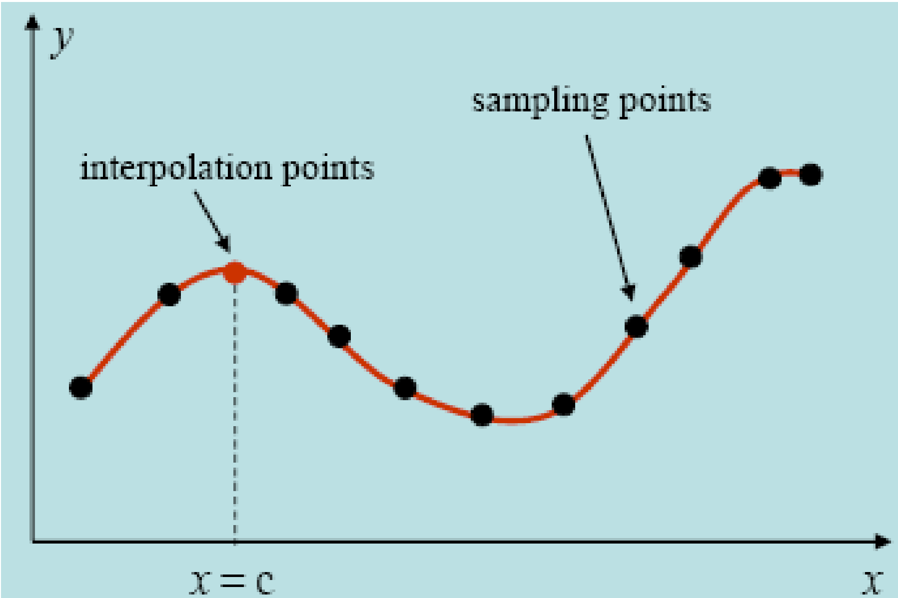
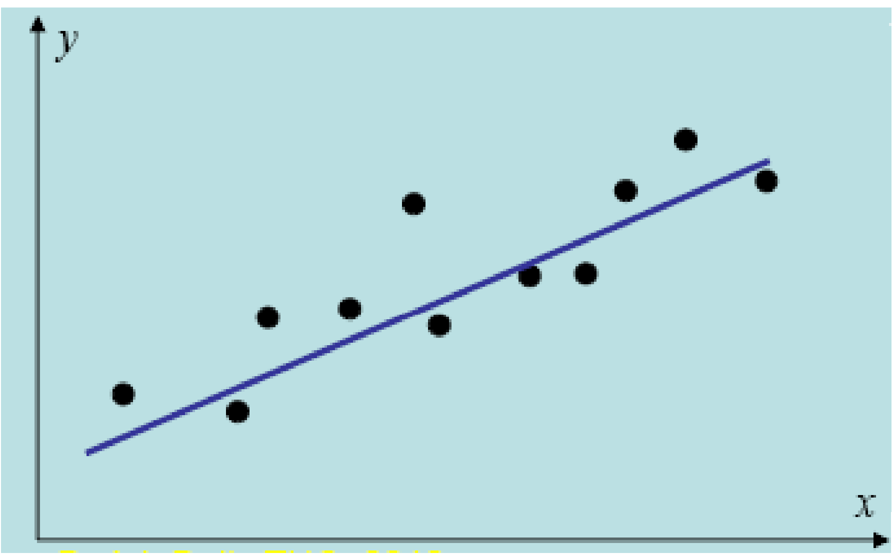
CE 205: Numerical Methods

Interpolation and Curve-fitting

Motivation





sampling data are acquired at *discrete points.*

- but the values at undefined points are wanted in many engineering applications

Interpolation

-for very precise data

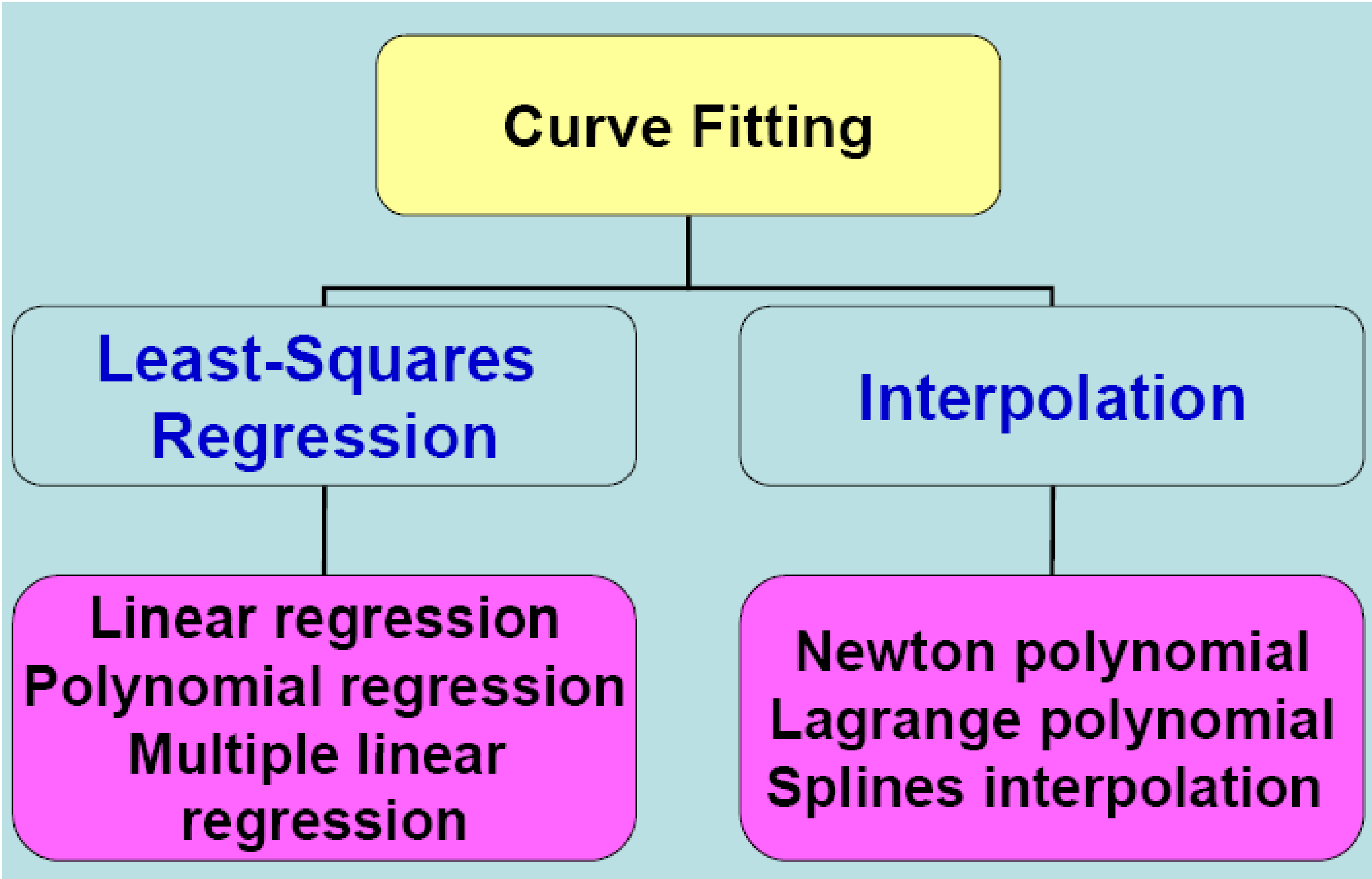
- Ex: density of water at different temp.

Least-squares Regression

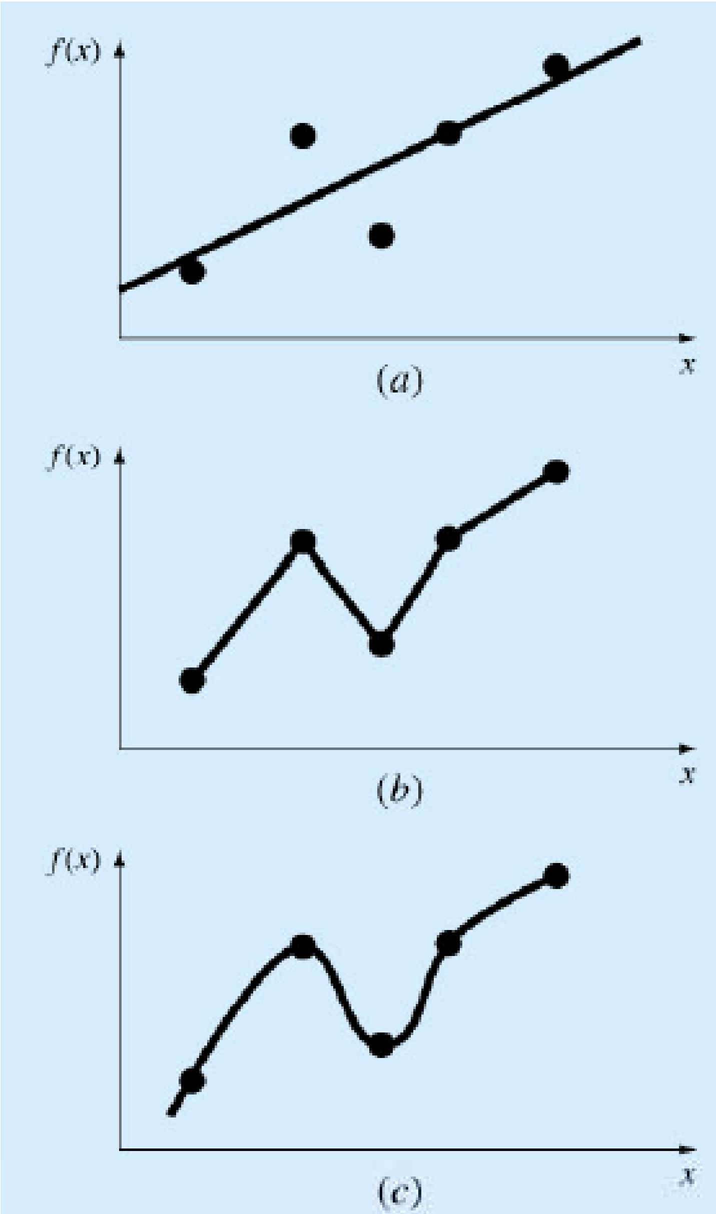
-for imprecise/noisy data

- Purpose is trend analysis

Interpolation and linear regression



Interpolation and linear regression



Least-square regression

Visually sketch a line that conforms to the data

Linear interpolation

Connect the data points consecutively by lines segments

Polynomial interpolation

Connect the data points consecutively by simple curves

Linear Least Squares Regression

A method to determine the "best" coefficients in a linear model for given data set [(x_1,y_1), (x_2,y_2)……(x_n,y_n)].

