

UNIT 2

Interpolation and Approximation

(8 Hours) — Numerical Methods (CACS252)

2.1 Introduction

Interpolation is the process of estimating the value of a function at a point x that lies between known data points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$. Estimating a value outside the range of the data is called **extrapolation** and is less reliable. **Approximation** (curve fitting) instead finds a simple function that follows the general trend of noisy data without passing through every point.

Existence and uniqueness. Through $n + 1$ distinct points there is exactly one polynomial of degree at most n that passes through them all. Lagrange's and Newton's formulas are different ways of writing this *same* polynomial.

Method	Data spacing	Idea
Lagrange polynomial	Equal or unequal	Weighted sum of basis polynomials $L_i(x)$
Newton forward / backward difference	Equal only	Uses finite differences Δ, ∇ and a variable p
Newton divided difference	Equal or unequal	Uses divided differences; easy to add new points
Cubic spline	Equal or unequal	Piecewise cubics joined smoothly
Least squares	Any (noisy data)	Minimises the sum of squared residuals

2.2 Errors in Polynomial Interpolation

Let $P_n(x)$ be the polynomial of degree $\leq n$ interpolating f at $x_0, \dots, x_n$. If f has $n + 1$ continuous derivatives, then for every x in the interval there is a point ξ in the interval such that

$$E(x) = f(x) - P_n(x) = [f^{(n+1)}(\xi) / (n + 1)!] \cdot (x - x_0)(x - x_1) \cdots (x - x_n)$$

Hence $|E(x)| \leq [M / (n + 1)!] \cdot |(x - x_0)(x - x_1) \cdots (x - x_n)|$, where $M = \max |f^{(n+1)}(t)|$.

Observations:

- The error is exactly zero at the nodes $x_0, \dots, x_n$.
- The error is smallest near the middle of the data and larger near the ends; extrapolation makes the product term large.

- Higher degree does not always mean smaller error (Runge's phenomenon with equally spaced nodes); for many points a spline is usually better than a single high-degree polynomial.

Example 2.1 — Error bound

Interpolate $\sin x$ at $x = 45^\circ, 50^\circ, 55^\circ, 60^\circ$ ($n = 3$) and estimate the error at $x = 52^\circ$.

Solution

For $f = \sin x$, $|f^{(4)}(t)| = |\sin t| \leq 1$, so $M = 1$. Working in degrees, each degree equals $\pi/180$ radians, so the factor $(\pi/180)^4 = 9.2807 \times 10^{-8}$ converts the product term.

$$|(52-45)(52-50)(52-55)(52-60)| = |7 \cdot 2 \cdot (-3) \cdot (-8)| = 336$$

$$|E(52)| \leq (1/4!) \cdot 336 \cdot (\pi/180)^4 = 1.3 \times 10^{-6}$$

Check: $\sin 52^\circ = 0.788011$, and the interpolating polynomial gives 0.788003; the actual error is 7.6×10^{-6} , which is dominated by the 4-decimal rounding of the table values.

2.3 Lagrange's Interpolating Polynomial

For $n + 1$ points the Lagrange polynomial is

$$P_n(x) = \sum y_i \cdot L_i(x), \quad L_i(x) = \prod (x - x_j) / (x_i - x_j) \quad (\text{product over } j = 0, \dots, n; j \neq i)$$

Each basis polynomial $L_i(x)$ equals 1 at $x = x_i$ and 0 at every other node, so $P_n(x_i) = y_i$. For example, with three points (linear-in-each-factor form):

$$P_2(x) = y_0 (x - x_1)(x - x_2) / [(x_0 - x_1)(x_0 - x_2)] + y_1 (x - x_0)(x - x_2) / [(x_1 - x_0)(x_1 - x_2)] + y_2 (x - x_0)(x - x_1) / [(x_2 - x_0)(x_2 - x_1)]$$

Example 2.2

Given $f(5) = 12$, $f(6) = 13$, $f(9) = 14$, $f(11) = 16$, find $f(10)$ using Lagrange's formula.

