

# Syllabus

## Section A



- **NUMERICAL DIFFERENTIATION AND INTEGRATION:-** Approximating the derivative, Numerical differentiation formulas, Introduction to Numerical quadrature, Newton-Cotes formula, Gaussian Quadrature.
- **SOLUTION OF NONLINEAR EQUATIONS:-** Bracketing methods for locating a root, Initial approximations and convergence criteria, Newton- Raphson and Secant methods, Solution of problems through a structural programming language such as C or Pascal.

## Section B

- **ERRORS IN NUMERICAL CALCULATIONS:-** Introduction, Numbers and their accuracy, Absolute, relative and percentage errors and their analysis, General error formula.
- **INTERPOLATION AND CURVE FITTING:-** Taylor series and calculation of functions, Introduction to interpolation, Lagrange approximation, Newton Polynomials, Chebyshev Polynomials, Least squares line, curve fitting, Interpolation by spline functions.

## Section C



- **SOLUTION OF LINEAR SYSTEMS:-** Direct Methods, Gaussian elimination and pivoting, Matrix inversion, UV factorization, Iterative methods for linear systems, Solution of problems through a structured programming language such as C or Pascal.
- **EIGEN VALUE PROBLEMS:-** Jacobi, Given's and Householder's methods for symmetric matrices, Rutishauser method for general matrices, Power and inverse power methods.

## Section D

- **SOLUTION OF DIFFERENTIAL EQUATIONS:-** Introduction to differential equations, Initial value problems, Euler's methods, Heun's method, Runge-Kutta methods, Taylor series method, PredictorCorrector methods, Systems of differential equations, Boundary value problems, Finite-difference method, Solution of problems through a structured programming language such as C or Pascal.
- **PARTIAL DIFFERENTIAL EQUATIONS, EIGENVALUES AND EIGENVECTORS:-** Solution of hyperbolic, parabolic and elliptic equations, The eigen value problem, The power method and the Jacobi's method for eigen value problems, Solution of problems through a structural programming language such as C or Pascal.



# SECTION A

# Approximations and Round-Off Errors

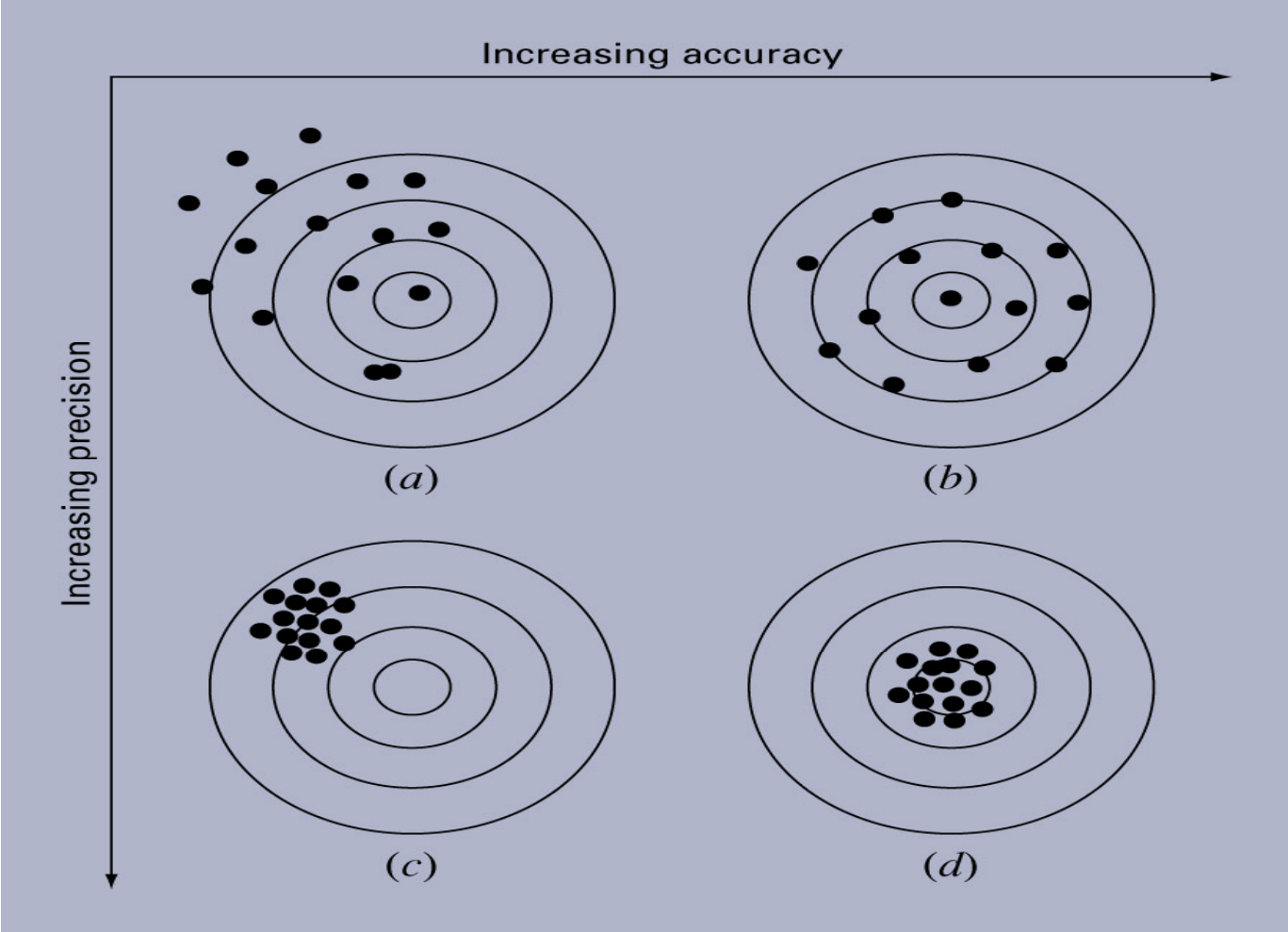


- For many engineering problems, we cannot obtain analytical solutions.
- Numerical methods yield approximate results, results that are close to the exact analytical solution. We cannot exactly compute the errors associated with numerical methods.
  - Only rarely given data are exact, since they originate from measurements. Therefore there is probably error in the input information.
  - Algorithm itself usually introduces errors as well, e.g., unavoidable round-offs, etc ...
  - The output information will then contain error from both of these sources.
- How confident we are in our approximate result?
- The question is “*how much error is present in our calculation and is it tolerable?*”



- Accuracy. How close is a computed or measured value to the true value
- Precision (or reproducibility). How close is a computed or measured value to previously computed or measured values.
- Inaccuracy (or bias). A systematic deviation from the actual value.
- Imprecision (or uncertainty). Magnitude of scatter.

Fig. 1.1



# Significant Figures



- Number of significant figures indicates precision. Significant digits of a number are those that can be *used* with *confidence*, e.g., the number of certain digits plus one estimated digit.

53,800 How many significant figures?

|                      |   |
|----------------------|---|
| $5.38 \times 10^4$   | 3 |
| $5.380 \times 10^4$  | 4 |
| $5.3800 \times 10^4$ | 5 |

Zeros are sometimes used to locate the decimal point not significant figures.

|            |   |
|------------|---|
| 0.00001753 | 4 |
| 0.0001753  | 4 |
| 0.001753   | 4 |

# Error Definitions



$$\text{True Value} = \text{Approximation} + \text{Error}$$

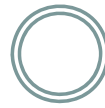
$$E_t = \text{True value} - \text{Approximation} (+/-)$$

True error

$$\text{True fractional relative error} = \frac{\text{true error}}{\text{true value}}$$

$$\text{True percent relative error, } \varepsilon_t = \frac{\text{true error}}{\text{true value}} \times 100\%$$





- For numerical methods, the true value will be known only when we deal with functions that can be solved analytically (simple systems). In real world applications, we usually not know the answer a priori. Then

$$\varepsilon_a = \frac{\text{Approximate error}}{\text{Approximation}} \times 100 \%$$

$$\varepsilon_a = \frac{\text{Current approximation} - \text{Previous approximation}}{\text{Current approximation}} \times 100\%$$

(+ / -)

- *Iterative approach*, example Newton's method



- Use absolute value.
- Computations are repeated until stopping criterion is satisfied.

$$|\mathcal{E}_a| < \mathcal{E}_s$$

Pre-specified % tolerance based on the knowledge of your solution

- If the following criterion is met

$$\mathcal{E}_s = (0.5 \times 10^{(2-n)})\%$$

you can be sure that the result is correct to at least n significant figures.

# Round-off Errors



- Numbers such as  $\pi$ ,  $e$ , or  $\sqrt{7}$  cannot be expressed by a fixed number of significant figures.
- Computers use a base-2 representation, they cannot precisely represent certain exact base-10 numbers.
- Fractional quantities are typically represented in computer using “floating point” form, e.g.,

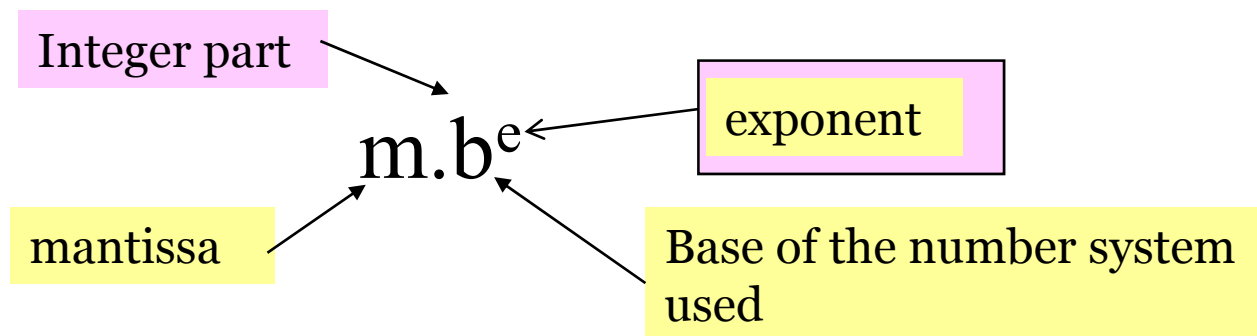




Figure 1.3

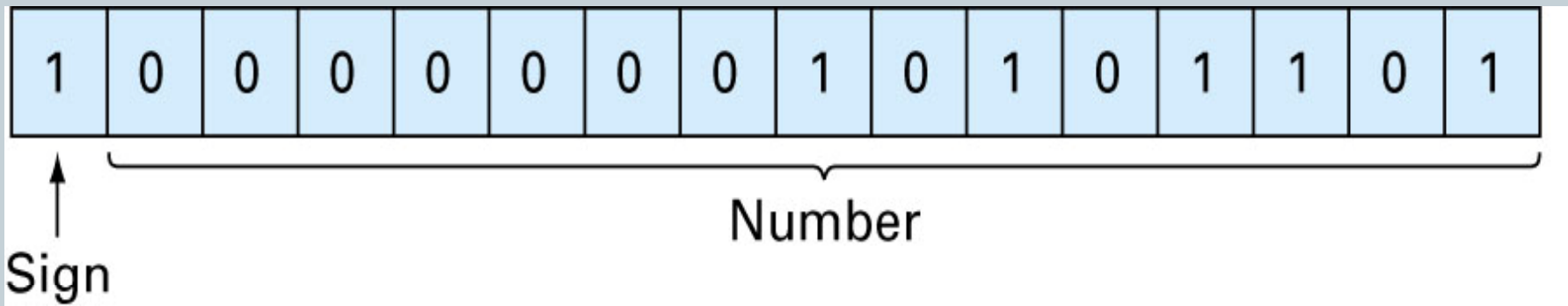
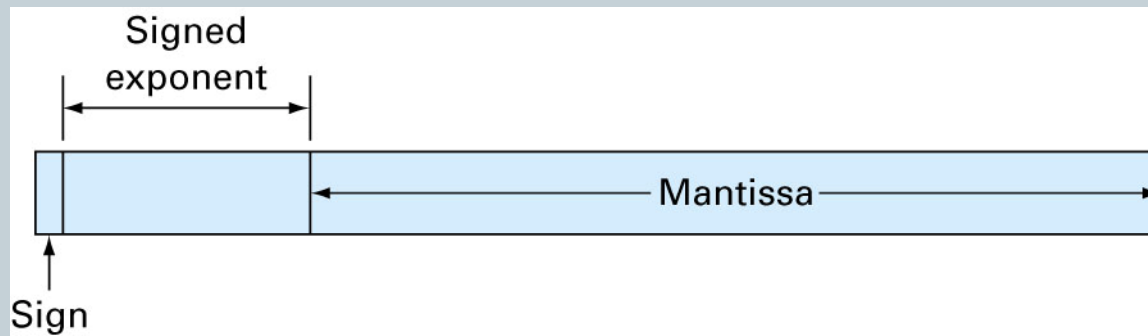


Figure 1.4





156.78      ►►      0.15678x10<sup>3</sup> in a floating point base-10 system

$$\frac{1}{34} = 0.029411765$$

Suppose only 4 decimal places to be stored

$$0.0294 \times 10^0$$

$$\frac{1}{2} \leq |m| < 1$$

- Normalized to remove the leading zeroes. Multiply the mantissa by 10 and lower the exponent by 1

$$0.294\underline{1} \times 10^{-1}$$

Additional significant figure is retained



Therefore  $\frac{1}{b} \leq |m| < 1$

for a base-10 system  $0.1 \leq m < 1$

for a base-2 system  $0.5 \leq m < 1$

- Floating point representation allows both fractions and very large numbers to be expressed on the computer. However,
  - Floating point numbers take up more room.
  - Take longer to process than integer numbers.
  - Round-off errors are introduced because mantissa holds only a finite number of significant figures.



