which *R*2 is minimized is the best fit values.

You can either find these best fit values using “brute force method”, i.e. minimize *R*2 by scanning the barameter spaces of m and c (see brute_force_linearfit.nb), or be smarter by using a more intelligent approach.

Least Squared Minimisation

$$R^2(a,b) \equiv \sum_{i=1}^{n} [y_i - (a + b\,x_i)]^2$$

$$\frac{\partial(R^2)}{\partial a} = -2 \sum_{i=1}^{n} [y_i - (a + b\,x_i)] = 0$$

$$\frac{\partial(R^2)}{\partial b} = -2 \sum_{i=1}^{n} [y_i - (a + b\,x_i)]\,x_i = 0.$$

$$n\,a + b \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i\,y_i.$$

In matrix form

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i\, y_i \end{bmatrix},$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i\, y_i \end{bmatrix}. \qquad \text{Eq. (1)}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \begin{bmatrix} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i \\ n \sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \end{bmatrix},$$

$$a = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$= \frac{\bar{y}\left(\sum_{i=1}^{n} x_i^2\right) - \bar{x} \sum_{i=1}^{n} x_i\, y_i}{\sum_{i=1}^{n} x_i^2 - n\,\bar{x}^2}$$

Eq. (2)

$$b = \frac{n \sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Eq. (3)

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

$$\bar{x} = \frac{1}{N}\sum_{i}^{N} x_i\,, \bar{y} = \frac{1}{N}\sum_{i}^{N} y_i$$

standard errors

$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad SS_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$SS_{yy} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

$$s = \sqrt{\frac{ss_{yy} - b\, ss_{xy}}{n-2}} = \sqrt{\frac{ss_{yy} - \frac{ss_{xy}^2}{ss_{xx}}}{n-2}}$$

$$SE(a) = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{ss_{xx}}} \qquad SE(b) = \frac{s}{\sqrt{ss_{xx}}}.$$

Eq. (4,5)

Exercise

Write a Mathematica code to calculate a, b and their standard errors based on Eqs. (2,3,4,5).

Use the data file:

http://www2.fizik.usm.my/tlyoon/teaching/ZCE111/1415SEM2/notes/data_for_linear_fit.dat

Matrix Manipulation

- The best values for a and b in the fitting equation can also be obtained by solving the matrix equation, Eq. (1).

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i \, y_i \end{bmatrix}.$$

- Develop a Mathematica code to implement the matrix calculation.
- See least_sq_fit_matrix.nb.

Mathematica's built-in functions for data fitting

- See Math_built_in_linearfit.nb.
- Syntax: **Take[]**
- Syntax: **FindFit[ ]**, **LinearModelFit**,**Normal**
- "**BestFit**", "**ParameterTable**"
- These are Mathematica's built in functions to fit a set of data against a linear formula, such as y = a + b x, and at the same time automatically provide errors of the best fit parameters – very handy way to fit a set of data against any linear formula.