A straight line model: $$y = \underbrace{a_0 + a_1}_{\text{Coefficients}} x + \underbrace{e}_{\text{Residual Error}}$$

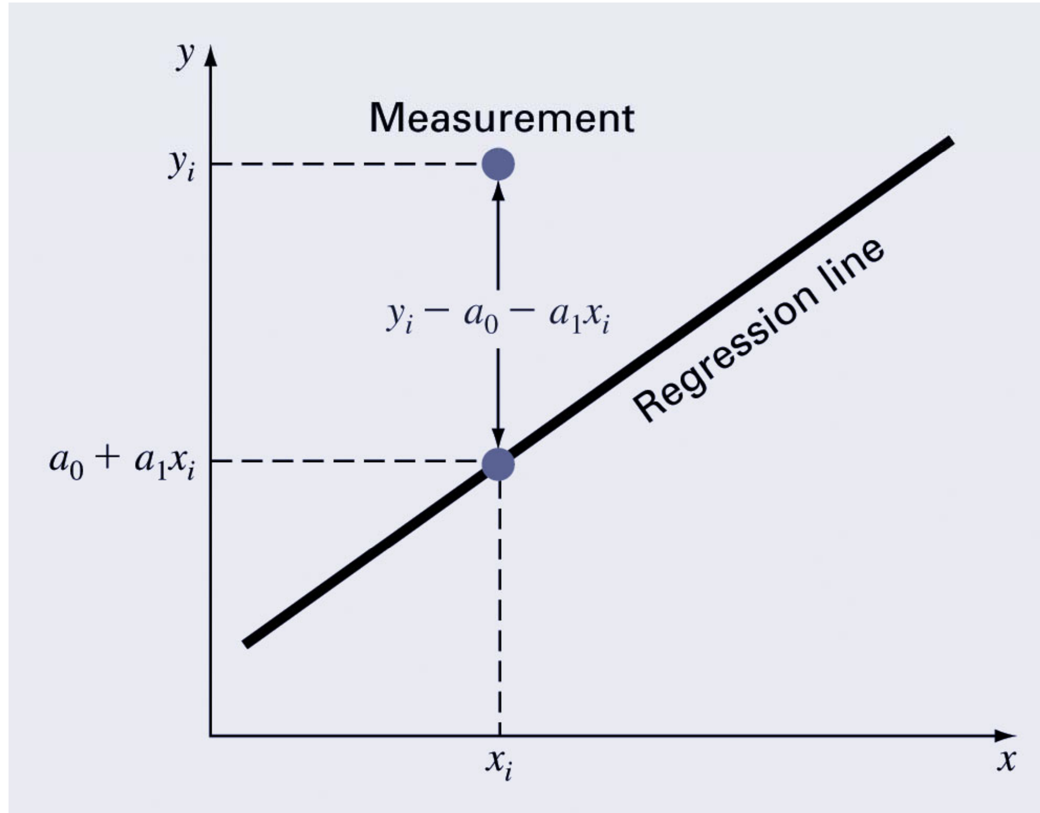
"Best" coefficients can be obtained by minimizing the sum of the squares of the *estimate* residuals.

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

Estimating the best-fit parameters

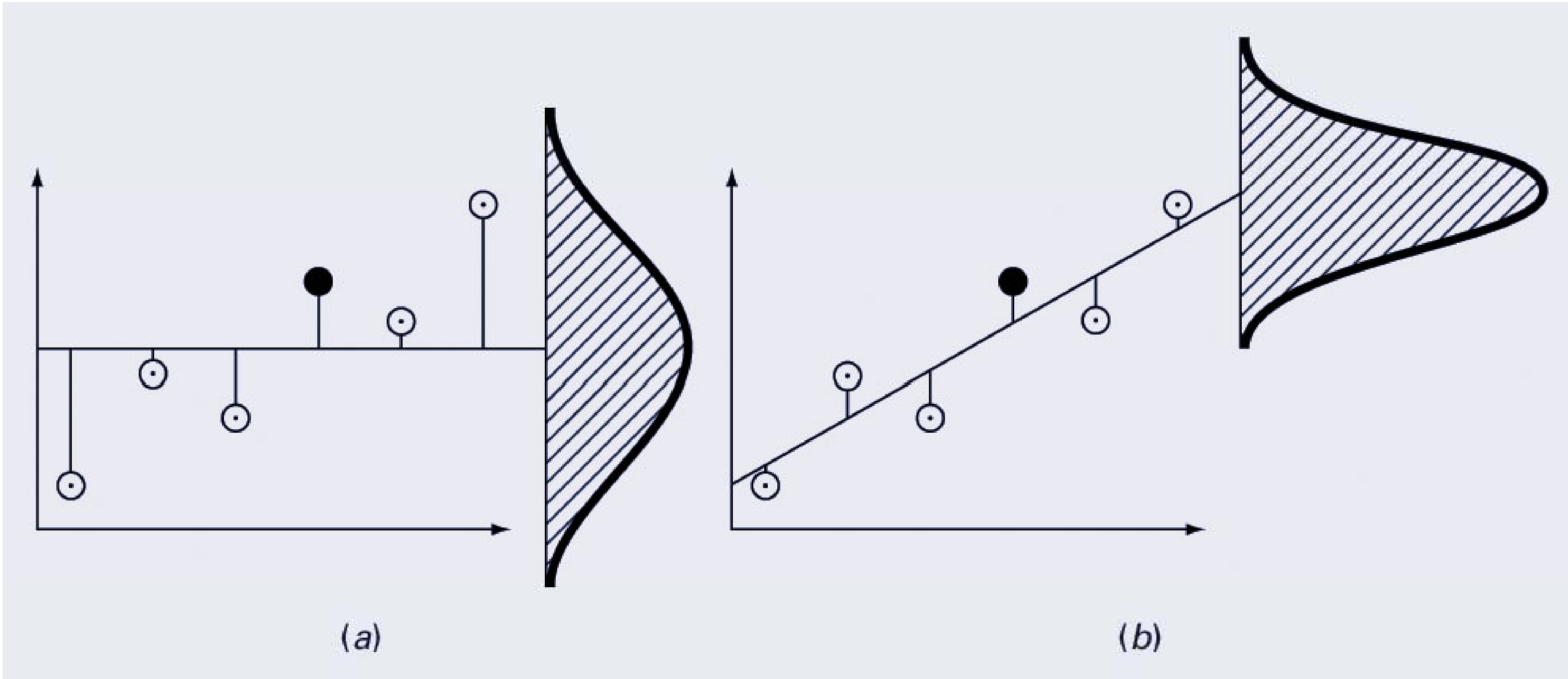
$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$

$$a_0 = \bar{y} - a_1 \bar{x}$$



Standard error of the estimate: $$s_{y/x} = \sqrt{\frac{S_r}{n-2}}$$

Improvement by linear regression



(a) the spread of data around the mean of the dependent data

(b) the spread of the data around the best fit line:

Sum of squares of residuals between data points and the mean

$$S_t = \sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2$$

Sum of squares of residuals between data points and the regression line

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

Assessing the "Goodness" of fit

S_t-S_r quantifies the improvement or error reduction due to describing data in terms of a straight line rather than as an average value