# Interpolation

# Taylor's Series and Interpolation

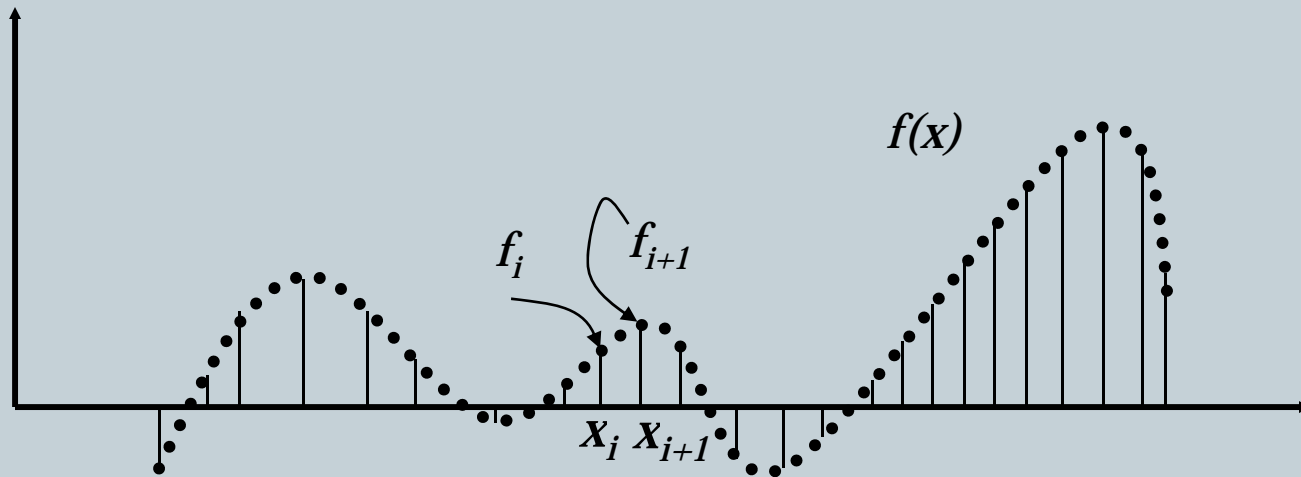
18

- Taylor Series interpolates at a specific point:
  - The function
  - Its first derivative
  - ...
- It may not interpolate at other points.
- We want an interpolant at several  $f(c)$ 's.

# Basic Scenario

19

- We are able to *prod* some **function**, but do not know what it really is.
- This gives us a list of data points:  $[x_i, f_i]$



# Interpolation & Curve-fitting

20

- Often, we have data sets from experimental/observational measurements
  - Typically, find that the **data/dependent variable/output** varies...
  - As the **control parameter/independent variable/input** varies. Examples:
    - ✦ Classic gravity drop: location changes with time
    - ✦ Pressure varies with depth
    - ✦ Wind speed varies with time
    - ✦ Temperature varies with location
- Scientific method: Given data identify underlying relationship
- Process known as **curve fitting**:

# Interpolation & Curve-fitting

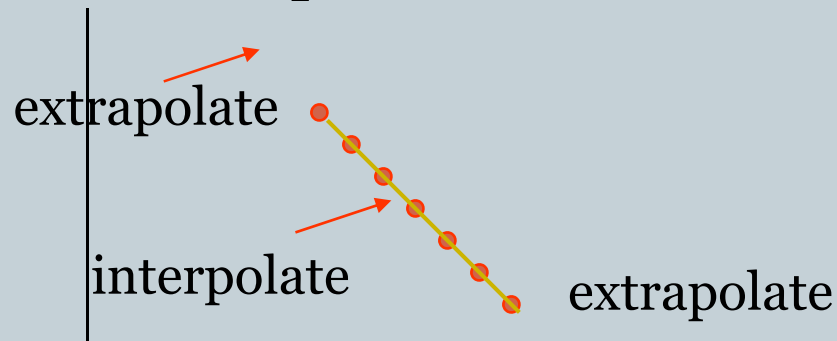
21

- Given a data set of  $n+1$  points  $(x_i, y_i)$  identify a function  $f(x)$  (the **curve**), that is in some (well-defined) sense the **best fit** to the data
- Used for:
  - Identification of underlying relationship (modelling/prediction)
  - Interpolation (filling in the gaps)
  - Extrapolation (predicting outside the range of the data)

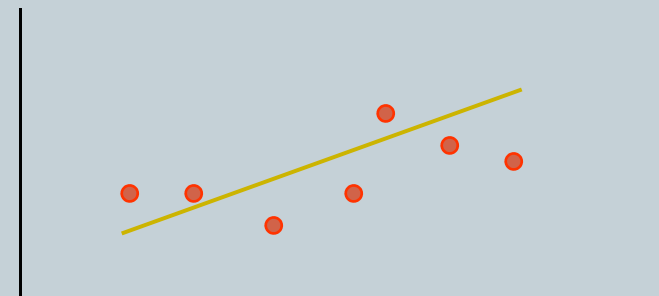
# Interpolation Vs Regression

22

- Distinctly different approaches depending on the quality of the data
- Consider the pictures below:



Pretty confident:  
there is a polynomial relationship  
Little/no scatter  
Want to find an expression  
that passes **exactly** through all the points



Unsure what the relationship is  
Clear scatter  
Want to find an expression  
that captures the trend:  
**minimize** some measure of the error  
Of all the points...

# Interpolation

23

- Concentrate first on the case where we believe there is no error in the data (and round-off is assumed to be negligible).
- So we have  $y_i=f(x_i)$  at  $n+1$  points  $x_0, x_1, \dots, x_i, \dots, x_n$ :  $x_j > x_{j-1}$
- (Often but not always evenly spaced)
- In general, we do not know the underlying function  $f(x)$
- Conceptually, interpolation consists of two stages:
  - Develop a simple function  $g(x)$  that
    - ✦ Approximates  $f(x)$
    - ✦ Passes through all the points  $x_i$
  - Evaluate  $f(x_t)$  where  $x_0 < x_t < x_n$

# Interpolation

24

- Clearly, the crucial question is the selection of the simple functions  $g(x)$
- Types are:
  - Polynomials
  - Splines
  - Trigonometric functions
  - Spectral functions...Rational functions etc...



# Curve Approximation

25

- We will look at three possible approximations (time permitting):
  - Polynomial interpolation
  - Spline (polynomial) interpolation
  - Least-squares (polynomial) approximation
- If you know your function is periodic, then trigonometric functions may work better.
  - Fourier Transform and representations

# Polynomial Interpolation

26

- Consider our data set of  $n+1$  points  $y_i=f(x_i)$  at  $n+1$  points  $x_0, x_1, \dots, x_i, \dots, x_n$ :  $x_j > x_{j-1}$
- In general, given  $n+1$  points, there is a unique polynomial  $g_n(x)$  of order  $n$ :
- That passes through all  $n+1$  points

$$g_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

# Polynomial Interpolation

27

- There are a variety of ways of expressing the same polynomial
- Lagrange interpolating polynomials
- Newton's divided difference interpolating polynomials
- We will look at both forms

# Polynomial Interpolation

28

- Existence – does there exist a polynomial that **exactly** passes through the  $n$  data points?
- Uniqueness – Is there more than one such polynomial?
  - We will assume uniqueness for now and prove it latter.

# Lagrange Polynomials

29

- Summation of terms, such that:
  - Equal to  $f()$  at a data point.
  - Equal to zero at all other data points.
  - Each term is a  $n^{\text{th}}$ -degree polynomial

Existence!!!

$$p_n(x) = \sum_{i=0}^n L_i(x) f(x_i)$$

$$L_i(x) = \prod_{k=0, k \neq i}^n \frac{(x - x_k)}{(x_i - x_k)}$$

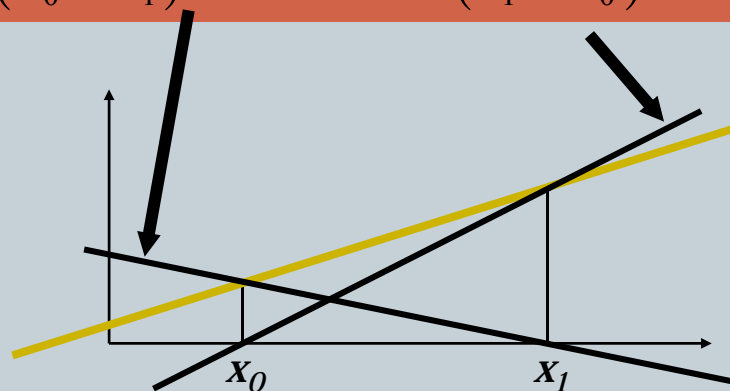
$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

# Linear Interpolation

30

- Summation of two lines:

$$\begin{aligned} p_1(x) &= \sum_{i=0}^1 L_i(x) f(x_i) \\ &= \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1) \end{aligned}$$



Remember this when we  
talk about piecewise-  
linear splines

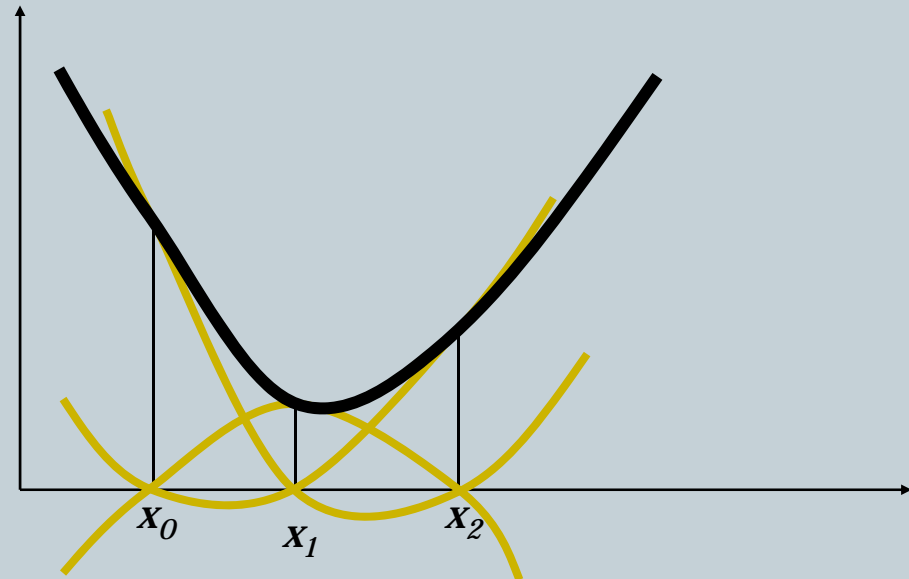
# Lagrange Polynomials

31

- 2<sup>nd</sup> Order Case => quadratic polynomials

Adding them all together, we get the interpolating quadratic polynomial, such that:

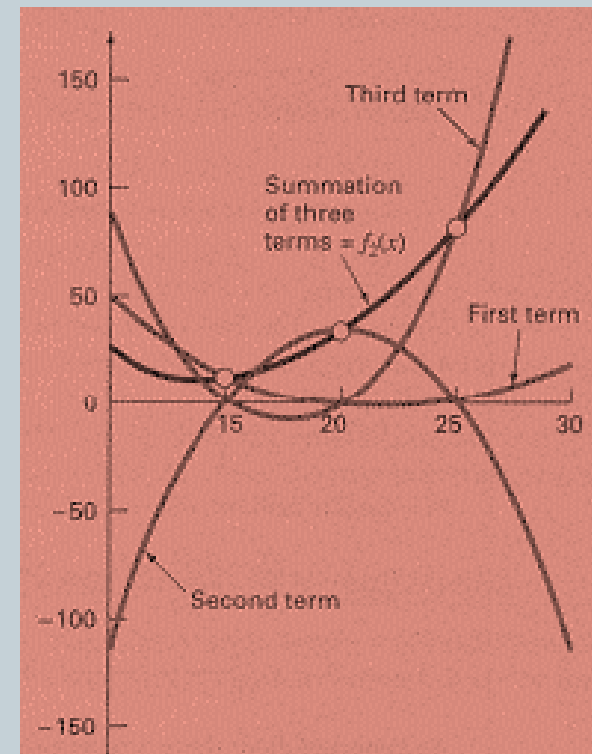
- $P(x_0) = f_0$
- $P(x_1) = f_1$
- $P(x_2) = f_2$



# Lagrange Polynomials

32

- Sum must be a unique 2<sup>nd</sup> order polynomial through all the data points.
- What is an efficient implementation?