Treating real experimental data and manipulate them

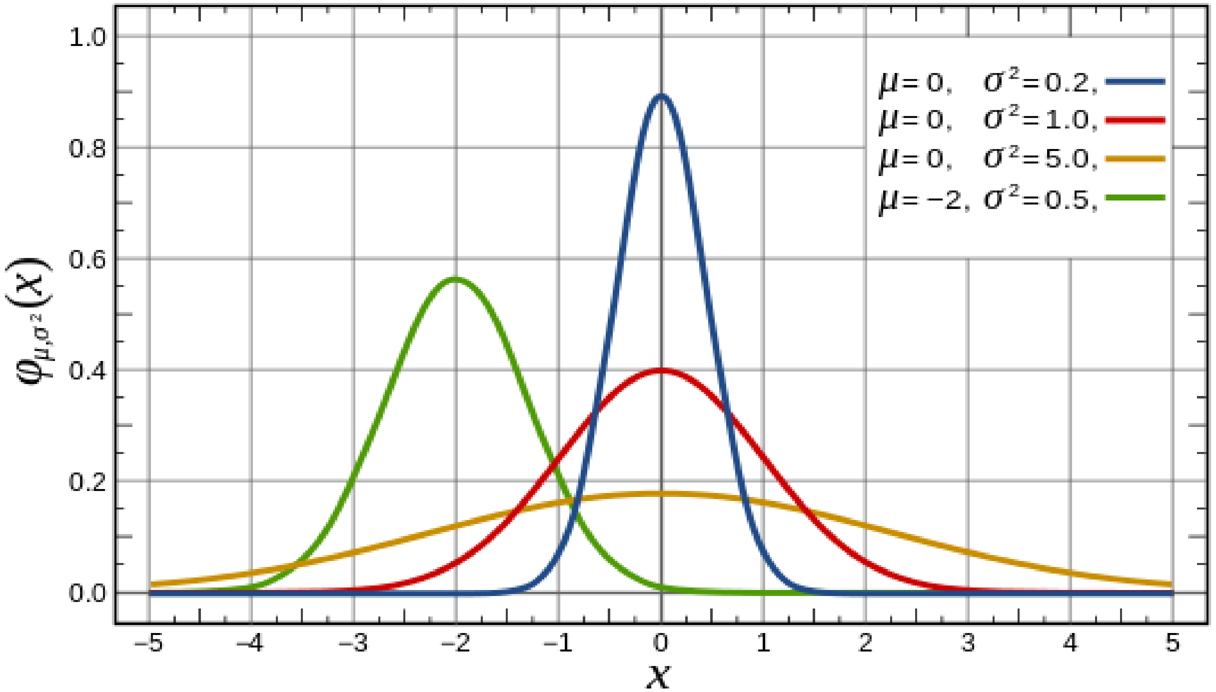
See the file varpendulum.nb

2GP1, VARIABLE_G_PENDULUM.PDF

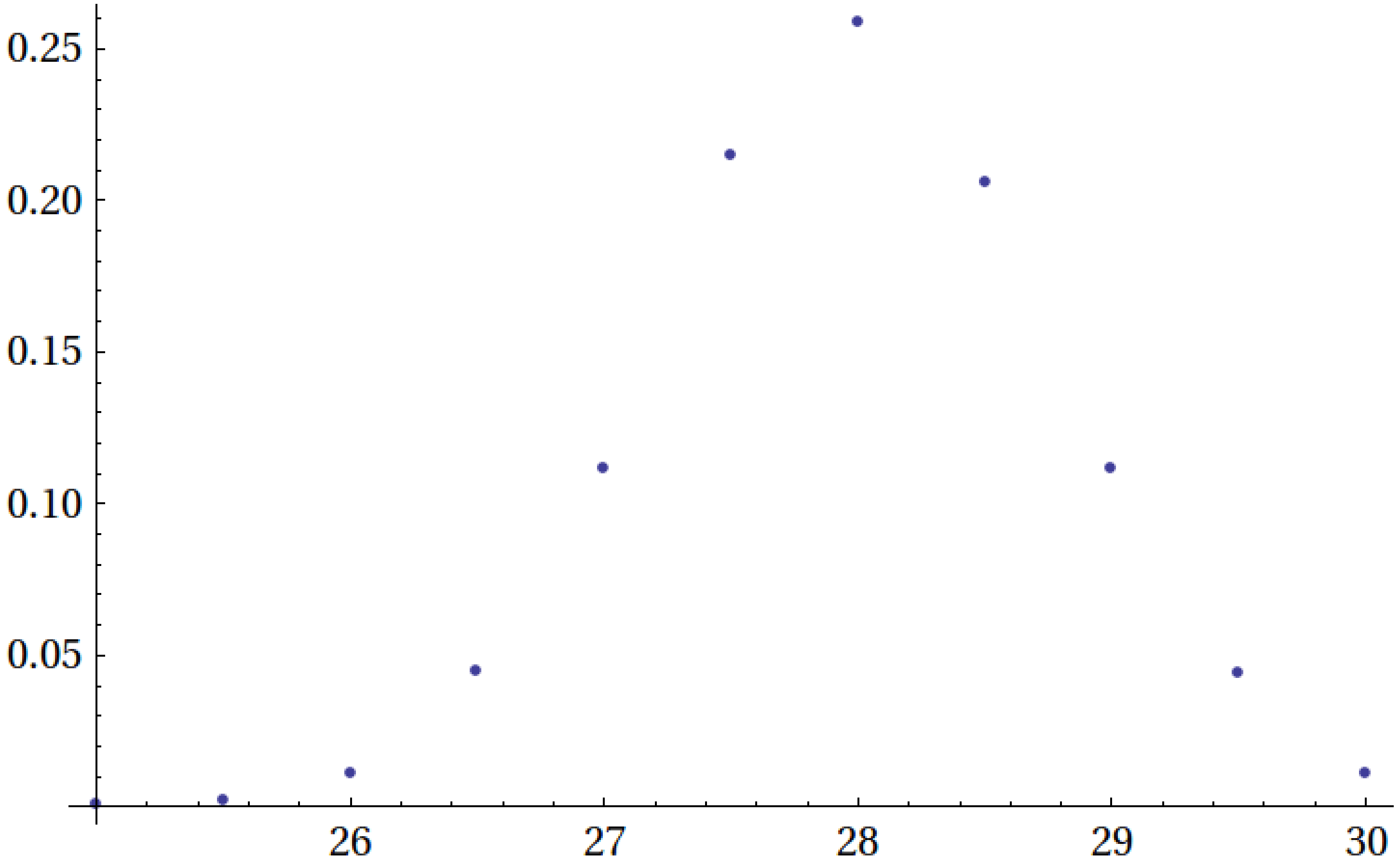
Gaussian function

- A Gaussian function has the form:It $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},$

