1. *Univariate Interpolation*: find function/curve $y = g(x)$, the *interpolant*, through the n points (x_i, y_i), $x_1 < x_2 < \ldots < x_n$; i.e.

$$g(x_i) = y_i\,, \qquad i = 1, 2, \ldots, n\,.$$

2. *Osculatory* or *Hermite Interpolation*: find function/curve $y = g(x)$ through the n points (x_i, y_i), whose derivative $y = g'(x)$ passes through the n points (x_i, y_i');i.e.

$$g(x_i) = y_i\,, \qquad g'(x_i) = y_i'\,, \qquad i = 1, 2, \ldots, n\,.$$

The data y_i and y_i' often arise from interpolating a given function $f(x)$ by a "simpler" function $g(x)$,

$$y_i = f(x_i) \equiv f_i\,, \qquad y_i' = f'(x_i) \equiv f_i'\,.$$

3. Generalisations: higher derivatives; *multivariate interpolation*.

Curve-fitting: fit one smooth curve, in some "best" sense (e.g. least-squares), to a set of data points, usually with errors. The curve cannot pass through all the data points and the interpolation equations have no solution in the usual sense.

4. The interpolant is usually chosen as a linear combination of known *basis* functions, b_j,

$$g(x) = \sum_{j=1}^{n} \alpha_j b_j(x)\,.$$

More abstractly, $g(x)$ is chosen from a vector space spanned by the basis $\{b_1, b_2, \ldots, b_n\}$. The interpolant $g(x)$ can be expressed as a linear combination of any other basis of this vector space. Some bases have more desirable properties than others.

To find the α_j solve,

$$\sum_{j=1}^{n} \alpha_j b_j(x_i) = y_i\,, \qquad i = 1, 2, \ldots, n\,; \qquad \text{or} \qquad \mathbf{B}\boldsymbol{\alpha} = \mathbf{y}\,,$$

where

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} b_1(x_1) & b_2(x_1) & \ldots & b_n(x_1) \\ b_1(x_2) & b_2(x_2) & \ldots & b_n(x_2) \\ \vdots & \vdots & \ldots & \vdots \\ b_1(x_n) & b_2(x_n) & \ldots & b_n(x_n) \end{pmatrix}.$$

Can solve using Gaussian elimination with partial pivoting directly, even for ill-conditioned systems, since the method gives small residuals. However, more efficient versions of the methods can be obtained by exploiting the special structure of the interpolation equations.

6.1 Polynomial Interpolation

1. If $b_i(x) = x^i$ $(i = 0, 1, 2, \ldots, n)$, then g is a polynomial in x of degree n,

$$f(x) = p_n(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_n x^n\,.$$

$$\mathbf{B} = \begin{pmatrix} 1 & x_0 & \ldots & x_0^n \\ 1 & x_1 & \ldots & x_1^n \\ \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & \ldots & x_n^n \end{pmatrix}; \qquad \det \mathbf{B} = \prod_{0 \le i < j \le n} (x_j - x_i)\,.$$

2. *There is a unique polynomial of degree* $\leq$ *n, which passes through* $n+1$ *points,* (x_i, y_i), $i = 0, 1, 2 \ldots, n$, *with distinct abscissae.* If the x_i are distinct, then $\det \mathbf{B} \neq 0$, $\mathbf{B}$ is non-singular and $\mathbf{B}\boldsymbol{\alpha} = \mathbf{y}$ has a unique solution $\boldsymbol{\alpha}$. Whence p_n exists and is unique.

3. Polynomial can be easily evaluated, using only $+$, $-$, $\times$.

The Weierstrass Approximation Theorem Let $f \in C[a, b]$. Given $\epsilon > 0$ there exists a polynomial $p_n(x)$ of sufficiently high degree n such that

$$|f(x) - p_n(x)| \leq \epsilon\,, \qquad a \leq x \leq b\,.$$

6.1.1 Lagrangian Interpolation

1. The interpolating vector space consists of all polynomials of degree $\leq n$.
   p_n can be expressed in terms of the other bases of this space:

   power basis — $\{1, x, x^2, \ldots, x^n\}$

   Taylor basis — $\{1, x-c, (x-c)^2/2!, \ldots, (x-c)^n/n!\}$

   Lagrangian basis — $\{\ell_0(x), \ell_1(x), \ldots, \ell_n(x)\}$ where

$$\ell_j(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}\,.$$

2. The Lagrangian basis polynomials have the property

$$\ell_j(x_i) = \begin{cases} 1\,, i = j\,; \\ 0\,, i \neq j\,. \end{cases}$$

   Thus $\mathbf{B} = \mathbf{I}$, $\boldsymbol{\alpha} = \mathbf{y}$ and

$$p_n(x) = y_0\ell_0(x) + y_1\ell_1(x) + \ldots + y_n\ell_n(x)$$

Lagrange's form of the interpolating polynomial or *lagrangian interpolation*.
$p_n(x)$ is unique, although its form is not.

$$
\begin{aligned}
p_1(x) &= \left(\frac{x-x_1}{x_0-x_1}\right) y_0 + \left(\frac{x-x_0}{x_1-x_0}\right) y_1 \\
p_2(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} y_2 \,.
\end{aligned}
$$

Example 6.1 Given the data in the Table find $f(0.72)$ by

(a) linear interpolation;

(b) quadratic interpolation, using Lagrange's form of the interpolating polynomial;

(c) cubic interpolation, using Lagrange's form of the interpolating polynomial;

x	0.5	0.6	0.7	0.8	0.9	1.0
y	0.49311	0.58812	0.68122	0.77209	0.86047	0.94608

Table 6.1: Interpolation data for Example 6.1.