# Newton Interpolation

33

- Consider our data set of  $n+1$  points  $y_i=f(x_i)$  at  $x_0, x_1, \dots, x_i, \dots, x_n$ :  $x_n > x_0$
- Since  $p_n(x)$  is the **unique** polynomial  $p_n(x)$  of order  $n$ , write it:

$$p_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \dots + b_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

$$b_0 = f(x_0)$$

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

$$b_2 = f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0}$$

⋮

$$b_n = f[x_n, x_{n-1}, \dots, x_0] = \frac{f[x_n, \dots, x_1] - f[x_{n-1}, \dots, x_0]}{x_n - x_0}$$

- $f[x_i, x_j]$  is a **first divided difference**
- $f[x_2, x_1, x_0]$  is a **second divided difference**, etc.

# Invariance Theorem

34

- Note, that the order of the data points does not matter.
- All that is required is that the data points are distinct.
- Hence, the divided difference  $f[x_0, x_1, \dots, x_k]$  is invariant under all permutations of the  $x_i$ 's.

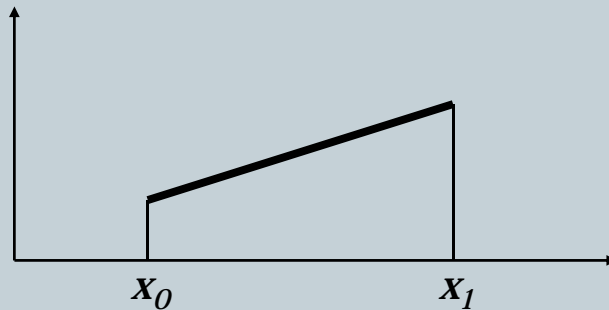
# Linear Interpolation

35

- Simple linear interpolation results from having only 2 data points.

$$p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0} (x - x_0)$$

slope



# Quadratic Interpolation

36

- Three data points:

$$\begin{aligned} p_2(x) &= f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &= f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + \frac{\left[ \frac{f(x_2) - f(x_1)}{x_2 - x_1} \right] - \left[ \frac{f(x_1) - f(x_0)}{x_1 - x_0} \right]}{x_2 - x_0}(x - x_0)(x - x_1) \\ &= f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \\ &\quad + \frac{\left( \left[ \frac{f(x_2) - f(x_1)}{x_2 - x_1} \right] (x - x_1) \right) (x - x_0) - \left( \left[ \frac{f(x_1) - f(x_0)}{x_1 - x_0} \right] (x - x_0) \right) (x - x_1)}{x_2 - x_0} \end{aligned}$$

# Newton Interpolation

37

- Let's look at the recursion formula:

- For the q: 
$$b_n = f[x_n, x_{n-1}, \dots, x_0] = \frac{f[x_n, \dots, x_1] - f[x_{n-1}, \dots, x_0]}{x_n - x_0}$$

where

$$f[x_i] = f(x_i)$$

$$\begin{aligned} b_2 = f[x_2, x_1, x_0] &= \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} \\ &= \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - b_1}{x_2 - x_0} \end{aligned}$$

# Evaluating for $x_2$

38

$$\begin{aligned} f(x_2) &= b_0 + b_1(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1) \\ &= f_0 + \cancel{b_1(x_2 - x_0)} + \left( \frac{f_2 - f_1}{\cancel{x_2 - x_1}} - \cancel{b_1} \right) \cancel{(x_2 - x_1)} \end{aligned}$$

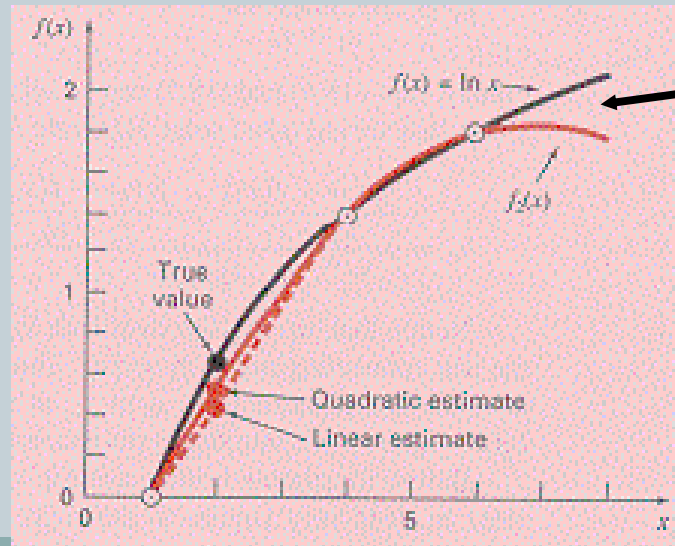
$$\begin{aligned} &= f_0 + b_1(x_1 - x_0) + f_2 - f_1 \\ &= f_0 + \left( \frac{f_1 - f_0}{\cancel{x_1 - x_0}} \right) \cancel{(x_1 - x_0)} + f_2 - f_1 \\ &= f_2 \end{aligned}$$

# Example: $\ln(x)$

39

- Interpolation of  $\ln(2)$ : given  $\ln(1)$ ;  $\ln(4)$  and  $\ln(6)$ 
  - Data points:  $\{(1,0), (4,1.3863), (6,1.79176)\}$
  - Linear Interpolation:  $0 + \{(1.3863-0)/(4-1)\}(x-1) = 0.4621(x-1)$
  - Quadratic Interpolation:  $0.4621(x-1) + \{(0.20273-0.4621)/(6-1)\}(x-4)$   
 $= 0.4621(x-1) - 0.051874(x-1)(x-4)$

corrected



Note the divergence for values outside of the data range.

## Example: $\ln(x)$

40

- Quadratic interpolation catches some of the curvature
- Improves the result somewhat
- Not always a good idea: see later...



# Calculating the Divided-Differences

41

- A *divided-difference* table can easily be constructed incrementally.
- Consider the function  $\ln(x)$ .

| x | ln(x)           |
|---|-----------------|
| 1 | <b>0.000000</b> |
| 2 | 0.693147        |
| 3 | 1.098612        |
| 4 | 1.386294        |
| 5 | 1.609438        |
| 6 | 1.791759        |
| 7 | 1.945910        |
| 8 | 2.079442        |
| x | ln(x)           |

# Calculating the Divided-Differences

42

| x | ln(x)           | f[l,l+1]        |
|---|-----------------|-----------------|
| 1 | <b>0.000000</b> |                 |
| 2 | 0.693147        | <b>0.693147</b> |
| 3 | 1.098612        | 0.405465        |
| 4 | 1.386294        | 0.287682        |
| 5 | 1.609438        | 0.223144        |
| 6 | 1.791759        | 0.182322        |
| 7 | 1.945910        | 0.154151        |
| 8 | 2.079442        | 0.133531        |
| x | ln(x)           | b10-b9)/(A10-A9 |

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

# Calculating the Divided-Differences

43

| x | ln(x)           | f[i,i+1]                        |                  |
|---|-----------------|---------------------------------|------------------|
| 1 | <b>0.000000</b> |                                 |                  |
| 2 | 0.693147        | <b>0.693147</b>                 |                  |
| 3 | 1.098612        | 0.405465                        | <b>-0.143841</b> |
| 4 | 1.386294        | 0.287682                        | -0.058892        |
| 5 | 1.609438        | 0.223144                        | -0.032269        |
| 6 | 1.791759        | 0.182322                        | -0.020411        |
| 7 | 1.945910        | 0.154151                        | -0.014085        |
| 8 | 2.079442        | 0.133531                        | -0.010310        |
| x | ln(x)           | b10-b9)/(A10-A9,c10-c9)/(a10-a8 |                  |

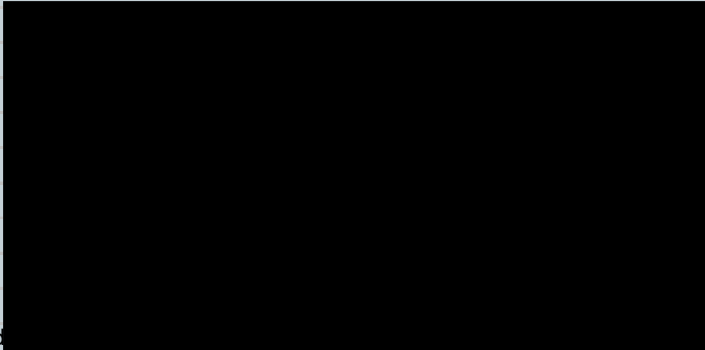
$$f[i+1,i+2] = \frac{f[i+1,i+2] - f[i,i+1]}{(x_{i+2} - x_i)}$$

# Calculating the Divided-Differences

44

$$f[i, \dots, i+3] = \frac{f[i+1, i+2, i+3] - f[i, i+1, i+2]}{(x_{i+3} - x_i)}$$

| x | ln(x)           | f[l,l+1]   |                  |                 |
|---|-----------------|--|------------------|-----------------|
| 1 | <b>0.000000</b> |  |                  |                 |
| 2 | 0.693147        | <b>0.693147</b>                                  |                  |                 |
| 3 | 1.098612        | 0.405465   | <b>-0.143841</b> |                 |
| 4 | 1.386294        | 0.287682   | -0.058892        | <b>0.028317</b> |
| 5 | 1.609438        | 0.223144   | -0.032269        | 0.008874        |
| 6 | 1.791759        | 0.182322   | -0.020411        | 0.003953        |
| 7 | 1.945910        | 0.154151   | -0.014085        | 0.002109        |
| 8 | 2.079442        | 0.133531   | -0.010310        | 0.001259        |
| x | ln(x)           | b10-b9)/(A10-A9;c10-c9)/(a10-a8;110-d9)/(a10-a7; |                  |                 |



# Calculating the Divided-Differences

45

$$f[i, \dots, i+4] = \frac{f[i+1, \dots, i+4] - f[i, \dots, i+3]}{(x_{i+4} - x_i)}$$

| x | ln(x)           | f[l,l+1]         |                  |                  |                  |
|---|-----------------|------------------|------------------|------------------|------------------|
| 1 | <b>0.000000</b> |                  |                  |                  |                  |
| 2 | 0.693147        | <b>0.693147</b>  |                  |                  |                  |
| 3 | 1.098612        | 0.405465         | <b>-0.143841</b> |                  |                  |
| 4 | 1.386294        | 0.287682         | -0.058892        | <b>0.028317</b>  |                  |
| 5 | 1.609438        | 0.223144         | -0.032269        | 0.008874         | <b>-0.004861</b> |
| 6 | 1.791759        | 0.182322         | -0.020411        | 0.003953         | -0.001230        |
| 7 | 1.945910        | 0.154151         | -0.014085        | 0.002109         | -0.000461        |
| 8 | 2.079442        | 0.133531         | -0.010310        | 0.001259         | -0.000212        |
| x | ln(x)           | b10-b9)/(A10-A9) | c10-c9)/(a10-a8) | d10-d9)/(a10-a7) | e10-e9)/(a10-a6) |