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

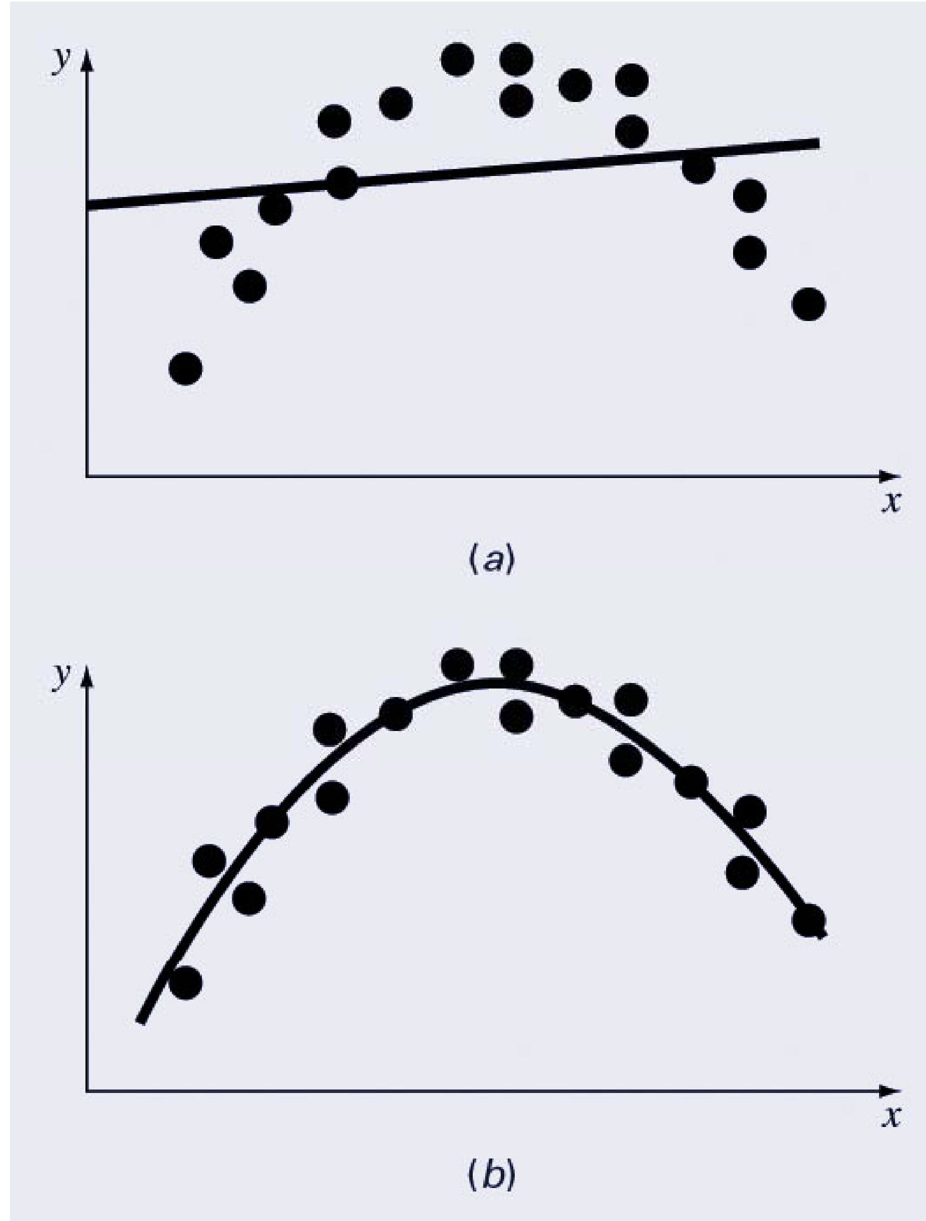
r^2 represents the percentage of the original uncertainty explained by the model (Coefficient of Determination)

- For a perfect fit, S_r=0 and r^2=1.
- If r^2=0, there is no improvement over simply picking the mean.
- If r^2<0, the model is *worse* than simply picking the mean!

Polynomial regression

Some engineering data is poorly represented by a straight line.

A higher order polynomial (e.g. parabola) may be well-suited



Polynomial regression

For a second order polynomial, the least-square regression can be extended to

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)^2$$

The coefficients can be determined using Gaussian elimination:

$$\begin{bmatrix} n & \Sigma x_i & \Sigma x_i^2 \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i^3 \\ \Sigma x_i^2 & \Sigma x_i^3 & \Sigma x_i^4 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \Sigma y_i \\ \Sigma x_i y_{1i} \\ \Sigma x_i y_i \end{Bmatrix}$$

Polynomial regression

The two-dimensional case can easily be extended to m-th order polynomial

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

The sum of squares of residuals:

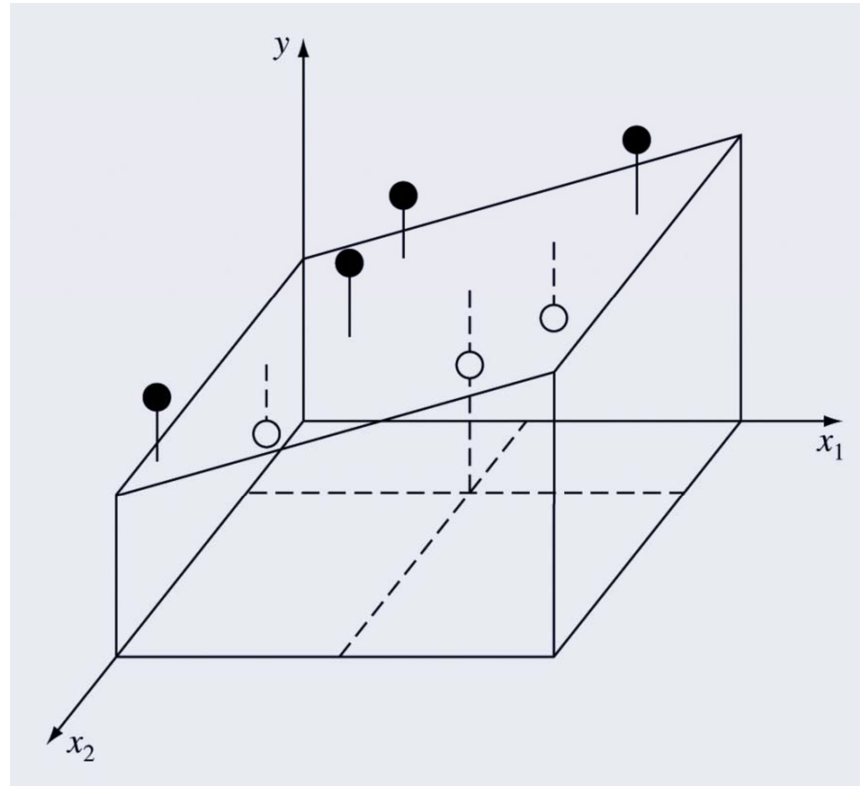
$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - \cdots - a_m x_i^m \right)^2$$

Can you formulate the matrix to determine the coefficients?

Multiple linear regression

y is a linear function of two or more independent variables:

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots a_m x_m$$



Again, the best fit is obtained by minimizing the sum of the squares of the estimate residuals:

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_{1,i} - a_2 x_{2,i} - \cdots a_m x_{m,i}\right)^2$$

Nonlinear relationship

Linear regression is predicated on the fact that the relationship between the dependent and independent variables is linear - this is not always the case.

Three common examples are:

$$\text{exponential}: \quad y = \alpha_1 e^{\beta_1 x}$$

$$\text{power}: \quad y = \alpha_2 x^{\beta_2}$$

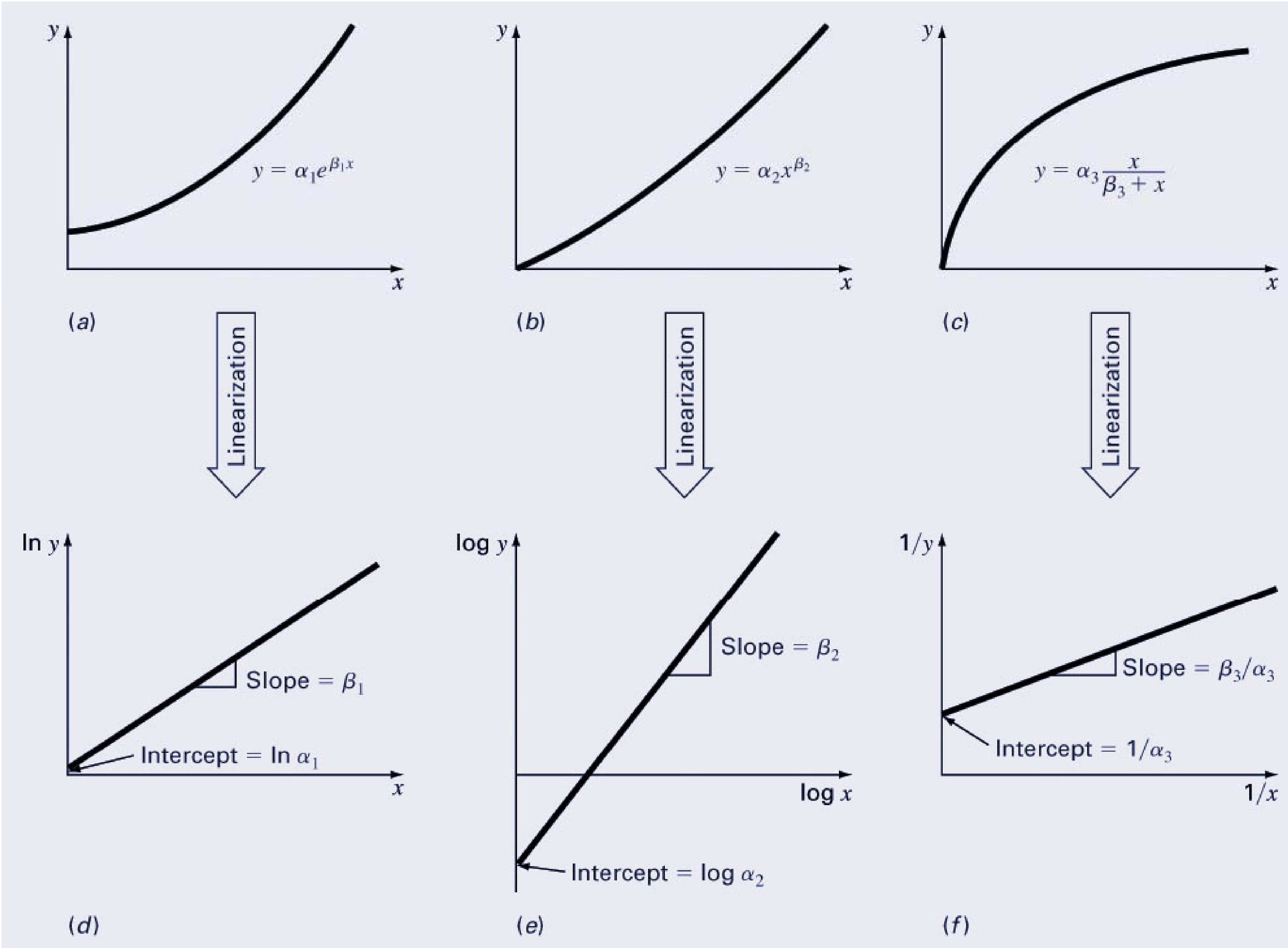
$$\text{saturation-growth-rate}: \quad y = \alpha_3 \frac{x}{\beta_3 + x}$$