- It is parametrised by two parameters, μ (average) and σ (width, or squared root of variance).



Download and **ListPlot** the data file, gaussian.dat.



Interpolation[]

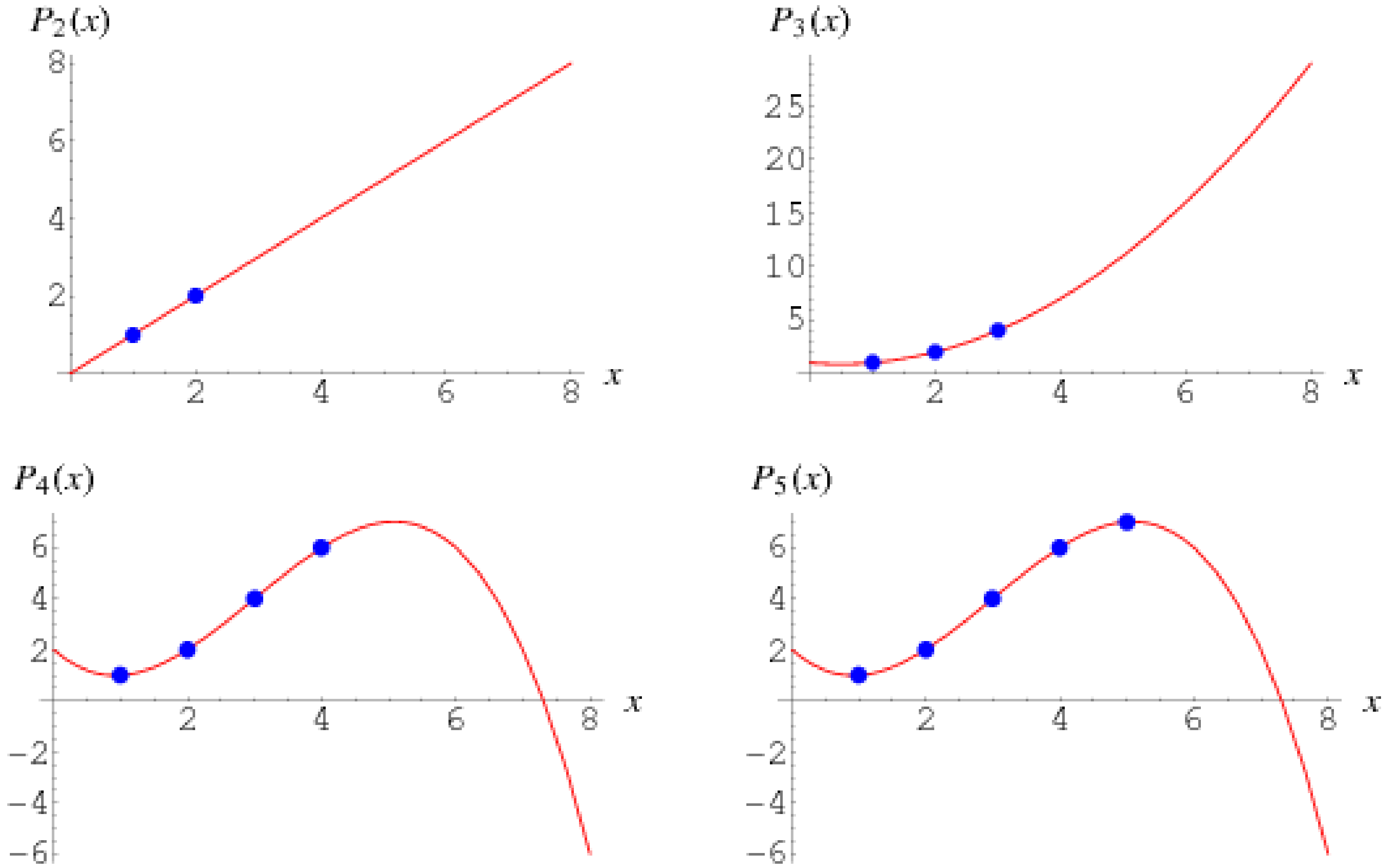
- These data points can be automatically linked up by a best curve using the Mathematica built-in function
- **Interpolation.**
- See interpolation_gaussian_data.nb

