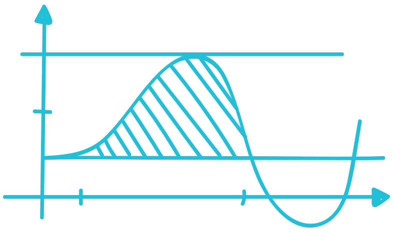
Numerical Methods



→Lec. 1: Curve Fitting - Linear Regression

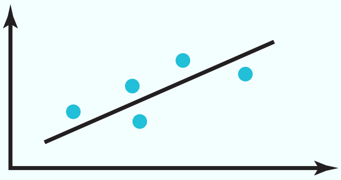
❋ **Numerical methods** are techniques by which mathematical problems are formulated so that they can be solved with **arithmetic** and **logical operations**.

❋ **Curve fitting** is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints.

Curve Fitting

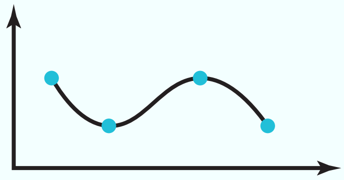
Regression

Deriving a single curve that represents the general trend of the data, where the data exhibit a significant degree of error.

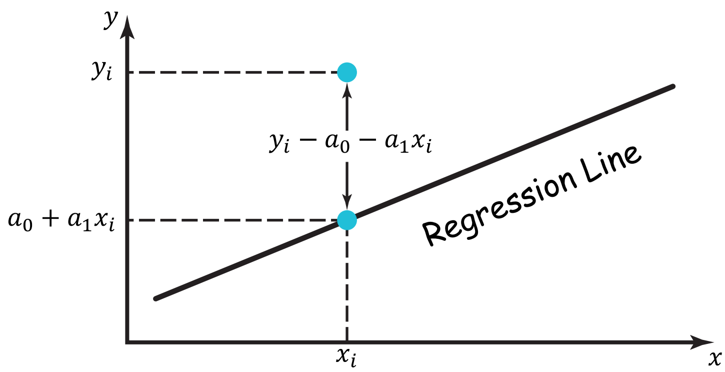


Interpolation

Fitting a curve or a series of curves that pass directly through each of the data points, where the data are known to be very precise.



1 Linear Regression



❋ **Linear regression** attempts to model the relationship between two variables by fitting a linear equation to observed data. The mathematical expression for a straight line is $y = a_0 + a_1 x$, where a_0 and a_1 are coefficients representing the intercept and the slope, respectively, and e is the error, or residual. A **residual** is the difference between an observed value and the fitted value provided by a model, which can be represented as $e = y_i - a_0 - a_1 x_i$.

The method of **least squares** is a parameter estimation method in **regression analysis** based on minimizing the sum of the squares of the **residuals**. **Linear least squares regression** has a number of advantages, including that it yields a unique line for a given set of n data points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, and that it removes the effect of the signs of the residual errors.

The sum of the squares of the residuals is given by:

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

✻ Least-Squares Fit of a Straight Line

To determine values for a_0 and a_1, such that the sum of square residuals is minimized, S_r is differentiated with respect to each unknown coefficient:

$$\frac{\partial S_r}{\partial a_0} = -2\sum(y_i - a_0 - a_1 x_i) \qquad \frac{\partial S_r}{\partial a_1} = -2\sum[(y_i - a_0 - a_1 x_i)x_i]$$

Setting these derivatives equal to zero will result in a minimum S_r. If this is done, the equations can be expressed as

$$0 = \sum y_i - \sum a_0 - \sum a_1 x_i \qquad 0 = \sum x_i y_i - \sum a_0 x_i - \sum a_1 x_i^2$$

We can express the equations as a set of two simultaneous linear equations with two unknowns (a_0 and a_1). These are called the **normal equations**.

$$n a_0 + \left(\sum x_i\right) a_1 = \sum y_i$$

$$\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 = \sum x_i y_i$$

normal equations

✻ Note:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \qquad a_0 = \bar{y} - a_1\bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively.