Linearizing nonlinear relationship

this may involve taking logarithms or inversion:

Model	Nonlinear	Linearized
exponential:	$y = \alpha_1 e^{\beta_1 x}$	$\ln y = \ln \alpha_1 + \beta_1 x$
power:	$y = \alpha_2 x^{\beta_2}$	$\log y = \log \alpha_2 + \beta_2 \log x$
saturation-growth-rate:	$y = \alpha_3 \dfrac{x}{\beta_3 + x}$	$\dfrac{1}{y} = \dfrac{1}{\alpha_3} + \dfrac{\beta_3}{\alpha_3}\dfrac{1}{x}$

Linearizing nonlinear relationship



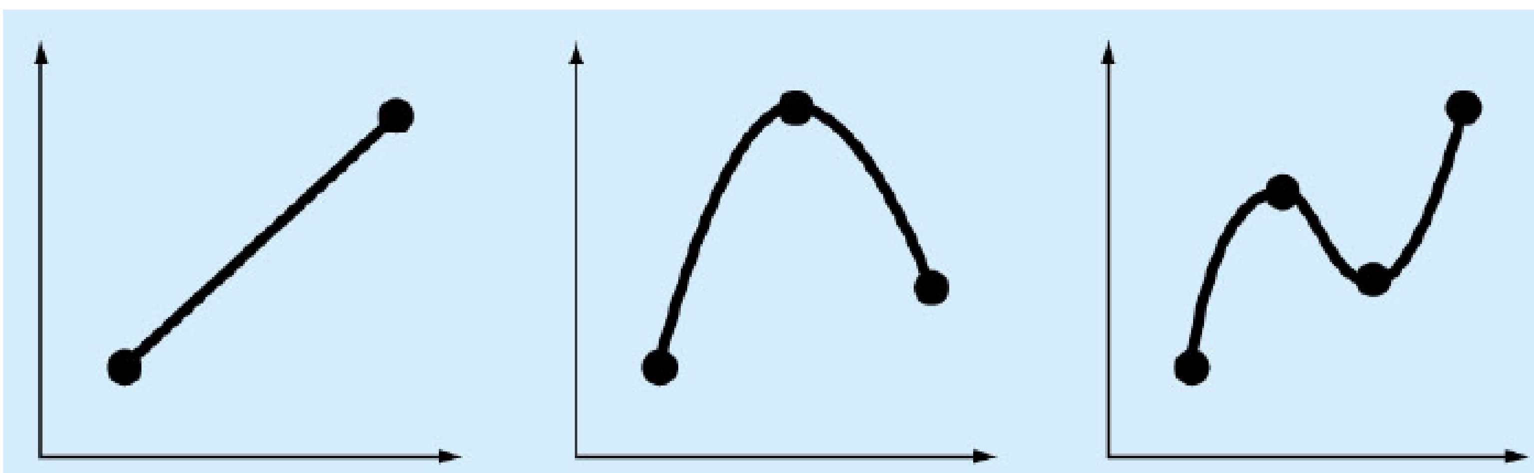
Interpolation

Estimating intermediate values between precise data points.

- more restrictive than fitting a curve
- Example: finding density of water at intermediate temperatures from a table

there is one and only one *nth-order* polynomial that fits *n+1 points*

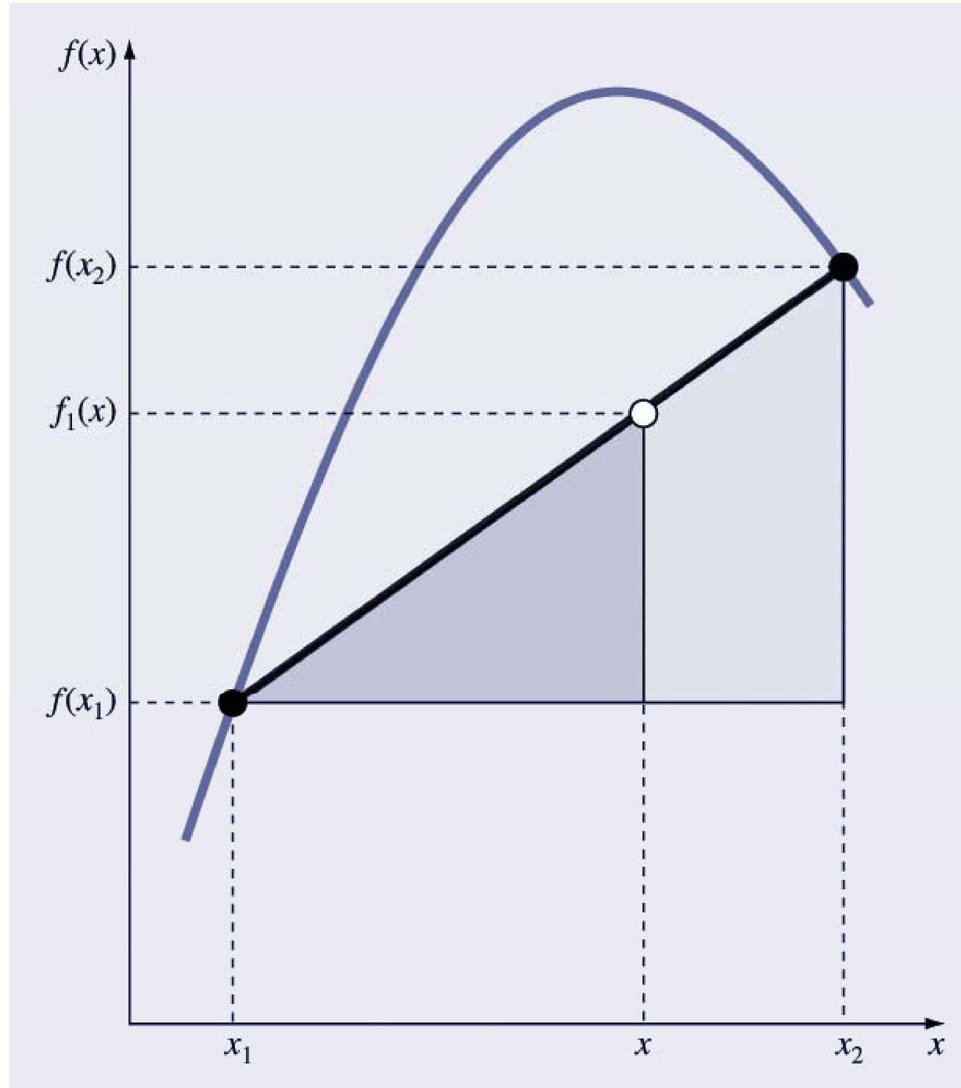
$$f(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1}$$



Newton's interpolating polynomials

<u>First order:</u>

The first-order Newton interpolating polynomial may be obtained from linear interpolation and similar triangles

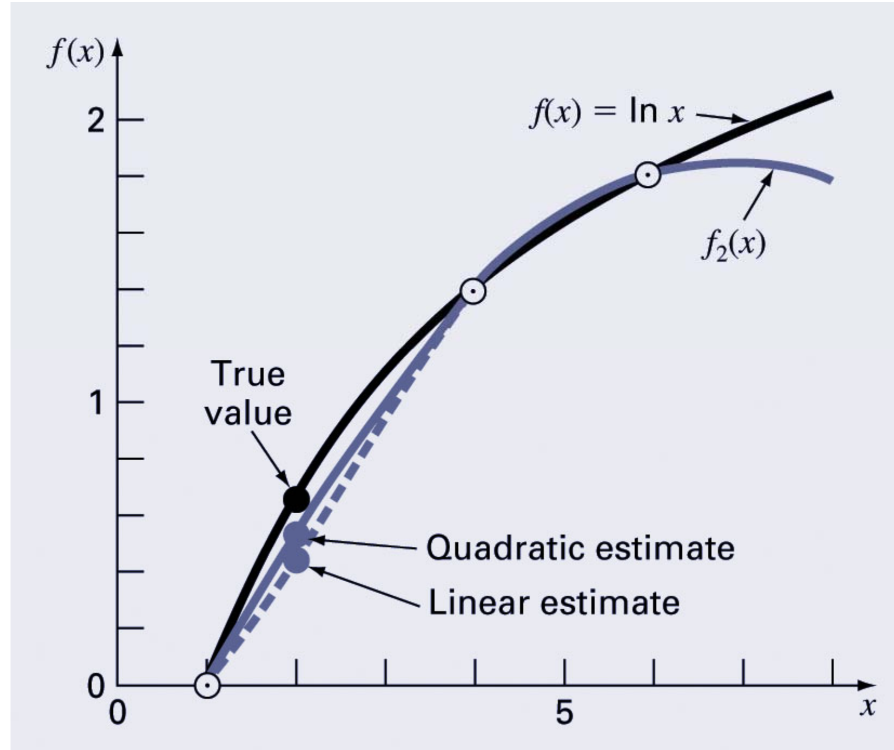


$$f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Newton's interpolating polynomials