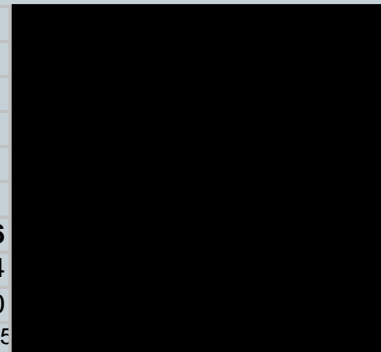


# Calculating the Divided-Differences

46

$$f[i, \dots, i+5] = \frac{f[i+1, \dots, i+5] - f[i, \dots, i+4]}{(x_{i+5} - x_i)}$$

| x | ln(x)           | f[i, i+1]  |                  |                 |                  |                 |
|---|-----------------|--|------------------|-----------------|------------------|-----------------|
| 1 | <b>0.000000</b> |  |                  |                 |                  |                 |
| 2 | 0.693147        | <b>0.693147</b>  |                  |                 |                  |                 |
| 3 | 1.098612        | 0.405465   | <b>-0.143841</b> |                 |                  |                 |
| 4 | 1.386294        | 0.287682   | -0.058892        | <b>0.028317</b> |                  |                 |
| 5 | 1.609438        | 0.223144   | -0.032269        | 0.008874        | <b>-0.004861</b> |                 |
| 6 | 1.791759        | 0.182322   | -0.020411        | 0.003953        | -0.001230        | <b>0.000726</b> |
| 7 | 1.945910        | 0.154151   | -0.014085        | 0.002109        | -0.000461        | 0.000154        |
| 8 | 2.079442        | 0.133531   | -0.010310        | 0.001259        | -0.000212        | 0.000050        |
| x | ln(x)           | b10-b9)/(A10-A9)(c10-c9)/(a10-a8)(d10-d9)/(a10-a7)(d10-d9)/(a10-a6)(e10-e9)/(a10-a5) |                  |                 |                  |                 |



# Calculating the Divided-Differences

47

$$f[i, \dots, i+6] = \frac{f[i+1, \dots, i+6] - f[i, \dots, i+5]}{(x_{i+6} - x_i)}$$

|          |                 |                                   |                                   |                                   |                                   |                                   |                                   |
|----------|-----------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| <b>x</b> | <b>ln(x)</b>    | <b>f[i, i+1]</b>                  |                                   |                                   |                                   |                                   |                                   |
| 1        | <b>0.000000</b> |                                   |                                   |                                   |                                   |                                   |                                   |
| 2        | 0.693147        | <b>0.693147</b>                   |                                   |                                   |                                   |                                   |                                   |
| 3        | 1.098612        | 0.405465                          | <b>-0.143841</b>                  |                                   |                                   |                                   |                                   |
| 4        | 1.386294        | 0.287682                          | -0.058892                         | <b>0.028317</b>                   |                                   |                                   |                                   |
| 5        | 1.609438        | 0.223144                          | -0.032269                         | 0.008874                          | <b>-0.004861</b>                  |                                   |                                   |
| 6        | 1.791759        | 0.182322                          | -0.020411                         | 0.003953                          | -0.001230                         | <b>0.000726</b>                   |                                   |
| 7        | 1.945910        | 0.154151                          | -0.014085                         | 0.002109                          | -0.000461                         | 0.000154                          | <b>-0.000095</b>                  |
| 8        | 2.079442        | 0.133531                          | -0.010310                         | 0.001259                          | -0.000212                         | 0.000050                          | -0.000017                         |
| x        | ln(x)           | $\frac{b_{10}-b_9}{(a_{10}-a_9)}$ | $\frac{c_{10}-c_9}{(a_{10}-a_8)}$ | $\frac{d_{10}-d_9}{(a_{10}-a_7)}$ | $\frac{e_{10}-e_9}{(a_{10}-a_6)}$ | $\frac{f_{10}-f_9}{(a_{10}-a_5)}$ | $\frac{g_{10}-g_9}{(a_{10}-a_4)}$ |



# Calculating the Divided-Differences

48

- Finally, we can calculate the last coefficient.

$$f[i, \dots, i+7] = \frac{f[i+1, \dots, i+7] - f[i, \dots, i+6]}{(x_{i+7} - x_i)}$$

| x | ln(x)           | f[l,l+1]                          |                                   |                                   |                                   |                                   |                                   |                                   | f[l,l+1,...,l+7] |
|---|-----------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|------------------|
| 1 | <b>0.000000</b> |                                   |                                   |                                   |                                   |                                   |                                   |                                   |                  |
| 2 | 0.693147        | <b>0.693147</b>                   |                                   |                                   |                                   |                                   |                                   |                                   |                  |
| 3 | 1.098612        | 0.405465                          | <b>-0.143841</b>                  |                                   |                                   |                                   |                                   |                                   |                  |
| 4 | 1.386294        | 0.287682                          | -0.058892                         | <b>0.028317</b>                   |                                   |                                   |                                   |                                   |                  |
| 5 | 1.609438        | 0.223144                          | -0.032269                         | 0.008874                          | <b>-0.004861</b>                  |                                   |                                   |                                   |                  |
| 6 | 1.791759        | 0.182322                          | -0.020411                         | 0.003953                          | -0.001230                         | <b>0.000726</b>                   |                                   |                                   |                  |
| 7 | 1.945910        | 0.154151                          | -0.014085                         | 0.002109                          | -0.000461                         | 0.000154                          | <b>-0.000095</b>                  |                                   |                  |
| 8 | 2.079442        | 0.133531                          | -0.010310                         | 0.001259                          | -0.000212                         | 0.000050                          | -0.000017                         | <b>0.000011</b>                   |                  |
| x | ln(x)           | $\frac{b_{10}-b_9}{(a_{10}-a_9)}$ | $\frac{c_{10}-c_9}{(a_{10}-a_8)}$ | $\frac{d_{10}-d_9}{(a_{10}-a_7)}$ | $\frac{e_{10}-e_9}{(a_{10}-a_6)}$ | $\frac{f_{10}-f_9}{(a_{10}-a_5)}$ | $\frac{g_{10}-g_9}{(a_{10}-a_4)}$ | $\frac{h_{10}-h_9}{(a_{10}-a_3)}$ |                  |



# Calculating the Divided-Differences

49

- All of the coefficients for the resulting polynomial are in bold.

| $x$ | $\ln(x)$        | $f[l, l+1]$                 |                             |                             |                             |                             |                             | $f[l, l+1, \dots, l+7]$     | $b$ |
|-----|-----------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----|
| 1   | <b>0.000000</b> |                             |                             |                             |                             |                             |                             |                             |     |
| 2   | 0.693147        | <b>0.693147</b>             |                             |                             |                             |                             |                             |                             |     |
| 3   | 1.098612        | 0.405465                    | <b>-0.143841</b>            |                             |                             |                             |                             |                             |     |
| 4   | 1.386294        | 0.287682                    | -0.058892                   | <b>0.028317</b>             |                             |                             |                             |                             |     |
| 5   | 1.609438        | 0.223144                    | -0.032269                   | 0.008874                    | <b>-0.004861</b>            |                             |                             |                             |     |
| 6   | 1.791759        | 0.182322                    | -0.020411                   | 0.003953                    | -0.001230                   | <b>0.000726</b>             |                             |                             |     |
| 7   | 1.945910        | 0.154151                    | -0.014085                   | 0.002109                    | -0.000461                   | 0.000154                    | <b>-0.000095</b>            |                             |     |
| 8   | 2.079442        | 0.133531                    | -0.010310                   | 0.001259                    | -0.000212                   | 0.000050                    | -0.000017                   | <b>0.000011</b>             |     |
| $x$ | $\ln(x)$        | $(b_1 - b_0) / (a_1 - a_0)$ | $(c_1 - c_0) / (a_1 - a_0)$ | $(d_1 - d_0) / (a_1 - a_0)$ | $(e_1 - e_0) / (a_1 - a_0)$ | $(f_1 - f_0) / (a_1 - a_0)$ | $(g_1 - g_0) / (a_1 - a_0)$ | $(h_1 - h_0) / (a_1 - a_0)$ |     |

# Polynomial Form for Divided-Differences

50

- The resulting polynomial comes from the divided-differences and the corresponding product terms:

$$\begin{aligned} p_7(x) = & 0 \\ & +0.693(x-1) \\ & -0.144(x-1)(x-2) \\ & +0.28(x-1)(x-2)(x-3) \\ & -0.0049(x-1)(x-2)(x-3)(x-4) \\ & +7.26 \cdot 10^{-4}(x-1)(x-2)(x-3)(x-4)(x-5) \\ & -9.5 \cdot 10^{-5}(x-1)(x-2)(x-3)(x-4)(x-5)(x-6) \\ & +1.1 \cdot 10^{-5}(x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7) \end{aligned}$$

# Many polynomials

51

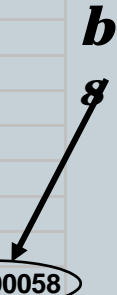
- Note, that the order of the numbers  $(x_i, y_i)$ 's only matters when writing the polynomial down.
  - The first column represents the set of linear splines between two adjacent data points.
  - The second column gives us quadratics thru three adjacent points.
  - Etc.

# Adding an Additional Data Point

52

- Adding an additional data point, simply adds an additional term to the existing polynomial.
- Hence, only  $n$  additional divided-differences need to be calculated for the  $n+1^{\text{st}}$  data point.

| $x$              | $\ln(x)$         | $f[l, l+1]$      |                   |                  |                   |                  |                   | $f[l, l+1, \dots, l+7]$ |
|------------------|------------------|------------------|-------------------|------------------|-------------------|------------------|-------------------|-------------------------|
| 1.0000000        | <b>0.0000000</b> |                  |                   |                  |                   |                  |                   |                         |
| 2.0000000        | 0.6931472        | <b>0.6931472</b> |                   |                  |                   |                  |                   |                         |
| 3.0000000        | 1.0986123        | 0.4054651        | <b>-0.1438410</b> |                  |                   |                  |                   |                         |
| 4.0000000        | 1.3862944        | 0.2876821        | -0.0588915        | <b>0.0283165</b> |                   |                  |                   |                         |
| 5.0000000        | 1.6094379        | 0.2231436        | -0.0322693        | 0.0088741        | <b>-0.0048606</b> |                  |                   |                         |
| 6.0000000        | 1.7917595        | 0.1823216        | -0.0204110        | 0.0039528        | -0.0012303        | <b>0.0007261</b> |                   |                         |
| 7.0000000        | 1.9459101        | 0.1541507        | -0.0140854        | 0.0021085        | -0.0004611        | 0.0001539        | <b>-0.0000954</b> |                         |
| 8.0000000        | 2.0794415        | 0.1335314        | -0.0103096        | 0.0012586        | -0.0002125        | 0.0000497        | -0.0000174        | <b>0.0000111</b>        |
| <b>1.5000000</b> | 0.4054651        | 0.2575348        | -0.0225461        | 0.0027192        | -0.0004173        | 0.0000819        | -0.0000215        | 0.0000082               |
|                  |                  |                  |                   |                  |                   |                  |                   | <b>-0.0000058</b>       |



# Adding More Data Points

53

- Quadratic interpolation:
  - does linear interpolation
  - Then add higher-order correction to catch the curvature
- Cubic, ...
- Consider the case where the data points are organized such the the first two are the endpoints, the next point is the mid-point, followed by successive mid-points of the half-intervals.
  - Worksheet:  $f(x)=x^2$  from -1 to 3.

# Uniqueness

54

- Suppose that two polynomials of degree  $n$  (or less) existed that interpolated to the  $n+1$  data points.
- Subtracting these two polynomials from each other also leads to a polynomial of at most  $n$  degree.

$$r_n(x) = p_n(x) - q_n(x)$$

# Uniqueness

55

- Since  $p$  and  $q$  both interpolate the  $n+1$  data points,
- This polynomial  $r$ , has at least  $n+1$  roots!!!
- This can not be! A polynomial of degree- $n$  can only have at most  $n$  roots.

- Therefore,  $\mathbf{r(x) \equiv 0}$

$$p_n(x) = a_n \prod_{i=1}^n (x - r_i)$$

$$p_{n+1}(x) = a_{n+1} \prod_{i=1}^{n+1} (x - r_i)$$

# Example

56

- Suppose  $f$  was a polynomial of degree  $m$ , where  $m < n$ .
- Ex:  $f(x) = 3x - 2$
- We have evaluations of  $f(x)$  at five locations:  $(-2, -8)$ ,  $(-1, -5)$ ,  $(0, -2)$ ,  $(1, 1)$ ,  $(2, 4)$



# Error

57

- Define the error term as:

$$\varepsilon_n(x) = f(x) - p_n(x)$$

- If  $f(x)$  is an  $n^{\text{th}}$  order polynomial  $p_n(x)$  is of course exact.
- Otherwise, since there is a perfect match at  $x_0, x_1, \dots, x_n$
- This function has at least  $n+1$  roots at the interpolation points.

$$\therefore \varepsilon_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n)h(x)$$

# Interpolation Errors

58

$$\varepsilon_n(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^n (x - x_i)$$

$$x \in [a, b], \xi \in (a, b)$$

- Proof is in the book.
- Intuitively, the first  $n+1$  terms of the Taylor Series is also an  $n^{\text{th}}$  degree polynomial.