Quantification of Error of Linear Regression

Standard Error of the Estimate $$s_{y/x} = \sqrt{\frac{S_r}{n-2}}$$

Standard Deviation $$s_y = \sqrt{\frac{S_t}{n-1}}$$

Coefficient of Determination $$r^2 = \frac{S_t - S_r}{S_t}$$

Correlation Coefficient $$r = \sqrt{r^2}$$

$$S_r = \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^2 \qquad S_t = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

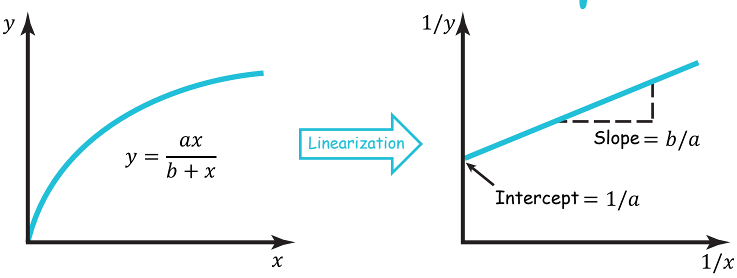
If $s_{y/x} < s_y$, we say "the method has merit"

✻ Linearization of Nonlinear Relationships

We can use linear regression to fit data to a **non-linear** function (curve) instead of a straight line by using **linearization** to convert the data to points that fit a linear model, then revert to the original model when we're done.

$$f(y) = a + bg(x) \implies Y = a_0 + a_1 X$$

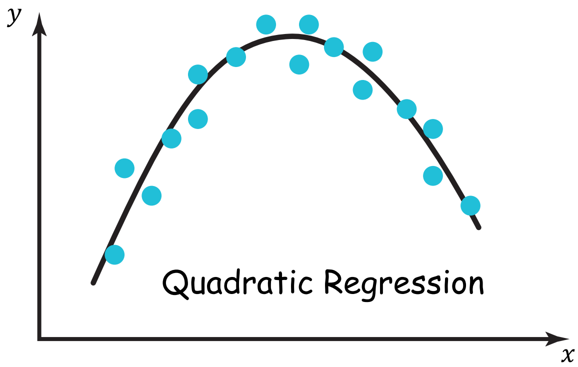
$$Y = f(y) \quad a = a_0 \quad b = a_1 \quad X = g(x)$$



Exponential model	Growth-rate model	Power model
$y = ae^{bx}$	$y = \frac{ax}{b+x}$	$y = ax^b$

→Lec. 2: Curve Fitting - Polynomial Regression

2 Polynomial Regression



✳ The **polynomial regression** model

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

gives us an inconsistent system of n linear equations which can be written in matrix form.

$$\begin{aligned} y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_m x_1^m \\ y_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_m x_2^m \\ &\vdots \\ y_n &= a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_m x_n^m \end{aligned} \implies \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

In matrix notation, the equation for a polynomial fit is given by $X\mathbf{a} = \mathbf{y}$. We can look for an $\mathbf{a}$ that minimizes the norm of the error, $\|\mathbf{y} - X\mathbf{a}\|$. This is an example of a **least squares problem**, the problem of minimizing a sum of squares. For this case the sum of the squares to be minimized is the sum of the squares of the residuals.

Minimize $$\|\mathbf{y} - X\mathbf{a}\|^2 = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 + \cdots - a_m x_i^m\right)^2 = S_r$$

From linear algebra, this problem can be solved by premultiplying by the **transpose** X^T, and the solution of the least squares problem comes down to solving the $(m+1) \times (m+1)$ linear system of equations $X^T X \hat{\mathbf{a}} = X^T \mathbf{y}$. These equations are called the **normal equations** of the least squares problem. This matrix equation can be inverted directly to yield the solution vector $\hat{\mathbf{a}} = (X^T X)^{-1} X^T \mathbf{y}$.

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} = \begin{bmatrix} n & \sum x & \sum x^2 & \cdots & \sum x^m \\ \sum x & \sum x^2 & \sum x^3 & \cdots & \sum x^{m+1} \\ \sum x^2 & \sum x^3 & \sum x^4 & \cdots & \sum x^{m+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x^m & \sum x^{m+1} & \sum x^{m+2} & \cdots & \sum x^{2m} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum xy \\ \vdots \\ \sum x^m y \end{bmatrix}$$

normal equations

$$\begin{bmatrix} n & \sum x & \sum x^2 & \cdots & \sum x^m \\ \sum x & \sum x^2 & \sum x^3 & \cdots & \sum x^{m+1} \\ \sum x^2 & \sum x^3 & \sum x^4 & \cdots & \sum x^{m+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x^m & \sum x^{m+1} & \sum x^{m+2} & \cdots & \sum x^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum xy \\ \vdots \\ \sum x^m y \end{bmatrix}$$

Linear Regression $m = 1$ Quadratic Regression $m = 2$

Standard Error of the Estimate

$$s_{y/x} = \sqrt{\frac{S_r}{n - (m+1)}}$$

→Lec. 3: Solving Nonlinear Equations

✻ In numerical analysis, a **root-finding algorithm** is an algorithm for finding **zeros**, also called "**roots**", of continuous functions. A zero of a function f, is a number x such that $f(x) = 0$. As, generally, the zeros of a function cannot be computed exactly nor expressed in closed form, root-finding algorithms provide approximations to zeros. Solving an equation $f(x) = g(x)$ is the same as finding the roots of the function $h(x) = f(x) - g(x)$.Thus root-finding algorithms allow solving any equation defined by continuous functions.