<u>Second order/quadratic:</u>

Availability of three points can improve the estimate by introducing curvature



$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$$

Values of coefficients can be obtained by a simple procedure:

$$b_0 = f(x_0)$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$b_2 = \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

Newton's interpolating polynomials

General form:

n-th order polynomial to fit *n*+1 data points

$$f_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \ldots.. + b_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

$$\begin{aligned} b_0 &= f\left(x_0\right) \\ b_1 &= f\left[x_1, x_0\right] \\ b_2 &= f\left[x_2, x_1, x_0\right] \\ &\vdots \\ b_n &= f\left[x_n, x_{n-1}, \cdots, x_1, x_0\right] \end{aligned}$$

f[….] represent divided differences

Divided Differences

How to calculate divided differences

$$f\left[x_i, x_j\right] = \frac{f(x_i) - f(x_j)}{x_i - x_j}$$

$$f\left[x_i, x_j, x_k\right] = \frac{f\left[x_i, x_j\right] - f\left[x_j, x_k\right]}{x_i - x_k}$$

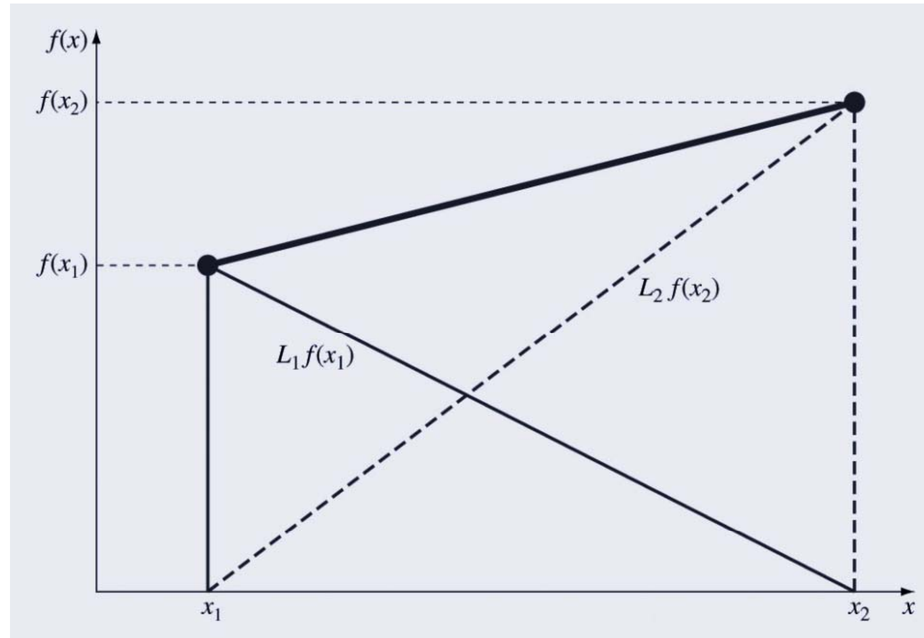
$$f\left[x_n, x_{n-1}, \cdots, x_2, x_1\right] = \frac{f\left[x_n, x_{n-1}, \cdots, x_2\right] - f\left[x_{n-1}, x_{n-2}, \cdots, x_1\right]}{x_n - x_1}$$

x_i	$f(x_i)$	First	Second	Third
x_1	$f(x_1)$	$f[x_2, x_1]$	$f[x_3, x_2, x_1]$	$f[x_4, x_3, x_2, x_1]$
x_2	$f(x_2)$	$f[x_3, x_2]$	$f[x_4, x_3, x_2]$	
x_3	$f(x_3)$	$f[x_4, x_3]$		
x_4	$f(x_4)$			

Lagrange interpolating polynomial

First order:

The first-order Lagrange polynomial may be obtained from weighted combination of two linear interpolations



$$f_1(x) = L_1 f\left(x_1\right) + L_2 f\left(x_2\right)$$

$$L_1 = \frac{x - x_2}{x_1 - x_2}, L_2 = \frac{x - x_1}{x_2 - x_1}$$

$$f_1(x) = \frac{x - x_2}{x_1 - x_2} f\left(x_1\right) + \frac{x - x_1}{x_2 - x_1} f\left(x_2\right)$$

Lagrange interpolating polynomial

Second order:

$$f_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

General form:

The Lagrange interpolating polynomial for *n* points

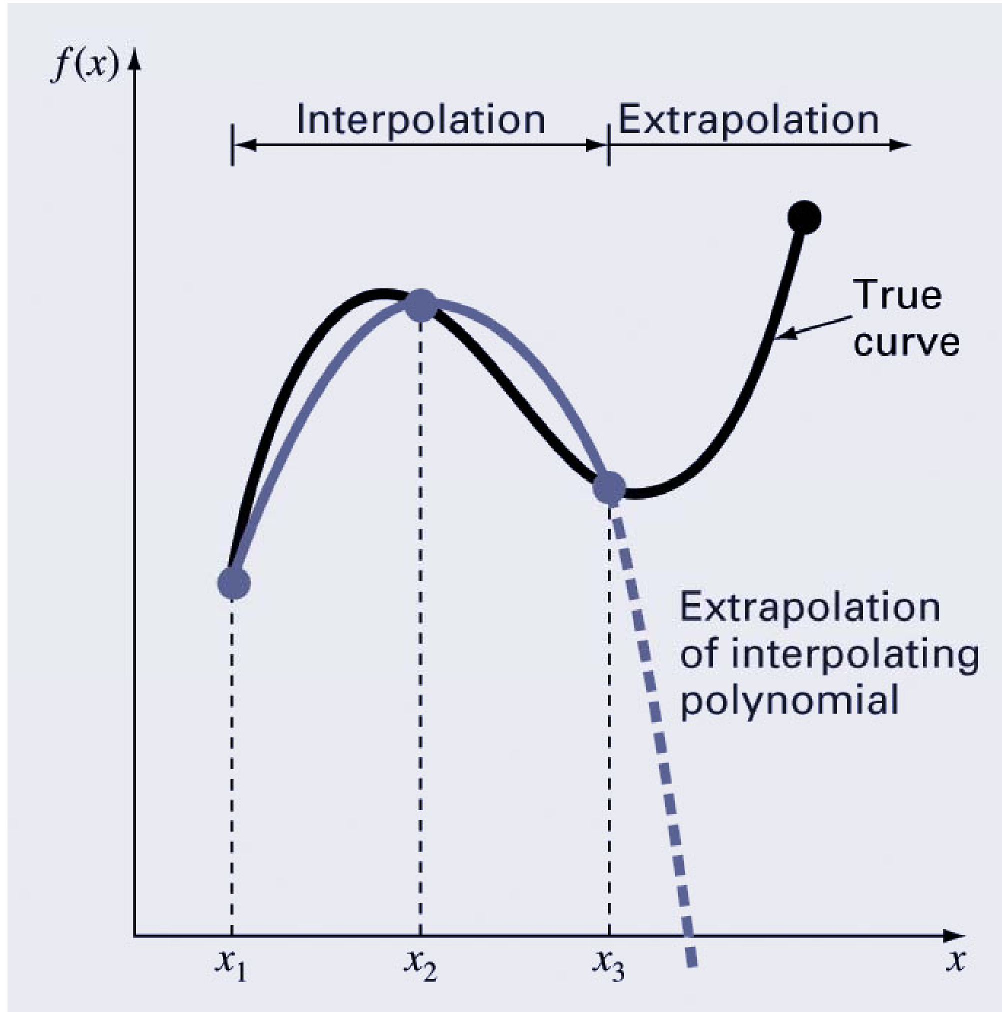
$$f_{n-1}(x_i) = \sum_{i=1}^{n} L_i(x) f(x_i)$$