Solution

The nodes are unequally spaced, so Lagrange's formula is appropriate. At $x = 10$:

$$L_0(10) = -4 / -24 = 0.166667$$

$$L_1(10) = -5 / 15 = -0.333333$$

$$L_2(10) = -20 / -24 = 0.833333$$

$$L_3(10) = 20 / 60 = 0.333333$$

$$f(10) \approx (12)(0.166667) + (13)(-0.333333) + (14)(0.833333) + (16)(0.333333)$$

$$f(10) \approx 14.6667$$

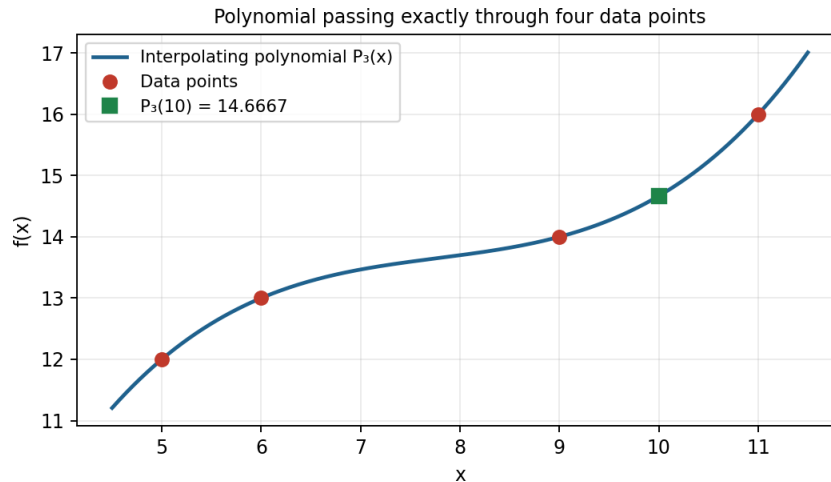


Figure 2.1 — The interpolating polynomial through (5,12), (6,13), (9,14), (11,16), evaluated at $x = 10$.

Note: Sum of the basis polynomials is always 1 ($\sum L_i(x) = 1$). Use this as a quick arithmetic check: here $0.166667 + -0.333333 + 0.833333 + 0.333333 = 1.000000$.

2.4 Newton's Interpolation

2.4.1 Finite Differences (equally spaced data)

For equally spaced nodes $x_i = x_0 + i \cdot h$, define the **forward difference** $\Delta y_i = y_{i+1} - y_i$ and higher orders $\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$. The **backward difference** is $\nabla y_i = y_i - y_{i-1}$. Note that $\nabla y_i = \Delta y_{i-1}$, so both use the same numbers in a difference table.

Newton's forward formula (best near the *beginning* of the table), with $p = (x - x_0)/h$:

$$y(x) = y_0 + p \cdot \Delta y_0 + [p(p-1)/2!] \cdot \Delta^2 y_0 + [p(p-1)(p-2)/3!] \cdot \Delta^3 y_0 + \dots$$

Newton's backward formula (best near the *end* of the table), with $p = (x - x_n)/h$:

$$y(x) = y_n + p \cdot \nabla y_n + [p(p+1)/2!] \cdot \nabla^2 y_n + [p(p+1)(p+2)/3!] \cdot \nabla^3 y_n + \dots$$

Example 2.3

The table gives $\sin x$. Find $\sin 52^\circ$ with the forward formula and $\sin 58^\circ$ with the backward formula.

x : $45^\circ, 50^\circ, 55^\circ, 60^\circ$ $\sin x$: 0.7071, 0.7660, 0.8192, 0.8660

x	$y = \sin x$	Δy	$\Delta^2 y$	$\Delta^3 y$
45	0.7071	0.0589	-0.0057	-0.0007
50	0.7660	0.0532	-0.0064	

x	y = sin x	Δy	$\Delta^2 y$	$\Delta^3 y$
55	0.8192	0.0468		
60	0.8660			

Solution

(a) $\sin 52^\circ$: $h = 5$, $p = (52 - 45)/5 = 1.4$.

$$y = 0.7071 + (1.4)(0.0589) + [(1.4)(0.4)/2](-0.0057) + [(1.4)(0.4)(-0.6)/6](-0.0007)$$

$$y \approx 0.7880 \text{ (true value } \sin 52^\circ = 0.7880)$$

(b) $\sin 58^\circ$: $p = (58 - 60)/5 = -0.4$.