Root-Finding Algorithms

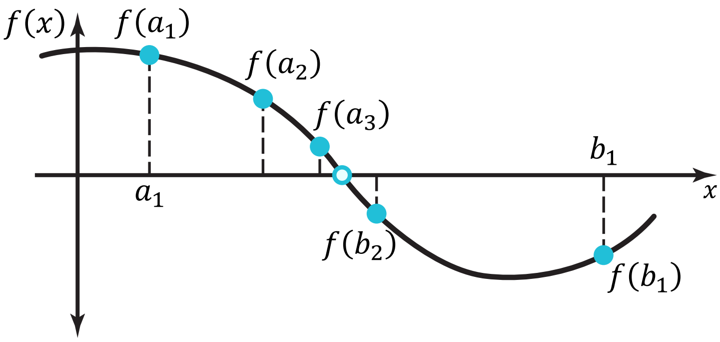
Bracketing Methods

These are based on two initial guesses that "bracket" the root—that is, are on either side of the root. For well-posed problems, the bracketing methods **always work** but **converge slowly**. They include the **bisection method** and the **false position method** "regula falsi".

Open Methods

These methods can involve one or more initial guesses, but there is no need for them to bracket the root. They do not always work (i.e., they **can diverge**), but when they do they usually **converge quicker**. They include the **Newton-Raphson method** and the **secant method**.

1 Bisection Method



✻ The input for the method is a continuous function f, an interval $[a, b]$, and the function values $f(a)$ and $f(b)$. The function values are of opposite signs. In this case a and b are said to bracket a root since, by the **intermediate value theorem**, the continuous function f must have at least one root in the interval (a, b). The method consists of repeatedly bisecting the interval defined by these values and then selecting the subinterval in which the function changes sign, and therefore must contain a root.

✻ Iteration tasks:

① Calculate x, the midpoint of the interval.

$$x = \frac{a + b}{2}$$

② Calculate the function value at the midpoint, $f(x)$.

③ If convergence is satisfactory return x and stop iterating.

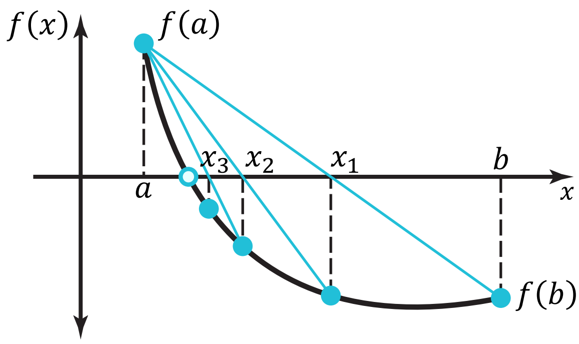
④ If, $f(x)f(a) > 0$ then the new interval is $[x, b]$, otherwise it's $[a, x]$.

Maximum Error Bound

$$\varepsilon = \frac{b - a}{2^n}$$

number of iterations

2 False Position Method



✻ The regula falsi (also called the linear interpolation method) method calculates the new solution estimate as the x-intercept of the line segment joining the endpoints of the function on the current bracketing interval. Essentially, the root is being approximated by replacing the actual function by a line segment on the bracketing interval and then using the false-position formula on that line segment.

✻ Iteration tasks:

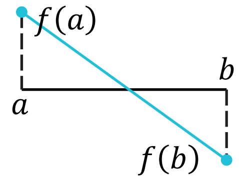
① Calculate x from the flase-position formula.

$$x = \frac{bf(a) - af(b)}{f(a) - f(b)}$$

② Calculate the function value at the intercept, $f(x)$.

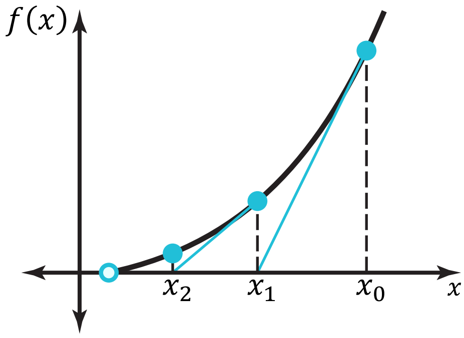
③ If convergence is satisfactory return x and stop iterating.