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

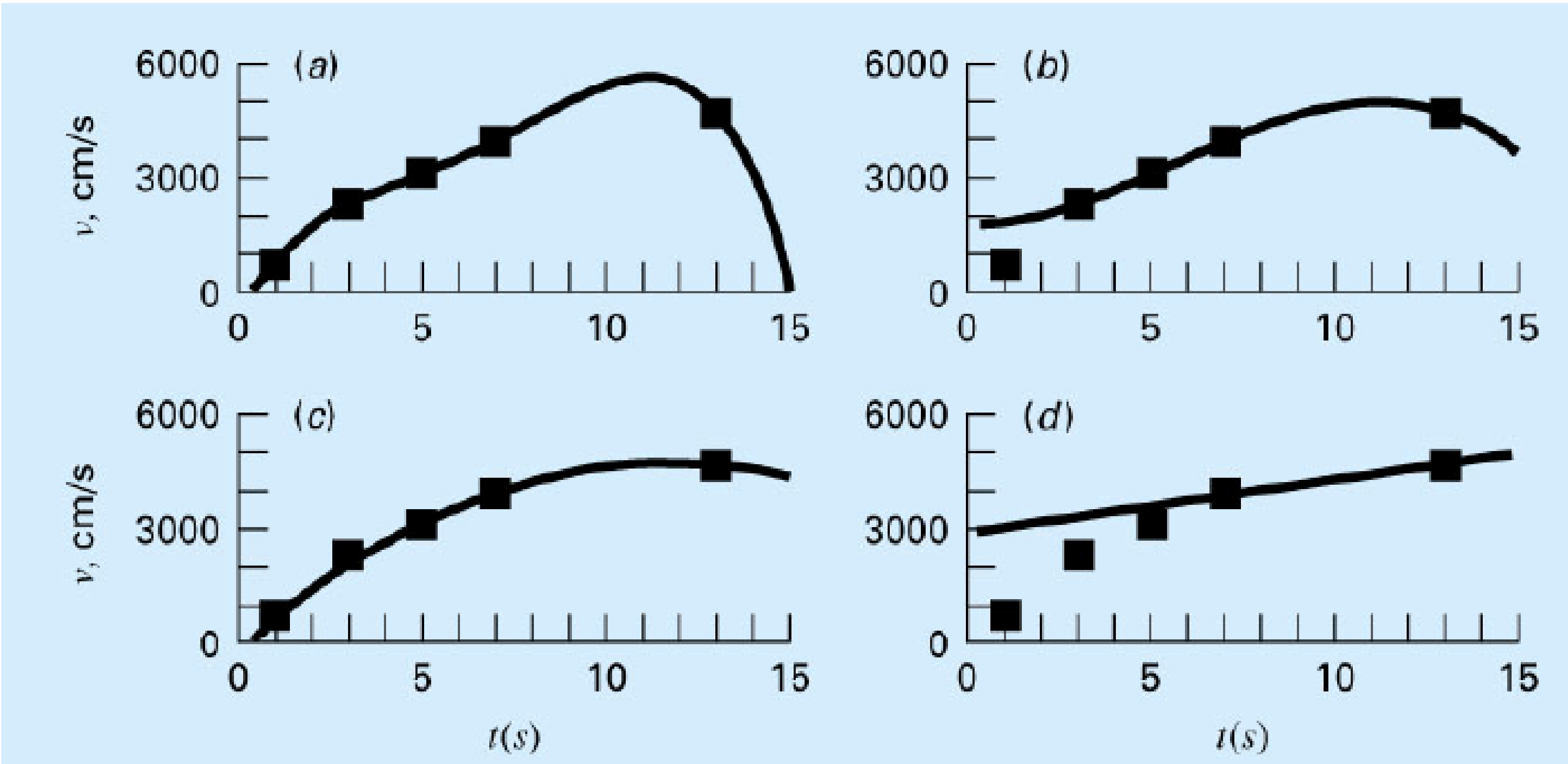
Extrapolation

the process of estimating a value of $f(x)$ that lies outside the range of the known base points.

Extrapolation represents a step into the unknown, and extreme care should be exercised when extrapolating!

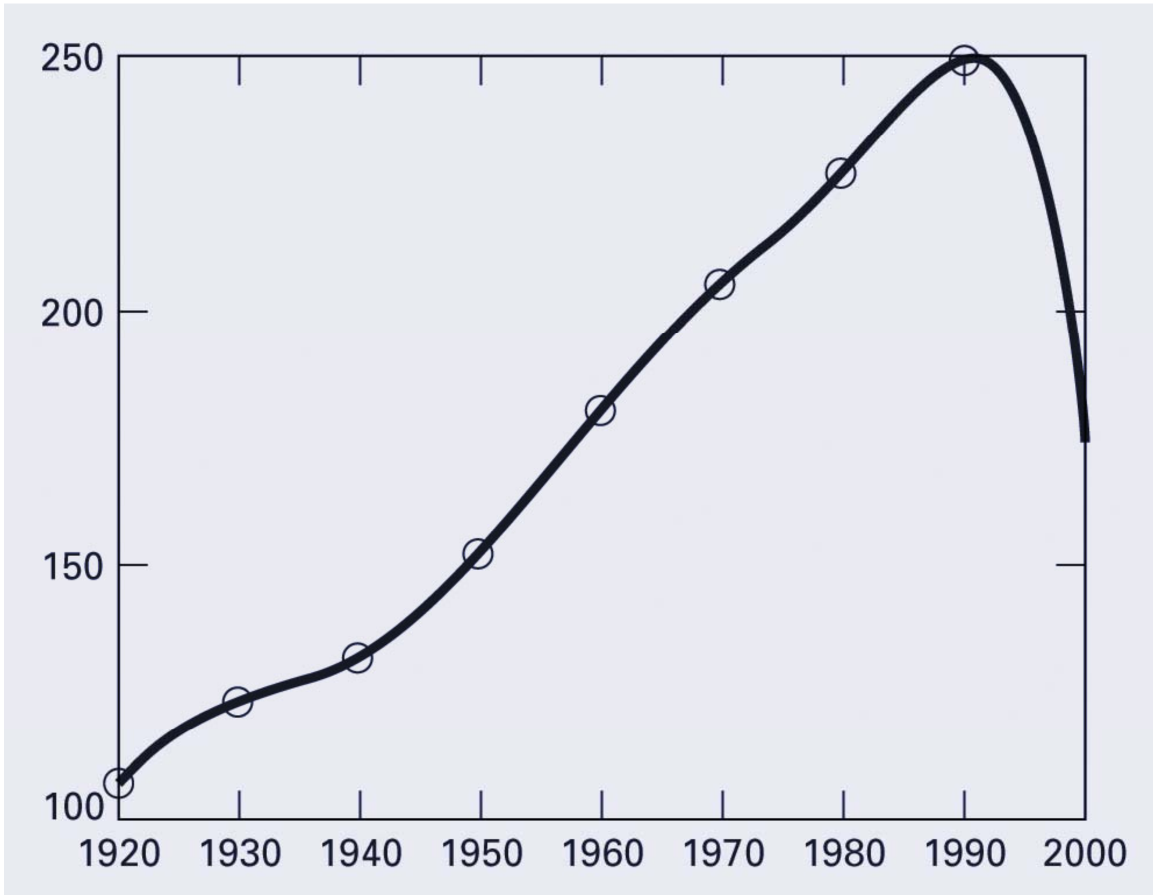


Hazards of higher order polynomials



Higher order polynomials tend to overshoot the trend of the data

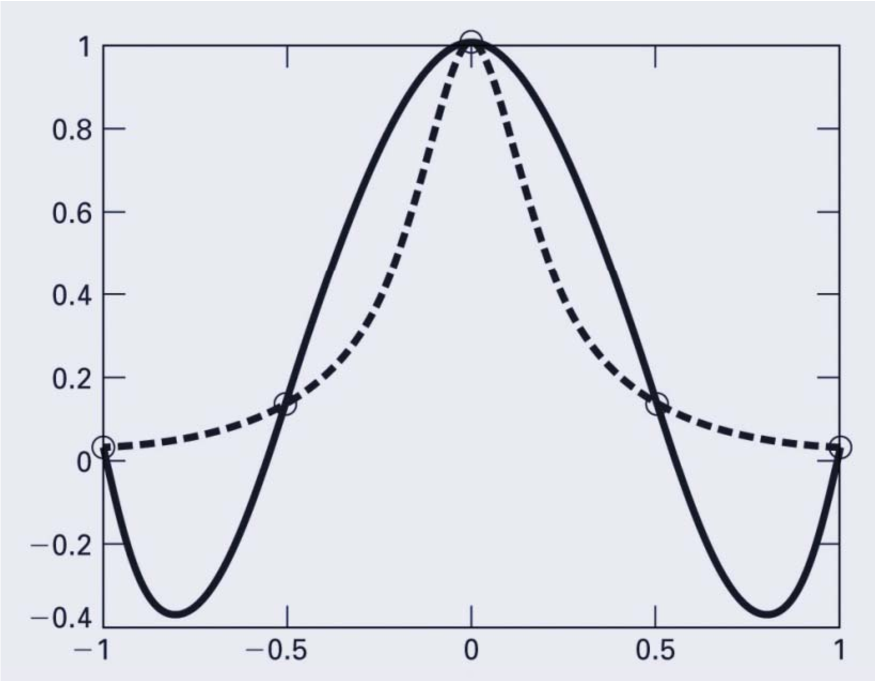
Hazards of higher order polynomials

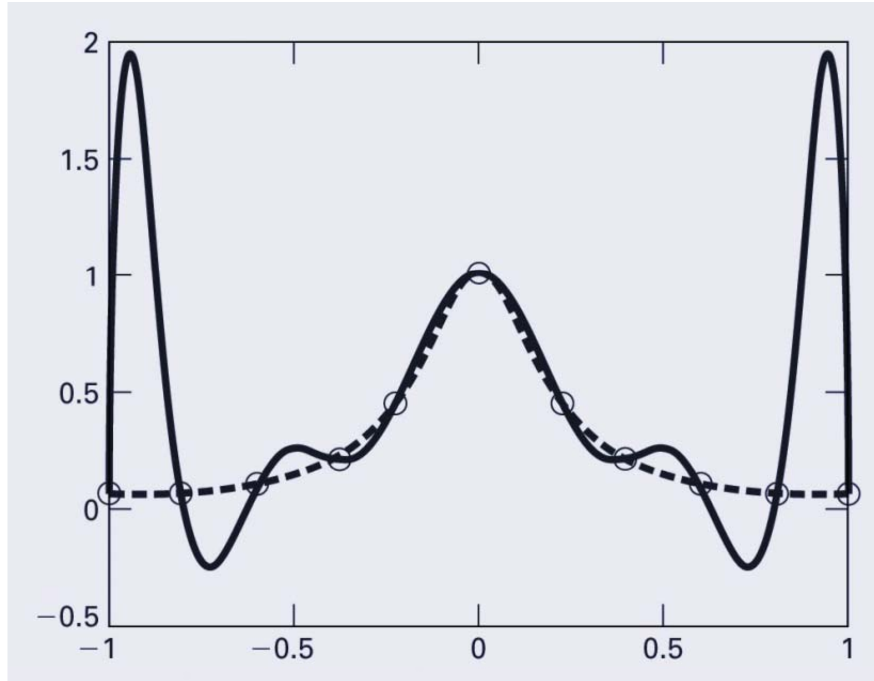


A seventh order population dataset

-Difficult to use for extrapolation

Hazards of higher order polynomials





Can also introduce oscillations to an interpolation or fit where they should not be.