$$y = 0.8660 + (-0.4)(0.0468) + [(-0.4)(0.6)/2](-0.0064) + [(-0.4)(0.6)(1.6)/6](-0.0007)$$

$$y \approx 0.8481 \text{ (true value } \sin 58^\circ = 0.8480)$$

2.4.2 Divided Differences (unequally spaced data)

Divided differences do not require equal spacing. Define $f[x_i] = y_i$, $f[x_i, x_{i+1}] = (y_{i+1} - y_i)/(x_{i+1} - x_i)$, and in general

$$f[x_i, \dots, x_{i+k}] = (f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]) / (x_{i+k} - x_i)$$

Newton's divided-difference formula:

$$P_n(x) = f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \dots + (x - x_0) \dots (x - x_{n-1}) f[x_0, \dots, x_n]$$

Divided differences are symmetric in their arguments, and adding a new data point only adds one new term — the earlier terms need not be recomputed.

Example 2.4

Use the same data as Example 2.2, (5, 12), (6, 13), (9, 14), (11, 16), and find $f(10)$ by Newton's divided-difference formula.

x	f[x]	1st divided diff.	2nd divided diff.	3rd divided diff.
5	12	1	-1/6	1/20
6	13	1/3	2/15	
9	14	1		
11	16			

Solution

$$P_3(x) = 12 + (x - 5)(1) + (x - 5)(x - 6)(-1/6) + (x - 5)(x - 6)(x - 9)(1/20)$$

$$P_3(10) = 12 + (5)(1) + (5)(4)(-1/6) + (5)(4)(1)(1/20) = 14.6667$$

This agrees with the Lagrange result in Example 2.2 (the interpolating polynomial is unique).

2.5 Cubic Spline Interpolation

A single polynomial through many points oscillates. A **cubic spline** $S(x)$ instead uses a different cubic $S_i(x)$ on each sub-interval $[x_i, x_{i+1}]$ and joins them smoothly. Conditions:

- $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ (the spline passes through every data point);
- S' is continuous at each interior node ($S_{i-1}'(x_i) = S_i'(x_i)$);
- S'' is continuous at each interior node ($S_{i-1}''(x_i) = S_i''(x_i)$);
- two end conditions; for the **natural spline**, $S''(x_0) = S''(x_n) = 0$.

Let $M_i = S''(x_i)$ and $h_i = x_{i+1} - x_i$. On $[x_i, x_{i+1}]$ the spline is

$$S_i(x) = M_i (x_{i+1} - x)^3 / (6h_i) + M_{i+1} (x - x_i)^3 / (6h_i) + (y_i - M_i h_i^2/6)(x_{i+1} - x)/h_i + (y_{i+1} - M_{i+1} h_i^2/6)(x - x_i)/h_i$$

Continuity of S' gives, for $i = 1, \dots, n - 1$, the tridiagonal system

$$h_{i-1} M_{i-1} + 2(h_{i-1} + h_i) M_i + h_i M_{i+1} = 6 \left[(y_{i+1} - y_i)/h_i - (y_i - y_{i-1})/h_{i-1} \right]$$

For equal spacing $h = 1$ this is $M_{i-1} + 4M_i + M_{i+1} = 6(y_{i+1} - 2y_i + y_{i-1})$.

Example 2.5

Fit a natural cubic spline through (1, 1), (2, 4), (3, 3), (4, 6) and estimate $y(2.5)$.

Solution

$h = 1$, natural ends: $M_0 = M_3 = 0$.

$$i = 1: M_0 + 4M_1 + M_2 = 6(y_2 - 2y_1 + y_0) = 6(3 - 8 + 1) = -24 \Rightarrow 4M_1 + M_2 = -24$$

$$i = 2: M_1 + 4M_2 + M_3 = 6(y_3 - 2y_2 + y_1) = 6(6 - 8 + 4) = 24 \Rightarrow M_1 + 4M_2 = 24$$

Solving: $M_1 = -8$, $M_2 = 8$.