④ If, $f(x)f(a) > 0$ then the new interval is $[x, b]$, otherwise it's $[a, x]$.



$$\frac{f(b) - 0}{b - x} = \frac{0 - f(a)}{x - a}$$

3 Newton-Raphson Method



✻ Newton's method, also known as the Newton-Raphson method, named after Isaac Newton and Joseph Raphson, starts with an initial guess, then approximates the function by its tangent line, and finally computes the x-intercept of this tangent line. This x-intercept will typically be a better approximation to the original function's root than the first guess, and the method can be iterated.

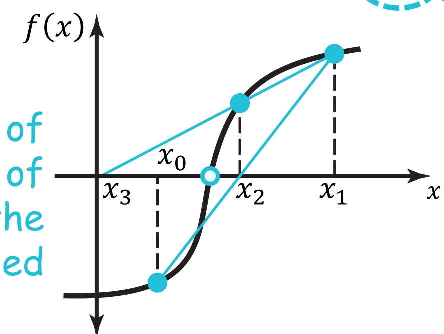
$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

Rearrange

Newton-Raphson Formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

4 Secant Method



✻ The secant method can be thought of as a finite-difference approximation of Newton's method by approximating the derivative by a backward finite divided difference.

$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

$$x_{i+1} = x_i - \frac{(x_{i-1} - x_i)f(x_i)}{f(x_{i-1}) - f(x_i)}$$

✻ Note:

$$\text{absolute error} = |\text{true value} - \text{approximation}|$$

$$\text{relative error} = \left|\frac{\text{true value} - \text{approximation}}{\text{true value}}\right|$$

$$\text{percent error} = 100\% \times \text{relative error}$$

→Lec. 4: Solving Linear Systems of Equations

1 Jacobi Method

❋ In numerical linear algebra, the Jacobi method (a.k.a. the Jacobi iteration method) is an iterative algorithm for determining the solutions of a strictly diagonally dominant system of linear equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges. The method is named after Carl Gustav Jacob Jacobi.

First Iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second Iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \longrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{12} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$A \quad \mathbf{x} = \mathbf{b}$$

2 Gauss-Seidel Method

❋ Note:
Often, the initial values are $x_i^{(0)} = 0$

❋ In numerical linear algebra, the Gauss-Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a system of linear equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either strictly diagonally dominant, or symmetric and positive definite.

First Iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second Iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

diagonally dominant matrix

❋ A square matrix A is diagonally dominant if:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \ \forall i$$

where a_{ij} denotes the entry in the ith row and jth column.
If a strict inequality ($>$) is used, this is called strict diagonal dominance.

Lec. 5: Curve Fitting - Interpolation

* The **polynomial interpolation** problem is to find a polynomial $p(x)$ which satisfies $p(x_0) = y_0, \dots, p(x_m) = y_m$ for given data points $(x_0, y_0), \dots, (x_m, y_m)$.

1 Vandermonde Matrix

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

* This problem can be reformulated in terms of linear algebra by means of the **Vandermonde matrix** V, as follows. V computes the values of $p(x)$ at the points $x = x_0, x_1, \dots, x_m$ via a matrix multiplication $V\mathbf{a} = \mathbf{y}$, where $\mathbf{a} = (a_0, \dots, a_n)$ is the vector of coefficients and $\mathbf{y} = (y_0, \dots, y_m) = \big(p(x_0), \dots, p(x_m)\big)$ is the vector of values (both written as column vectors). The Vandermonde matrix is named after **Alexandre-Théophile Vandermonde**.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 & -x_1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 y_1 - x_1 y_2 \\ y_2 - y_1 \end{bmatrix}$$

If $n = m$ and $x_0, \dots, x_n$ are distinct, then V is a square matrix with non-zero determinant, i.e. an invertible matrix. Thus, given V and $\mathbf{y}$, one can find the required $p(x)$ by solving for its coefficients $\mathbf{a} = V^{-1}\mathbf{y}$.

2 Lagrange Polynomial

* Given a set of $n+1$ nodes $\{x_0, x_1, \dots, x_n\}$, which must all be distinct, the **Lagrange basis** for polynomials of degree $\leq n$ for those nodes is the set of polynomials $\{\ell_0(x), \ell_1(x), \dots, \ell_n(x)\}$ each of degree n which take values $\ell_j(x_m) = 0$ if $m \neq j$ and $\ell_j(x_j) = 1$.

$$\ell_j(x) = \prod_{\substack{0 \le m \le n \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$$

The **Lagrange interpolating polynomial** for those nodes through the corresponding values $\{y_0, y_1, \dots, y_n\}$ is the linear combination:

$$L(x) = \sum_{j=0}^{n} y_j \, \ell_j(x) \qquad \sum_{j=0}^{n} \ell_j(x) = 1$$