# Interpolation Errors

59

- Use the point  $x$ , to expand the polynomial.

$$x \notin \{x_0, x_1, \dots, x_n\}$$

$$\varepsilon_n(x) = f(x) - p_n(x) = f[x_0, x_1, \dots, x_n, x] \prod_{i=0}^n (x - x_i)$$

- Point is, we can take an arbitrary point  $x$ , and create an  $(n+1)^{\text{th}}$  polynomial that goes thru the point  $x$ .

# Interpolation Errors

60

- Combining the last two statements, we can also get a feel for what  $f[x_0, x_1, \dots, x_n]$  represent.

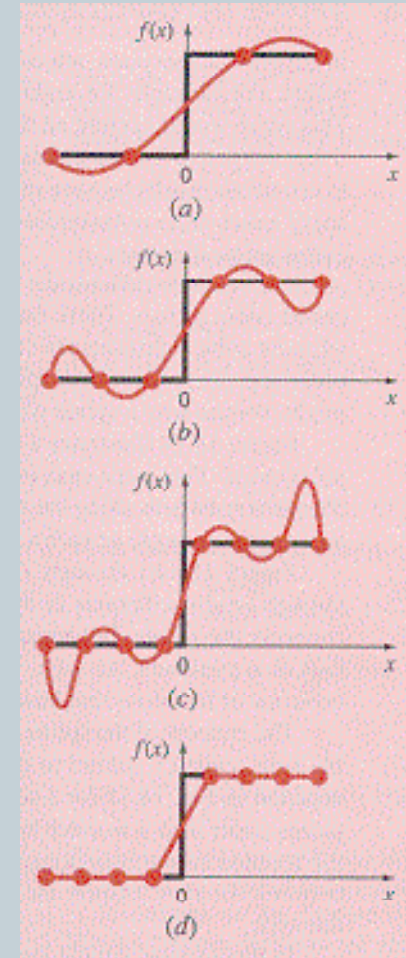
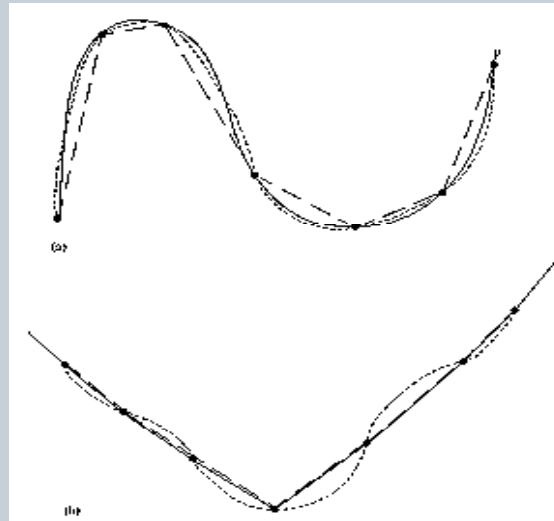
$$f[x_0, x_1, \dots, x_n] = \frac{1}{n!} f^{(n)}(\xi)$$

- Corollary 1 in book – If  $f(x)$  is a polynomial of degree  $m < n$ , then all  $(m+1)^{\text{st}}$  divided differences and higher are zero.

# Problems with Interpolation

61

- Is it always a good idea to use higher and higher order polynomials?
- Certainly not: 3-4 points usually good: 5-6 ok:
- See tendency of polynomial to “wobble”
- Particularly for sharp edges: see figures



# Chebyshev nodes

62

- Equally distributed points may not be the optimal solution.
- If you could select the  $x_i$ 's, what would they be?
- Want to minimize the  $\prod_{i=0}^n (x - x_i)$  term.
- These are the Chebyshev nodes.
  - For  $x=-1$  to  $1$ :

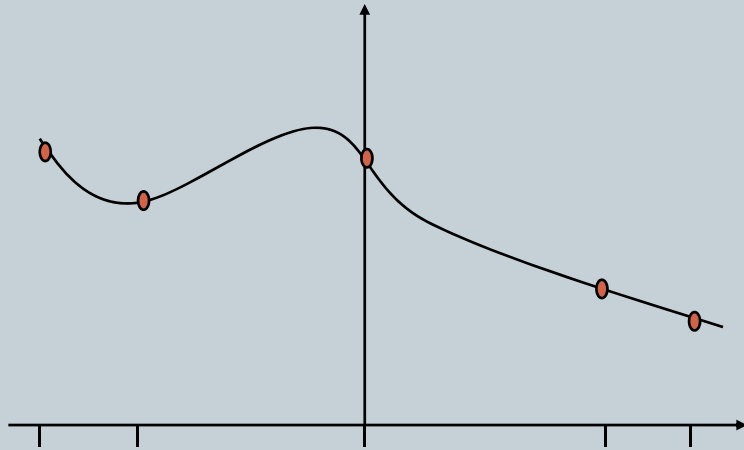
$$\prod_{i=0}^n (x - x_i)$$

$$x_i = \cos \left[ \left( \frac{i}{n} \right) \pi \right], \quad (0 \leq i \leq n)$$

# Chebyshev nodes

63

- Let's look at these for  $n=4$ .
- Spreads the points out in the center.



$$x_0 = \cos \left[ \left( \frac{0}{4} \right) \pi \right] = 1$$

$$x_1 = \cos \left[ \left( \frac{1}{4} \right) \pi \right] = \frac{\sqrt{2}}{2} \approx 0.707$$

$$x_2 = \cos \left[ \left( \frac{2}{4} \right) \pi \right] = 0$$

$$x_3 = \cos \left[ \left( \frac{3}{4} \right) \pi \right] = -\frac{\sqrt{2}}{2} \approx -0.707$$

$$x_4 = \cos \left[ \left( \frac{4}{4} \right) \pi \right] = -1$$

# Polynomial Interpolation in Two-Dimensions

64

- Consider the case in higher-dimensions.



# Finding the Inverse of a Function

65

- What if I am after the inverse of the function  $f(x)$ ?
  - For example  $\arccos(x)$ .
- Simply reverse the role of the  $x_i$  and the  $f_i$ .



## Section B

- Solving Non-Linear Equations
- (Root Finding)

# Root finding Methods



- What are root finding methods?
- Methods for determining a solution of an equation.
- Essentially finding a root of a function, that is, a zero of the function.

# Root finding Methods



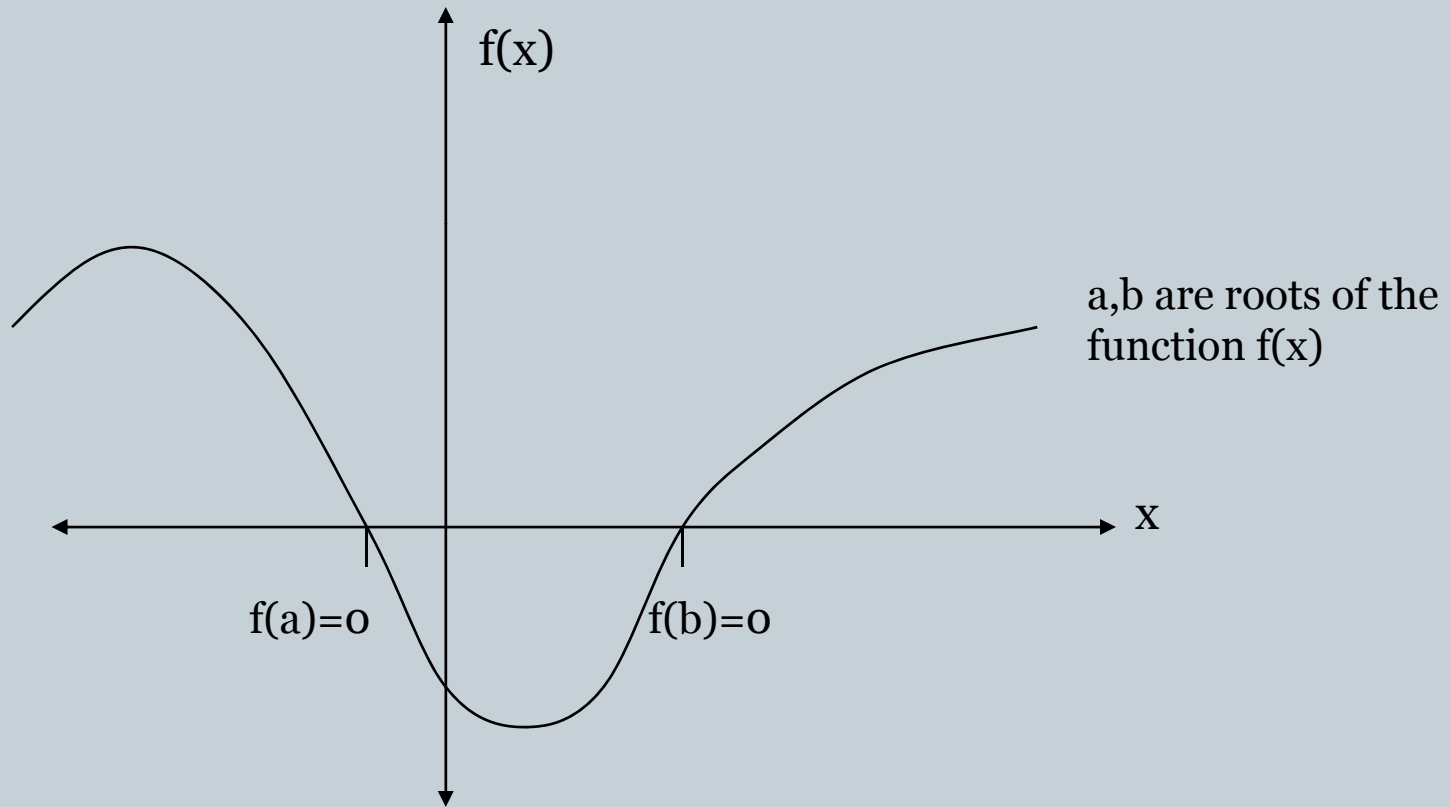
- Where are they used?
- Some applications for root finding are: systems of equilibrium, elliptical orbits, the van der Waals equation, and natural frequencies of spring systems.
- The problem of solving non-linear equations or sets of non-linear equations is a common problem in science, applied mathematics.

## Cont...



- The problem of solving non-linear equations or sets of non-linear equations is a common problem in science, applied mathematics.
- The goal is to solve  $f(x) = 0$ , for the function  $f(x)$ .
- The values of  $x$  which make  $f(x) = 0$  are the roots of the equation.

Cont...





- There are many methods for solving non-linear equations. The methods, which will be highlighted have the following properties:
  1. the function  $f(x)$  is expected to be continuous. If not the method may fail.
  2. the use of the algorithm requires that the solution be bounded.
  3. once the solution is bounded, it is refined to specified tolerance.