Alternate approach: spline interpolation

Splines eliminate oscillations by using small subsets of points for each interval rather than every point. This is especially useful when there are jumps in the data:

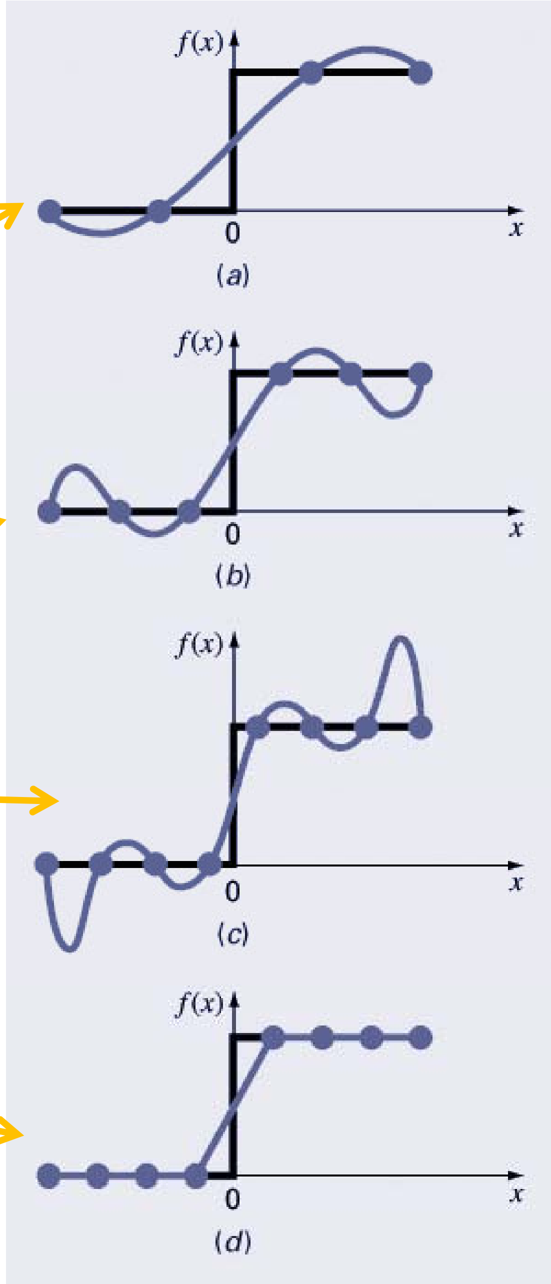
a) 3^{rd} order polynomial

b) 5^{th} order polynomial

c) 7^{th} order polynomial

d) Linear spline

seven 1st order polynomials generated by using pairs of points at a time

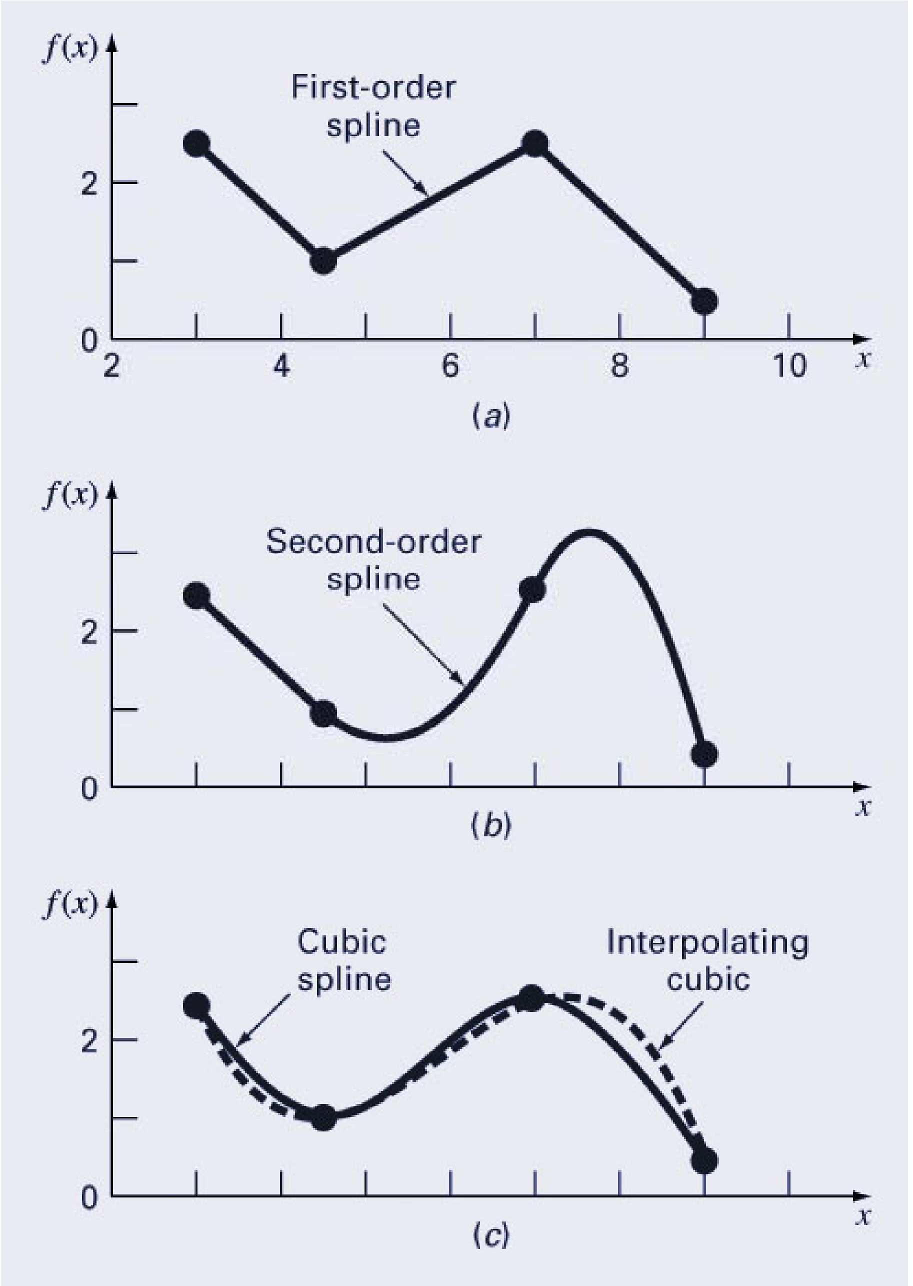


Three classes of spline interpolation

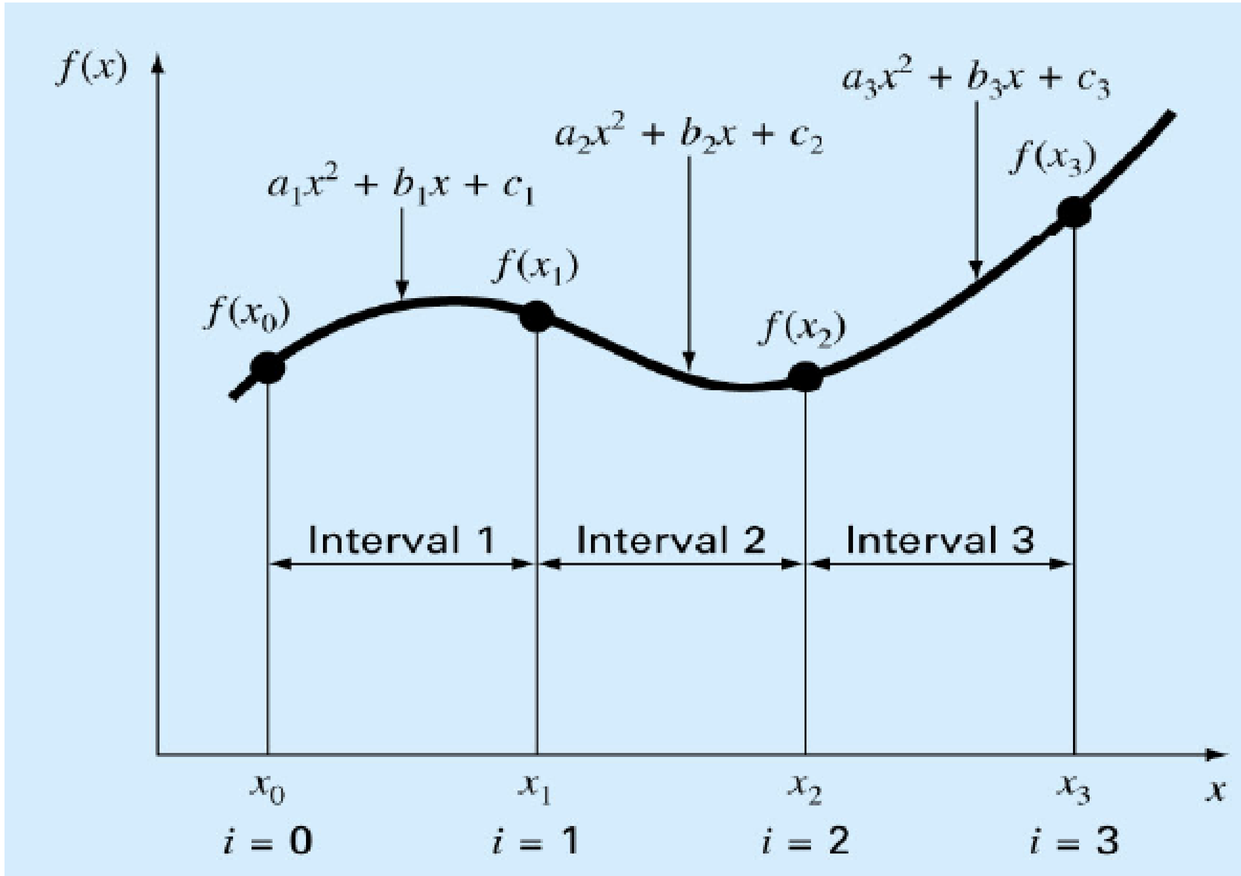
First-order splines
- Straight-line equations between each pair of points
- Go through the points