Solution

(a)

$$
p_1(0.72) = \frac{(.72-.8)}{(.7-.8)}(.68122) + \frac{(.72-.7)}{(.8-.7)}(.77209) = 0.69939\,.
$$

(b)

$$
p_2(0.72) = \frac{(.72-.7)(.72-.8)}{(.6-.7)(.6-.8)}(.58812) + \frac{(.72-.6)(.72-.8)}{(.7-.6)(.7-.8)}(.68122)
$$

$$
\begin{aligned}
& + \frac{(.72-.6)(.72-.7)}{(.8-.6)(.8-.7)}(.77209) \\
= \quad & -0.04705 + 0.65397 + 0.09265 \\
= \quad & 0.69957\,.
\end{aligned}
$$

(c) Complicated formula; use programming: $p_3(0.72) = 0.69958$.

3. *Truncation Error*: Let $y_i = f(x_i)$ $(i = 0, 1, 2, \ldots, n)$; $f \in C^{n+1}[a, b]$; $x, x_0, x_1, x_2, \ldots, x_n \in [a, b]$. For any such f (not unique),

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\eta)}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n) = e_n(x)\,,$$

where $\eta \in [\min(x, x_0), \max(x, x_n)]$.

The polynomial $\ell(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$ and the truncation error $e_n(x)$ vanish if $x = x_i$. When x lies among the x_i, $\ell(x)$ oscillates but these oscillations generally have smaller amplitudes when x lies near the centre of the interval $[x_0, x_n]$.

To minimise the truncation error choose the data points x_i to be distributed as equally as possible about x.

High-degree polynomial interpolants usually exhibit large oscillations between the data points, giving poor interpolation.

4. $p_n(x)$ need not converge to $f(x)$ as $n \to \infty$.

E.g. if Runge's function,

$$f(x) = \frac{1}{1 + 25x^2}\,, \quad |x| < 1\,,$$

is interpolated by $n + 1$ equally-spaced points, then $p_n(x) \not\to f(x)$ as $n \to \infty$ for $0.726\ldots \leq |x| < 1$.

6.1.2 Newton-Gregory Interpolation (A)

A disadvantage of the Lagrangian basis is that the degree of the interpolating polynomial has to be chosen at the outset; a change in the order of the interpolating polynomial requires the whole calculation to be redone. This can be overcome by the Newton-Gregory basis

$$\{1, (x-x_0), (x-x_0)(x-x_1), (x-x_0)(x-x_1)(x-x_2), \ldots, (x-x_0)(x-x_1)(x-x_2)\ldots(x-x_{n-1})\}\,.$$

$$\begin{aligned} p_n(x) &= f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) + \ldots \\ &\quad + f[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})\,, \end{aligned}$$

$$f[x_0,x_1,\ldots,x_k] = \frac{f[x_1,x_2,\ldots,x_k] - f[x_0,x_1,\ldots,x_{k-1}]}{x_k - x_0}\,.$$

6.2 Piecewise Polynomial Interpolation

1. High degree polynomial interpolants through the n points (x_i, y_i) may oscillate wildly. It is often preferable to interpolate between consecutive *knots* x_i, x_{i+1} by low degree polynomials. The interpolant $g(x)$ is then a *piecewise polynomial*.

   Piecewise linear: continuous but derivative discontinuous (compound trapezoidal rule).

   Piecewise quadratic: continuous and derivative continuous, but poor error properties.

2. *Piecewise cubic* interpolant $C(x)$: continuous and derivative continuous.

   $C(x)$ is a cubic polynomial with 4 disposable parameters on each of the $n-1$ subintervals (x_i, x_{i+1}) $(i = 1, 2, \ldots, n-1)$. Thus $C(x)$ has $4(n-1)$ disposable parameters.

   $C(x)$ passes through two points (x_i, y_i) and (x_{i+1}, y_{i+1}) on each subinterval $[x_i, x_{i+1}]$, imposing $2(n-1)$ conditions on $C(x)$.

   Continuity of $C'(x)$ at each of the $n-2$ interior knots x_{i+1}, $i = 2, 3, \ldots, n-1$ imposes a further $n-2$ conditions on $C(x)$.

   $4(n-1) - 2(n-1) - (n-2) = n$ disposable parameters remain:

   Hermite piecewise cubic interpolation: $C'(x)$ is given or approximated at each of the n knots x_i, $i = 1, 2, \ldots, n$, imposing n further conditions, $C'(x_i) = y_i'$.

   Cubic spline interpolation: Continuity of $C''(x)$ at each interior knot imposes a further $n-2$ conditions, leaving two disposable parameters. Need 2 more conditions to completely determine $C(x)$; e.g. "not-a-knot" conditions — $C'''(x)$ continuous at x_2, x_{n-1}.

3. The Hermite piecewise cubic interpolant may be easier to determine than the cubic spline interpolant. Evaluation requires about the same amount of work.

Splines may look smoother, but they are underdetermined having two disposable parameters leftover and they may be non-monotonic, even when interpolating monotonic data.

Figure 6.1: A plot of the polynomial $(x-x_0)(x-x_1)(x-x_2)(x-x_3)(x-x_4)$, where x_0, x_1, x_2, x_3, x_4 are spaced a distance h apart. The horizontal and vertical scales are h and $10h$, respectively. The polynomial decreases rapidly with x for $x < x_0$ and increases rapidly with x for $x > x_4$.