# Cont...



Four such methods are:

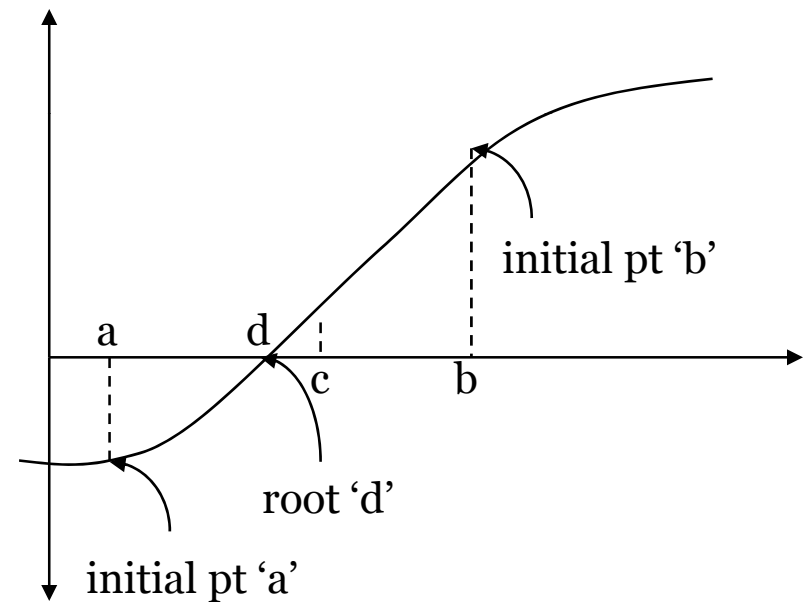
- Interval Halving (Bisection method)
- Regula Falsi (False position)
- Secant method
- Fixed point iteration
- Newton's method



# Bisection Method



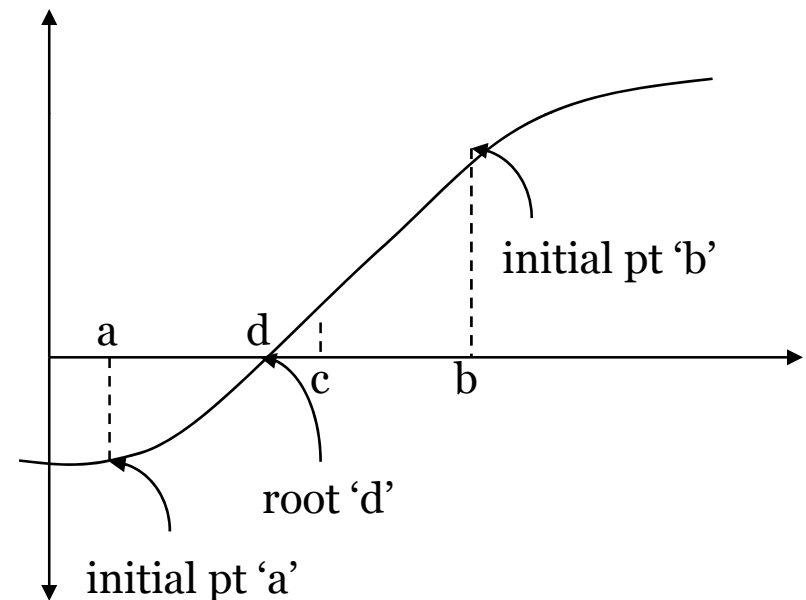
- It is the simplest root-finding algorithm.
- Requires previous knowledge of two initial guesses,  $a$  and  $b$ , so that  $f(a)$  and  $f(b)$  have opposite signs.



# Bisection Method



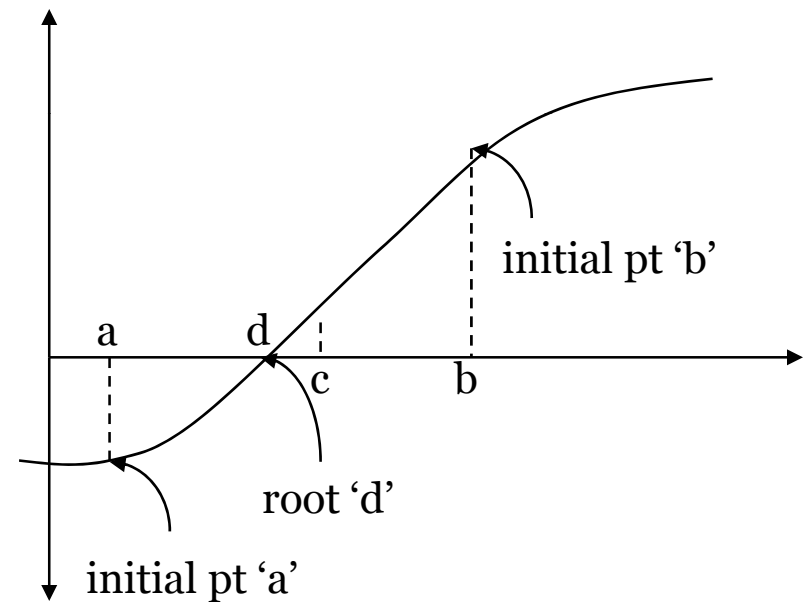
- Two estimates are chosen so that the solution is bracketed. ie  $f(a)$  and  $f(b)$  have opposite signs.
- In the diagram this  $f(a) < 0$  and  $f(b) > 0$ .
- **The root  $d$  lies between  $a$  and  $b$ !**



# Bisection Method



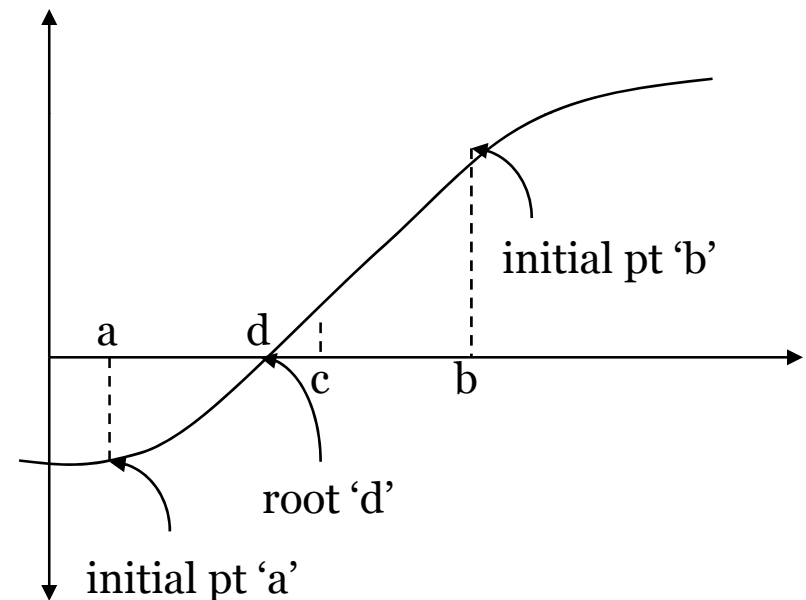
- **The root  $d$  must always be bracketed (be between  $a$  and  $b$ )!**
- Iterative method.



# Bisection Method



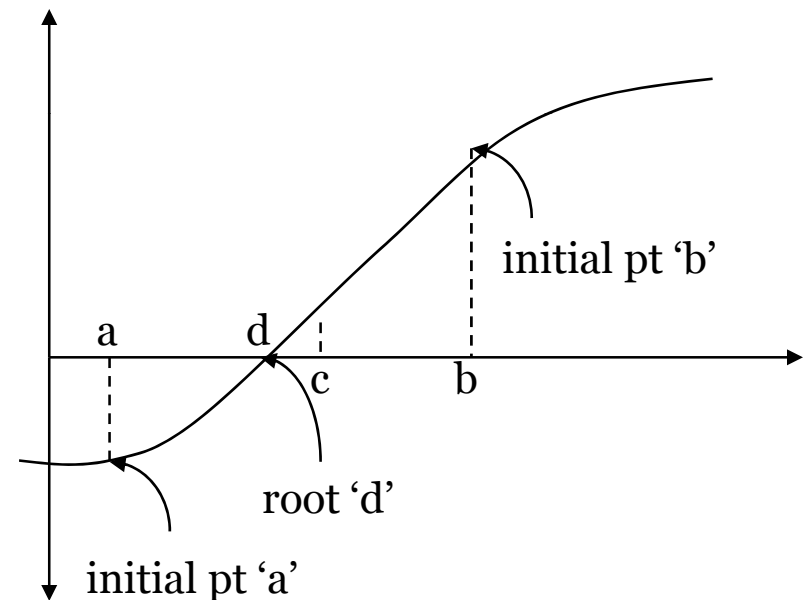
- The interval between  $a$  and  $b$  is halved by calculating the average of  $a$  and  $b$ .
- **The new point  $c = (a+b)/2$ .**
- This produces two possible intervals:  $a < x < c$  and  $c < x < b$ .



# Bisection Method



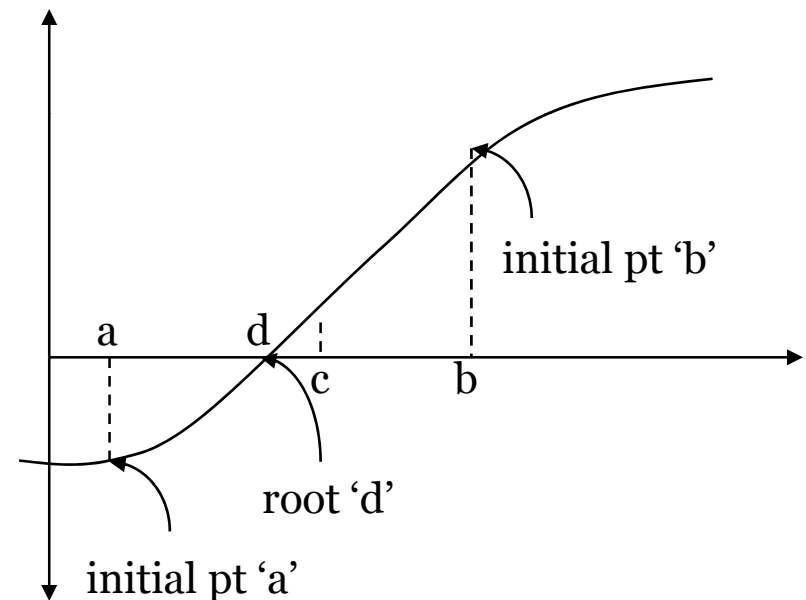
- This produces are two possible intervals:  $a < x < c$  and  $c < x < b$ .
- If  $f(c) > 0$ , then  $x = d$  must be to the left of  $c$ : interval  $a < x < c$ .
- If  $f(c) < 0$ , then  $x = d$  must be to the right of  $c$ : interval  $c < x < b$ .



# Bisection Method



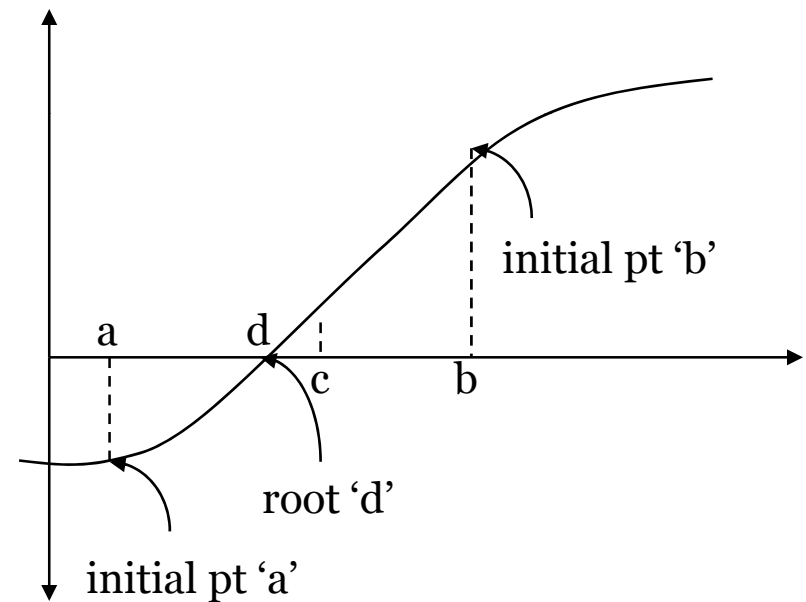
- If  $f(c) > 0$ , let  $a_{\text{new}} = a$  and  $b_{\text{new}} = c$  and repeat process.
- If  $f(c) < 0$ , let  $a_{\text{new}} = c$  and  $b_{\text{new}} = b$  and repeat process.
- **This reassignment ensures the root is always bracketed!!**



# Bisection Method



- This produces are two possible intervals:  $a < x < c$  and  $c < x < b$ .
- If  $f(c) > 0$ , then  $x = d$  must be to the left of  $c$ : interval  $a < x < c$ .
- If  $f(c) < 0$ , then  $x = d$  must be to the right of  $c$ : interval  $c < x < b$ .