Second-order splines
- Quadratic equations between each pair of points
- Go through the points
- Match first derivatives at the interior points

Third-order splines
- Cubic equations between each pair of points
- Go through the points
-Match first and second derivatives at the interior points
-Different from cubic interpolating polynomial



Quadratic splines



(n+1) datapoints → n intervals → 3n unknowns

→ 3n conditions required

Quadratic splines

Function values of adjacent polynomials equal at the interior knots → (2n – 2) conditions

First and last functions must pass through the endpoints → 2 conditions

The first derivatives at the interior knots must be equal → (n – 1) conditions

Assume that the second derivative is zero at the first point → 1 condition

<u>Total (2n – 2)+ 2 + (n -1) + 1 = 3n conditions</u>

Cubic splines

cubic splines are preferred because they provide the simplest representation that exhibits the desired appearance of smoothness.

The general form of the i-th spline:

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

(n+1) datapoints → n intervals → 4n unknowns

→ 4n conditions required

Quadratic splines

Function values of adjacent polynomials equal at the interior knots → (2n – 2) conditions

First and last functions must pass through the endpoints → 2 conditions

The first derivatives at the interior knots must be equal → (n – 1) conditions

The second derivatives at the interior knots must be equal → (n – 1) conditions

<u>Total (2n – 2)+ 2 + (n -1) + (n-1) = (4n-2) conditions</u>

Still need two more conditions…..

Several options for the final two conditions

Natural end conditions

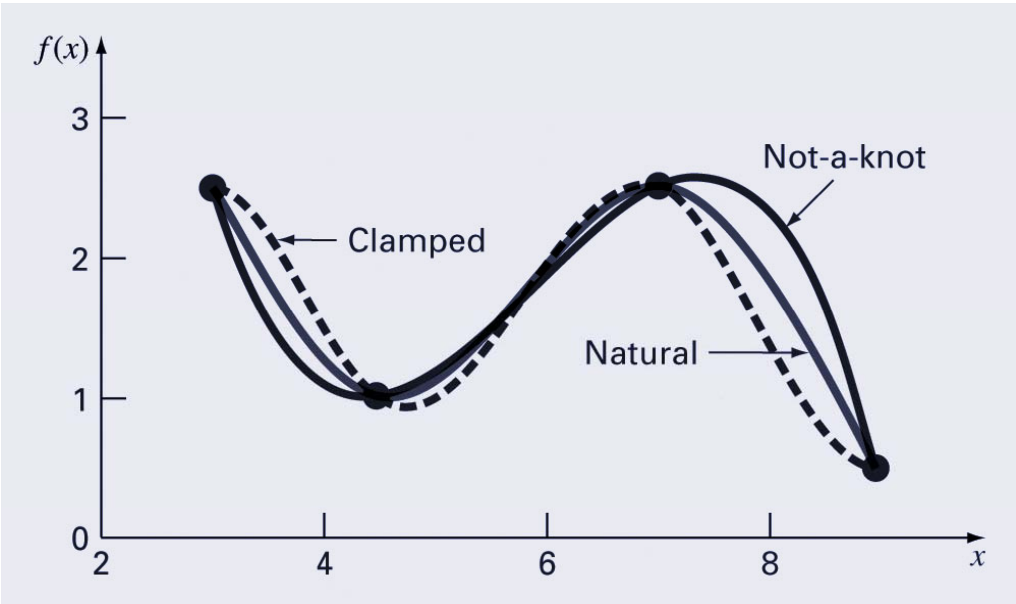
- the second derivative at the end knots are zero.

Clamped end conditions

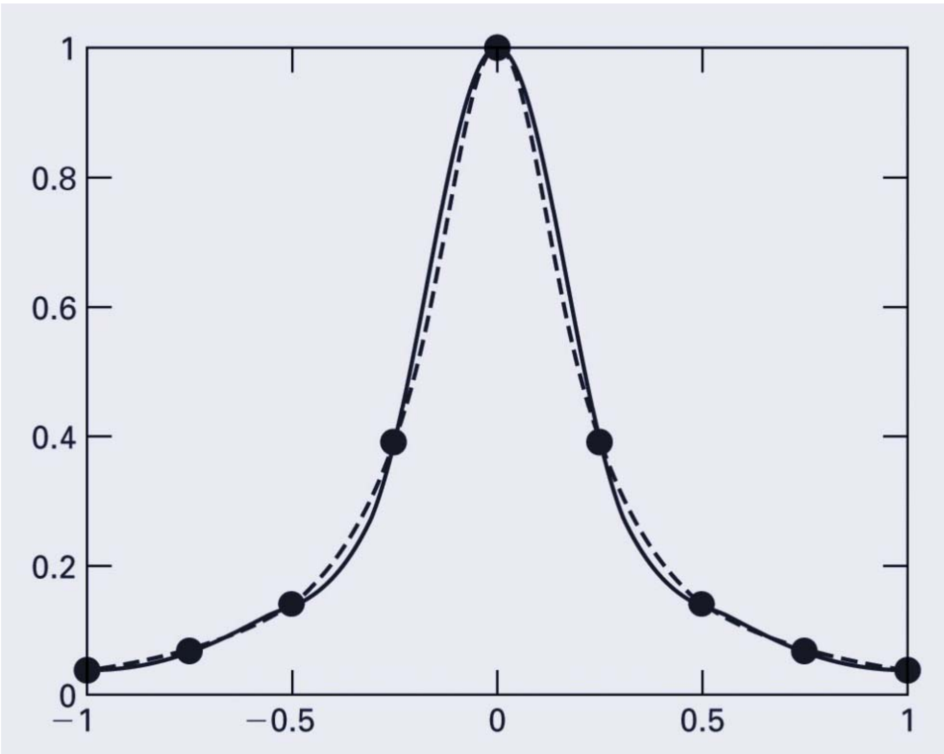
- the first derivatives at the first and last knots are known.

"Not-a-knot" end conditions

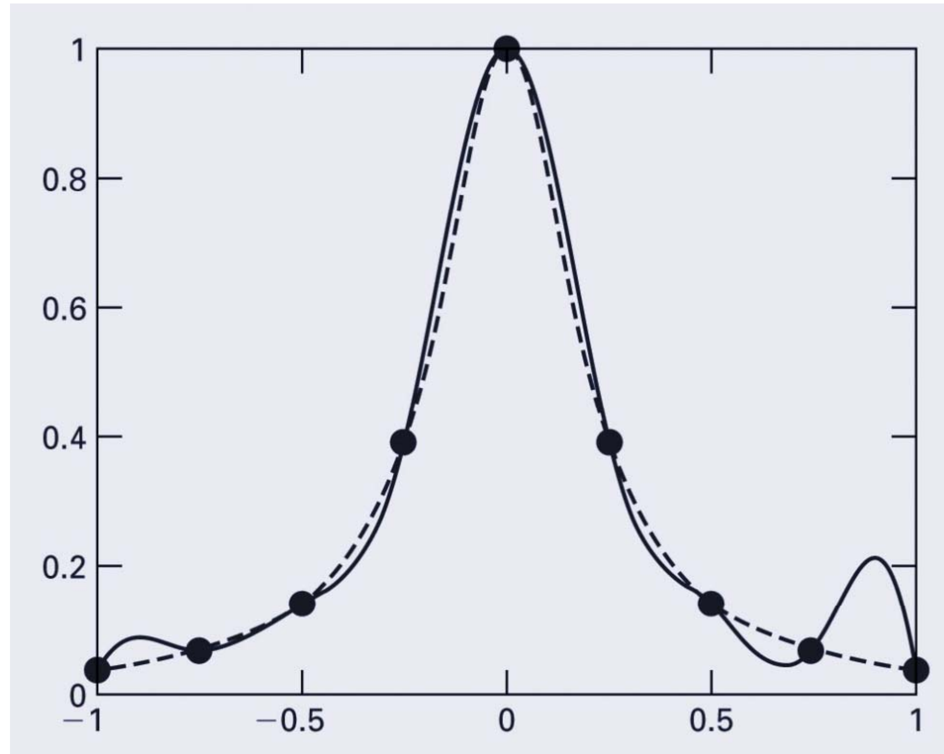
- force continuity of the *third* derivative at the second and penultimate points



Examples: “Not-a-knot” and Clamped end



“Not-a-knot” end conditions



Clamped end conditions

Piecewise polynomial examples