$$\text{On } [2, 3]: S_1(x) = M_1(3 - x)^3/6 + M_2(x - 2)^3/6 + (y_1 - M_1/6)(3 - x) + (y_2 - M_2/6)(x - 2)$$

$$S(2.5) = (-8)(0.5)^3/6 + (8)(0.5)^3/6 + (4 + 8/6)(0.5) + (3 - 8/6)(0.5) = 0 + 2.6667 + 0.8333 = 3.5000$$

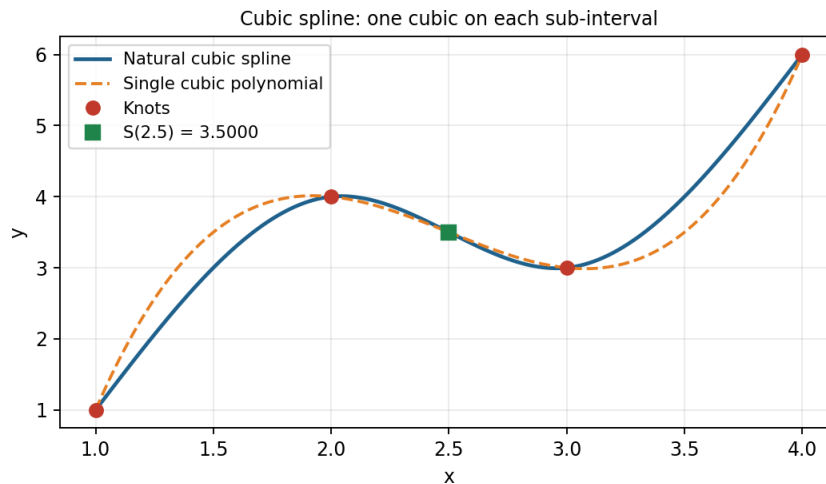


Figure 2.2 — Natural cubic spline through four knots; $S(2.5) = 3.5$.

Note: A spline has continuous first and second derivatives, so the curve looks smooth. This is why splines are used in computer graphics and data smoothing.

2.6 Least Squares Method (Curve Fitting)

When data contain measurement error, forcing a curve through every point is not sensible. The **method of least squares** chooses the parameters of a model $y = \varphi(x)$ to minimise $S = \sum [y_i - \varphi(x_i)]^2$, the sum of squared vertical deviations (residuals).

2.6.1 Straight-line fit $y = a + bx$

Setting $\partial S / \partial a = 0$ and $\partial S / \partial b = 0$ gives the **normal equations**:

$$\Sigma y = n \cdot a + b \cdot \Sigma x \quad \Sigma xy = a \cdot \Sigma x + b \cdot \Sigma x^2$$

Solving: $b = [n \Sigma xy - \Sigma x \Sigma y] / [n \Sigma x^2 - (\Sigma x)^2]$, $a = (\Sigma y - b \Sigma x) / n = \bar{y} - b \bar{x}$.

Example 2.6

Fit a straight line $y = a + bx$ to: (1, 2.2), (2, 2.8), (3, 3.6), (4, 4.5), (5, 5.1).

x	y	x^2	xy
1	2.2	1	2.2
2	2.8	4	5.6
3	3.6	9	10.8
4	4.5	16	18.0
5	5.1	25	25.5

x	y	x^2	xy
$\Sigma x = 15$	$\Sigma y = 18.2$	$\Sigma x^2 = 55$	$\Sigma xy = 62.1$

Solution

$$n = 5; b = (5 \cdot 62.1 - 15 \cdot 18.2) / (5 \cdot 55 - 15^2) = 37.5 / 50 = 0.750$$

$$a = (18.2 - 0.750 \cdot 15) / 5 = 1.390$$

Best-fit line: $y = 1.39 + 0.75x$. Prediction at $x = 6$: $y = 5.89$.

2.6.2 Non-linear data

Many non-linear models can be turned into a straight line by a change of variables, and then the linear formulas are applied to the transformed data. Fitting a parabola needs a third normal equation.

Model	Transformation	Linear form	Constants
$y = a \cdot e^{bx}$	$Y = \ln y$	$Y = \ln a + b \cdot x$	$a = e^A$, where A is the intercept
$y = a \cdot x^b$	$Y = \ln y, X = \ln x$	$Y = \ln a + b \cdot X$	$a = e^A$
$y = a \cdot b^x$	$Y = \log y$	$Y = \log a + x \cdot \log b$	$a = 10^A, b = 10^B$
$y = a + b/x$	$X = 1/x$	$y = a + b \cdot X$	direct

Parabola $y = a + bx + cx^2$ — the normal equations are

$$\Sigma y = n \cdot a + b \cdot \Sigma x + c \cdot \Sigma x^2$$

$$\Sigma xy = a \cdot \Sigma x + b \cdot \Sigma x^2 + c \cdot \Sigma x^3$$

$$\Sigma x^2 y = a \cdot \Sigma x^2 + b \cdot \Sigma x^3 + c \cdot \Sigma x^4$$

Example 2.7

Fit $y = a \cdot e^{bx}$ to (1, 7), (2, 11), (3, 17), (4, 27).