NonlinearModelFit

- Now, how would you ask Mathematica to find out the values of s and m that best fit the data point against a gaussian function?
- Use **NonlinearModelFit**
- See nonlinearfit_gaussian.nb for the use of built-in function to fit a set of data points onto a non-linear function, such as the gaussian distribution.

Lagrange Interpolating Polynomial

- Given a set of n data point, how do you perform an interpolation, i.e., generate a function that goes through all points (without using the built-in function Interpolation)?
- A simple way is to use Lagrange Interpolating Polynomial
- http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.



$P_2(x)$
8
6
4
2
2
4
6
8
x
$P_3(x)$
25
20
15
10
5
2
4
6
8
x
$P_4(x)$
6
4
2
−2
−4
−6
2
4
6
8
x
$P_5(x)$
6
4
2
−2
−4
−6
2
4
6
8
x

Explicit form of Lagrange interpolating polynomial

- The Lagrange interpolating polynomial is the polynomial P(x) of degree <=(n-1) that passes through the n points (x1,y1), (x2,y2), …, (xn,yn)

$$P(x) = \sum_{j=1}^{n} P_j(x), \qquad P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}.$$

Exercise: Write a Mathematica code to realise the Lagrange interpolating polynomial, using the sample data http://www2.fizik.usm.my/tlyoon/teaching/ZCE111/1415SEM2/nc

Syntax: **Product**

- See http://www2.fizik.usm.my/tlyoon/teaching/ZCE111/1415SEM2/nc
-

Polynomial Interpolation

- A more rigorous method to perform interpolation is by the way of Polynomial_interpolation. http://en.wikipedia.org/wiki/Polynomial_interpolation
- Given a set of n + 1 data points (x_i, y_i) where no two x_i are the same, one is looking for a polynomial *p* of degree at most *n* with the property
- $p(x_i) = y_i$, $i = 0, 1, 2, \ldots, n$.

Polynomial Interpolation

- Suppose that the interpolation polynomial is in the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0. \qquad (1)$$

- The statement that p interpolates the data points means that

$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}.$$

- If we substitute equation (1) in here, we get a system of linear equations in the coefficients a_k. The system in matrix-vector form reads

Vandermonde matrix

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

- For derivation of the Vandermonde matrix, see the lecture note by Dmitriy Leykekhman, University of Connecticut.

Exercise

- using the sample data using the sample data, monomial.dat
- write a code to construct the Vandermonde matrix
- Solve the matrix equation for the coefficients a_k.
- Then obtain the resultant interpolating polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0. \qquad (1)$$

- Overlap the interpolating polynomial on the raw data to see if they agree with each other.
- Compare your interpolating polynomial with that using Mathematica.