# Bisection Method



$$c_i = (a_i + b_i) / 2$$

if  $f(c_i) > 0$ ;  $a_{i+1} = a$  and  $b_{i+1} = c$

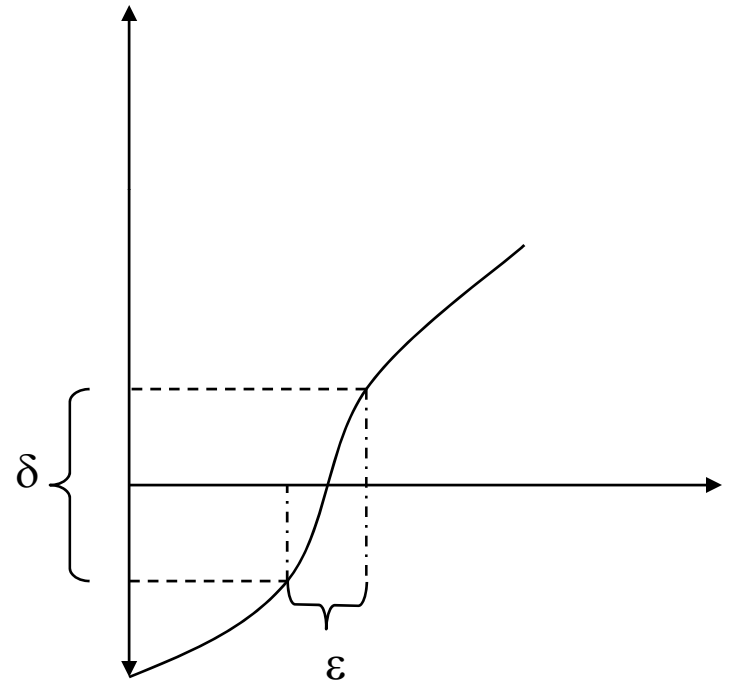
if  $f(c_i) < 0$ ;  $a_{i+1} = c$  and  $b_{i+1} = b$



# Bisection Method



- Bisection is an iterative process, where the initial interval is halved until the size of the interval decreases until it is below some predefined tolerance  $\varepsilon$ :  $|a-b| \geq \varepsilon$  or  $f(x)$  falls below a tolerance  $\delta$ :  $|f(c) - f(c-1)| \leq \delta$ .



# Bisection Method



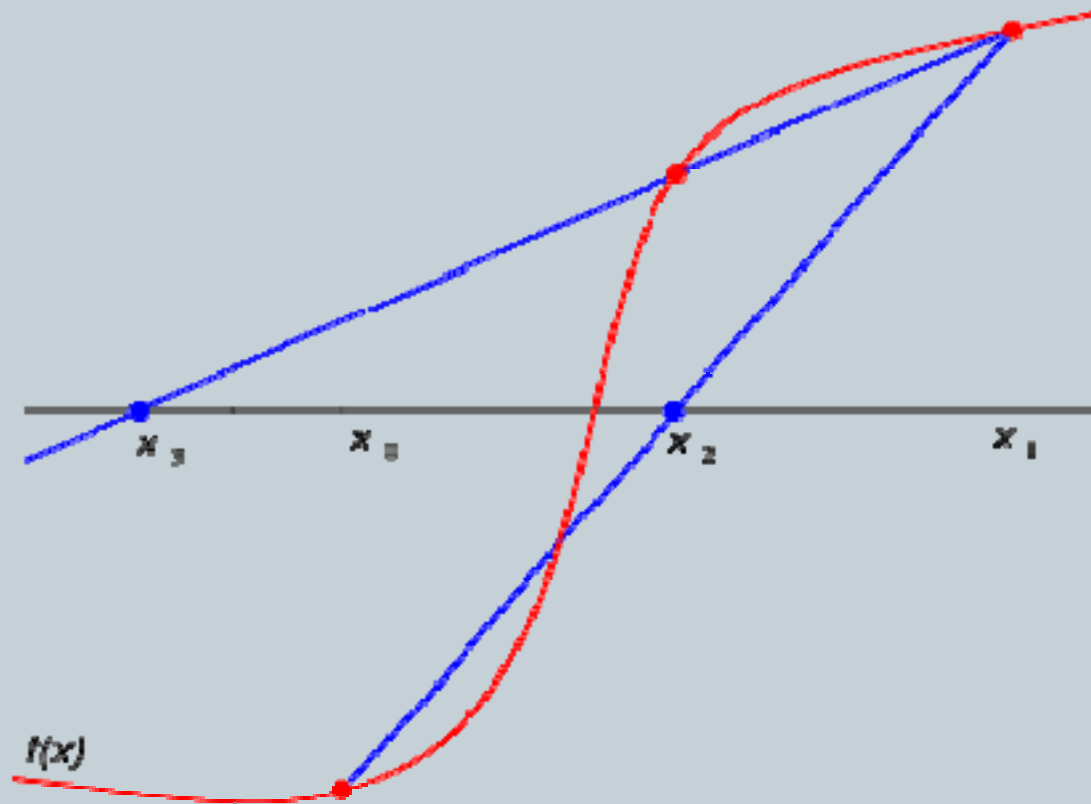
- Advantages
  1. Is guaranteed to work if  $f(x)$  is continuous and the root is bracketed.
  2. The number of iterations to get the root to specified tolerance is known in advance

# Bisection Method



- Disadvantages
  1. Slow convergence.
  2. Not applicable when there are multiple roots. Will only find one root at a time.

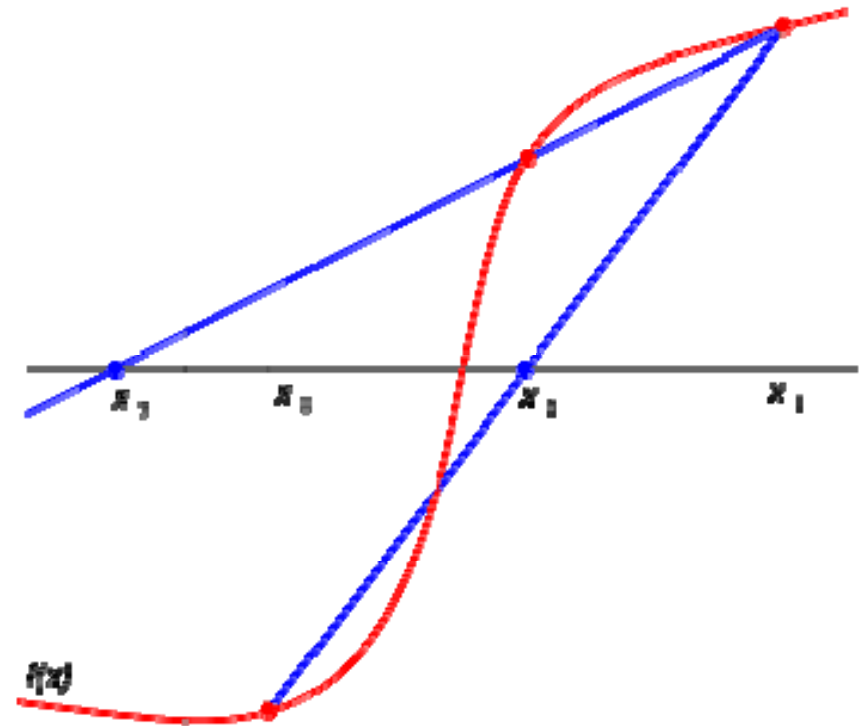
# Secant Method



# Overview of Secant Method



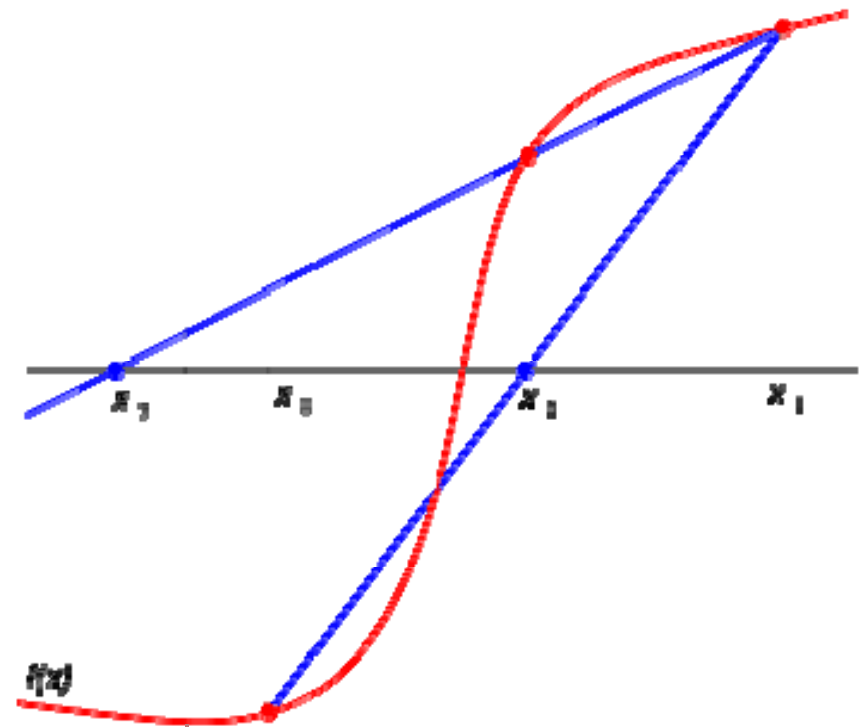
- Again to initial guesses are chosen.
- **However there is not requirement that the root is bracketed!**
- The method proceeds by drawing a line through the points to get a new point closer to the root.
- This is repeated until the root is found.



# Secant Method



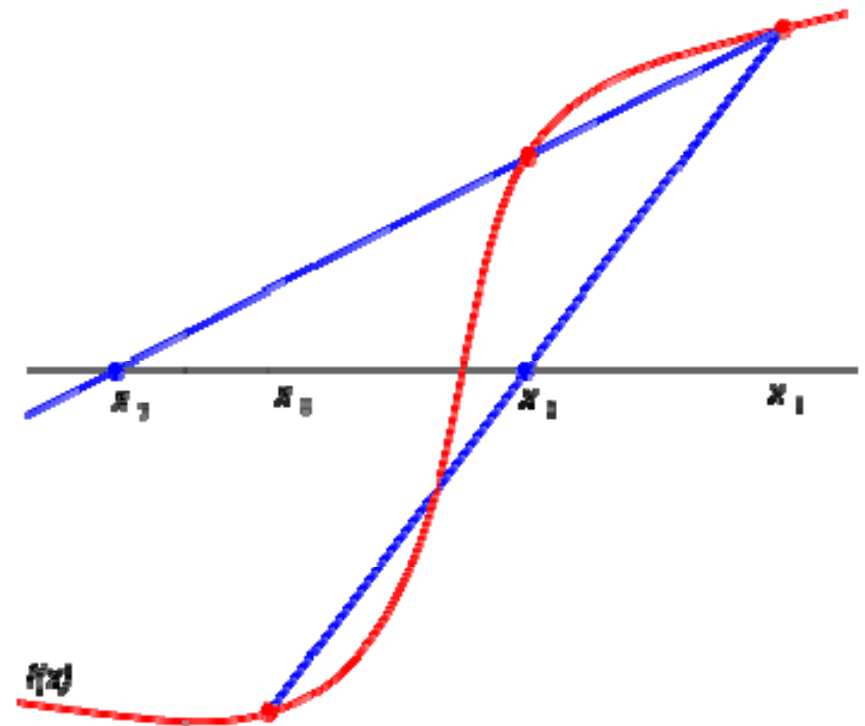
- First we find two points  $(x_0, x_1)$ , which are hopefully near the root (we may use the bisection method).
- A line is then drawn through the two points and we find where the line intercepts the x-axis,  $x_2$ .



# Secant Method



- If  $f(x)$  were truly linear, the straight line would intercept the  $x$ -axis at the root.
- However since it is not linear, the intercept is not at the root but it should be close to it.