Taking logarithms: $\ln y = \ln a + b \cdot x$, i.e. $Y = A + bx$ with $A = \ln a$.

x	y	$Y = \ln y$	$x \cdot Y$	x^2
1	7	1.9459	1.9459	1
2	11	2.3979	4.7958	4
3	17	2.8332	8.4996	9
4	27	3.2958	13.1833	16

x	y	Y = ln y	x·Y	x ²
Σ = 10		10.4729	28.4247	30

Solution

$$b = (4 \cdot 28.4247 - 10 \cdot 10.4729) / (4 \cdot 30 - 10^2) = 0.4485$$

$$A = (10.4729 - 0.4485 \cdot 10) / 4 = 1.4969, \text{ so } a = e^{1.4969} = 4.4680$$

Fitted curve: $y = 4.468 \cdot e^{(0.449x)}$.

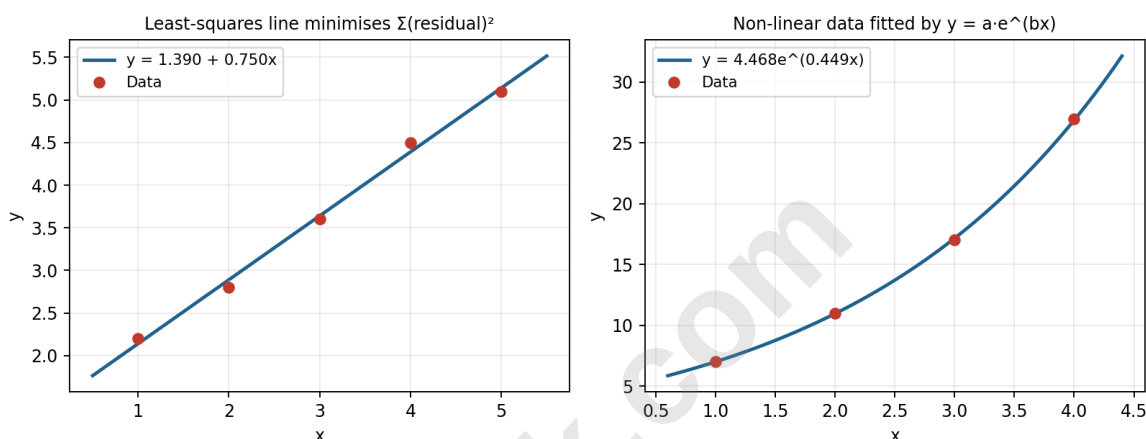


Figure 2.3 — Left: least-squares line (dotted segments are the residuals). Right: exponential fit.

Key Takeaways — Unit 2

- ✓ Through $n + 1$ distinct points there is a unique interpolating polynomial of degree $\leq n$; all methods give the same polynomial.
- ✓ Error: $E(x) = f^{(n+1)}(\xi)/(n + 1)! \cdot \Pi(x - x_i)$; it vanishes at the nodes and grows near the ends / outside the data.
- ✓ Lagrange: $P_n(x) = \sum y_i L_i(x)$; works for unequal spacing; $\sum L_i(x) = 1$.
- ✓ Newton forward (near start, $p = (x - x_0)/h$) and backward (near end, $p = (x - x_n)/h$) need equal spacing.
- ✓ Divided differences handle unequal spacing and allow new points to be added cheaply.
- ✓ Natural cubic spline: piecewise cubics with continuous S , S' , S'' and $S'' = 0$ at the ends.
- ✓ Least squares: minimise $\sum \text{residual}^2$; straight line normal equations $\sum y = na + b\sum x$, $\sum xy = a\sum x + b\sum x^2$; linearise exponential / power laws with logarithms.

— End of Unit 2 —