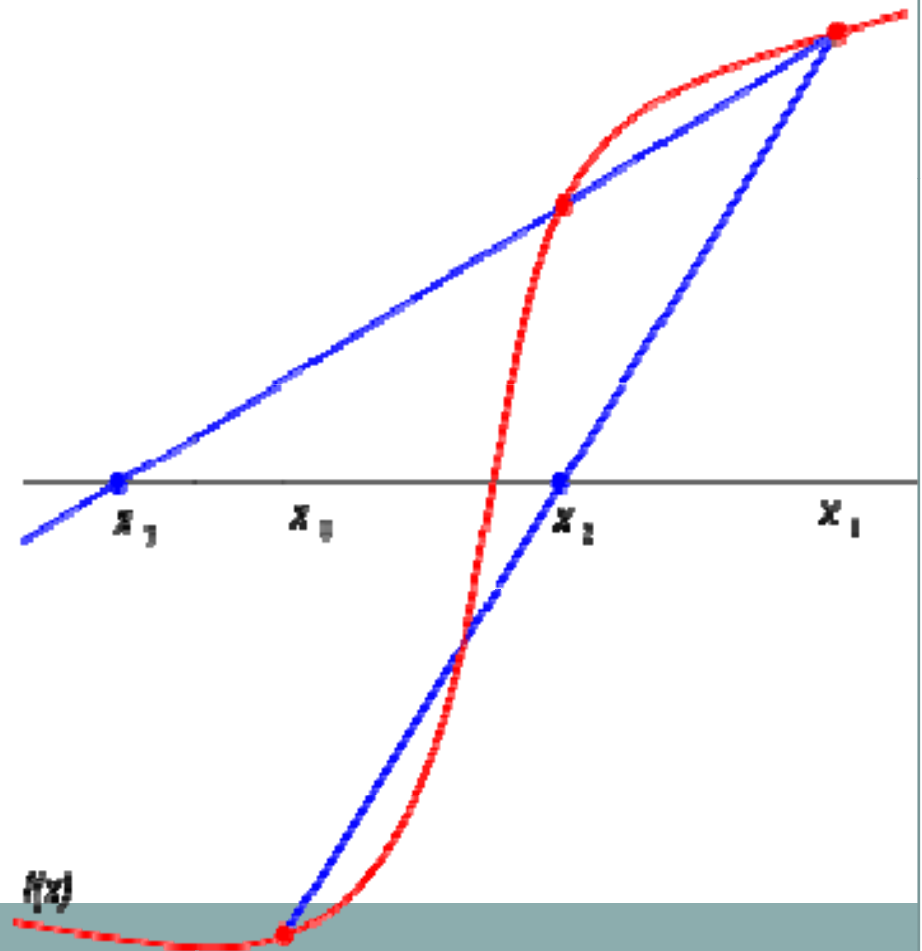
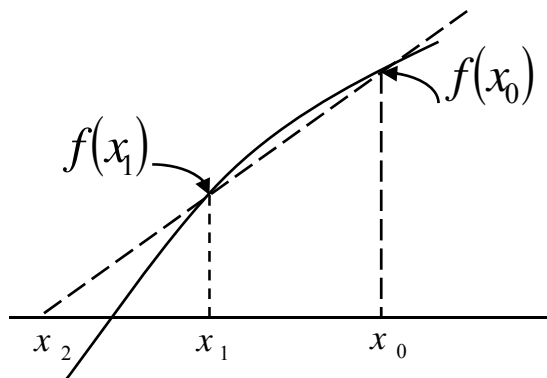


# Secant Method



- From similar triangles we can write that,

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





# Secant Method

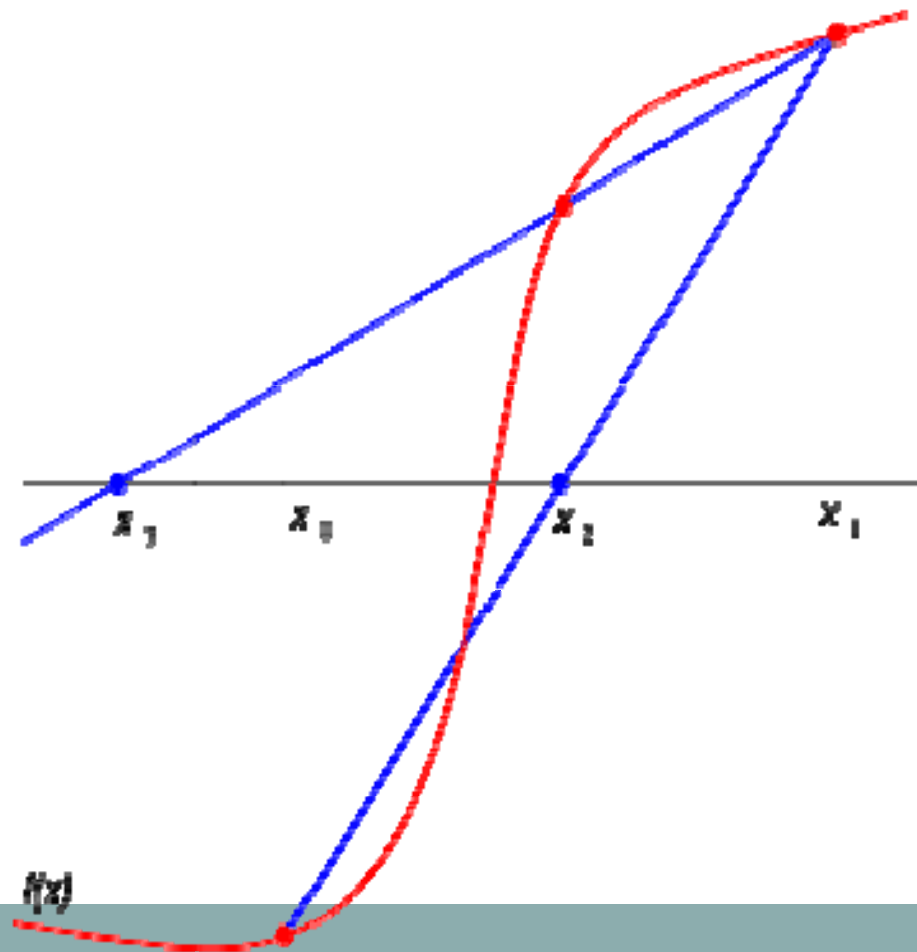


- From similar triangles we can write that,

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

- Solving for  $x_2$  we get:

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

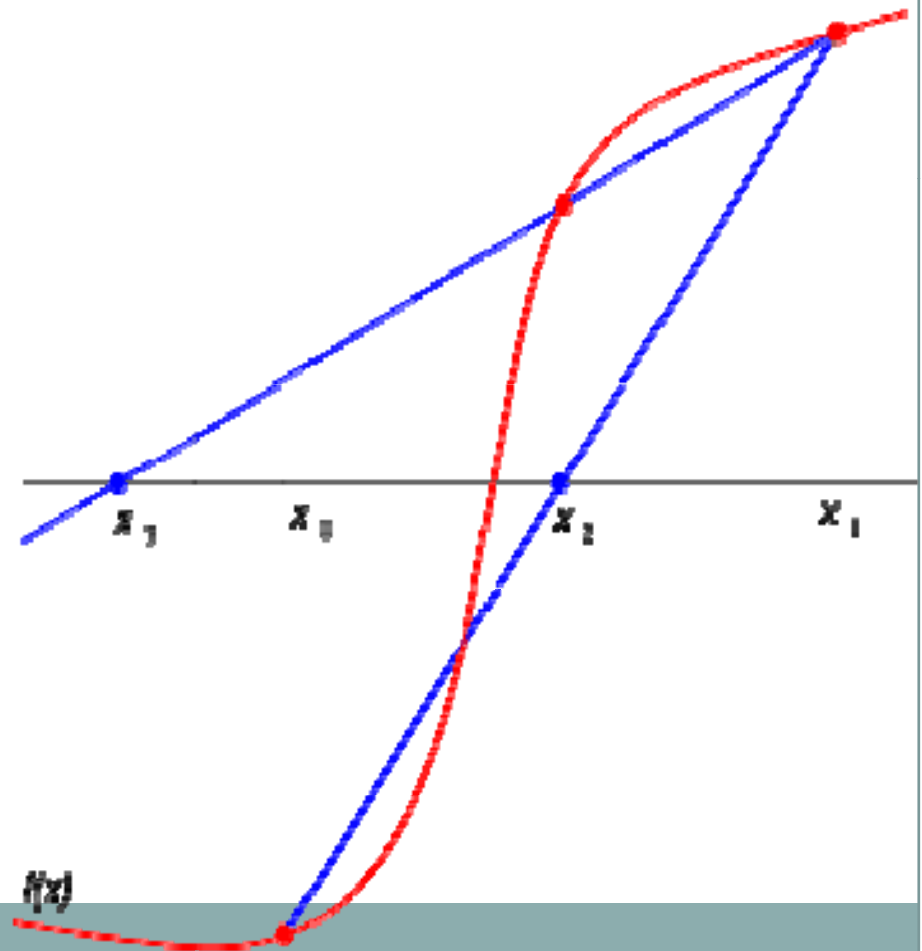


# Secant Method



- Iteratively this is written as:

$$x_{n+1} = x_n - f(x_n) \frac{(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$



# Algorithm



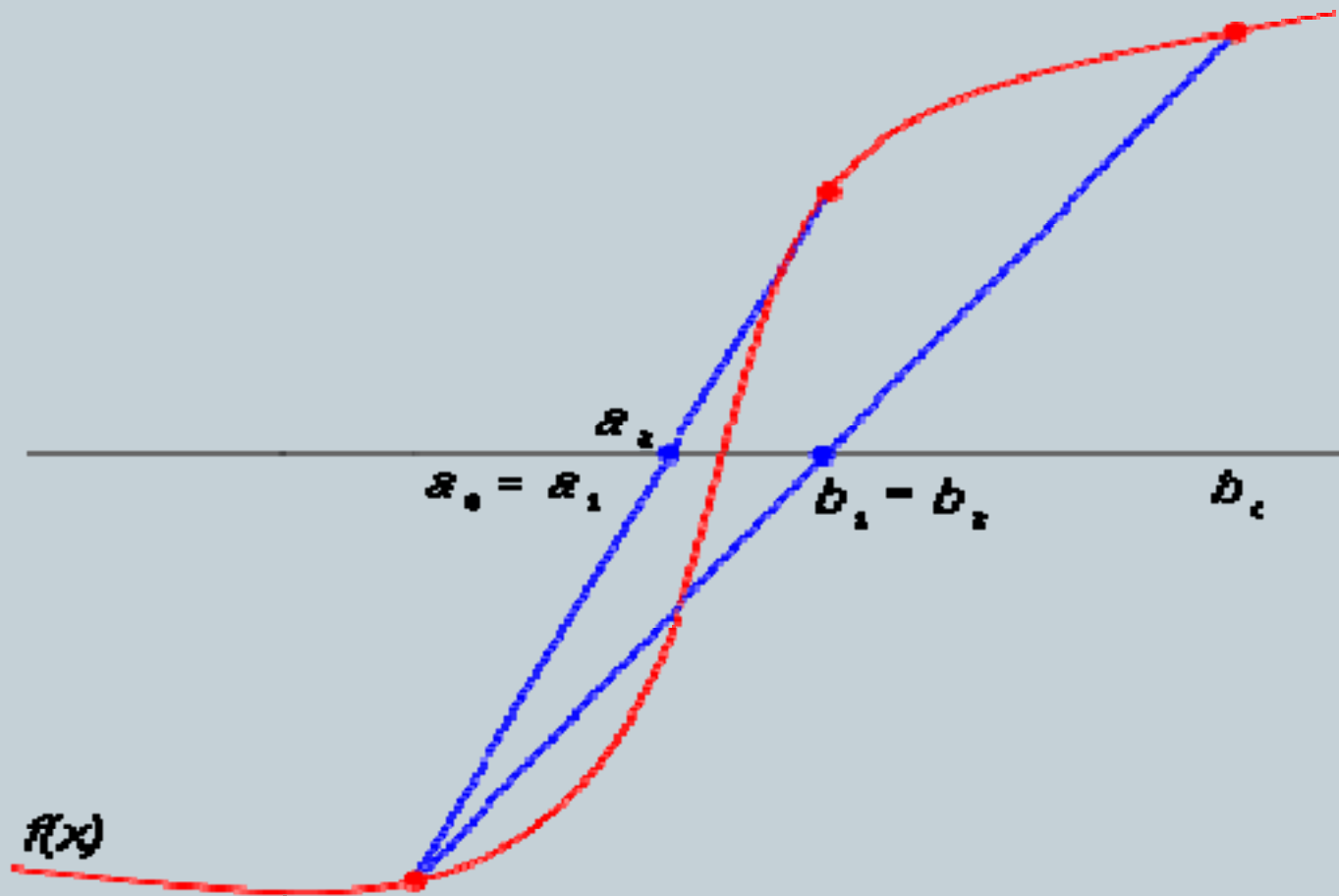
- Given two guesses  $x_0, x_1$  near the root,
- If  $|f(x_0)| < |f(x_1)|$  then
- Swap  $x_0$  and  $x_1$ .
- Repeat
- Set  $x_2 = x_1 - f(x_1) * \frac{x_0 - x_1}{f(x_0) - f(x_1)}$
- Set  $x_0 = x_1$
- Set  $x_1 = x_2$
- Until  $|f(x_2)| < \text{tolerance value}$ .

## Cont...



- Because the new point should be closer the root after the 2<sup>nd</sup> iteration we usually choose the last two points.
- After the first iteration there is only one new point. However  $x_1$  is chosen so that it is closer to root than  $x_0$ .
- **This is not a “hard and fast rule”!**

# False Position



# False Position



- The method of false position is seen as an improvement on the secant method.
- The method of false position avoids the problems of the secant method by ensuring that the root is bracketed between the two starting points and remains bracketing between successive pairs.

# False Position



- This technique is similar to the bisection method except that the next iterate is taken as the line of interception between the pair of  $x$ -values and the  $x$ -axis rather than at the midpoint.

# False Position

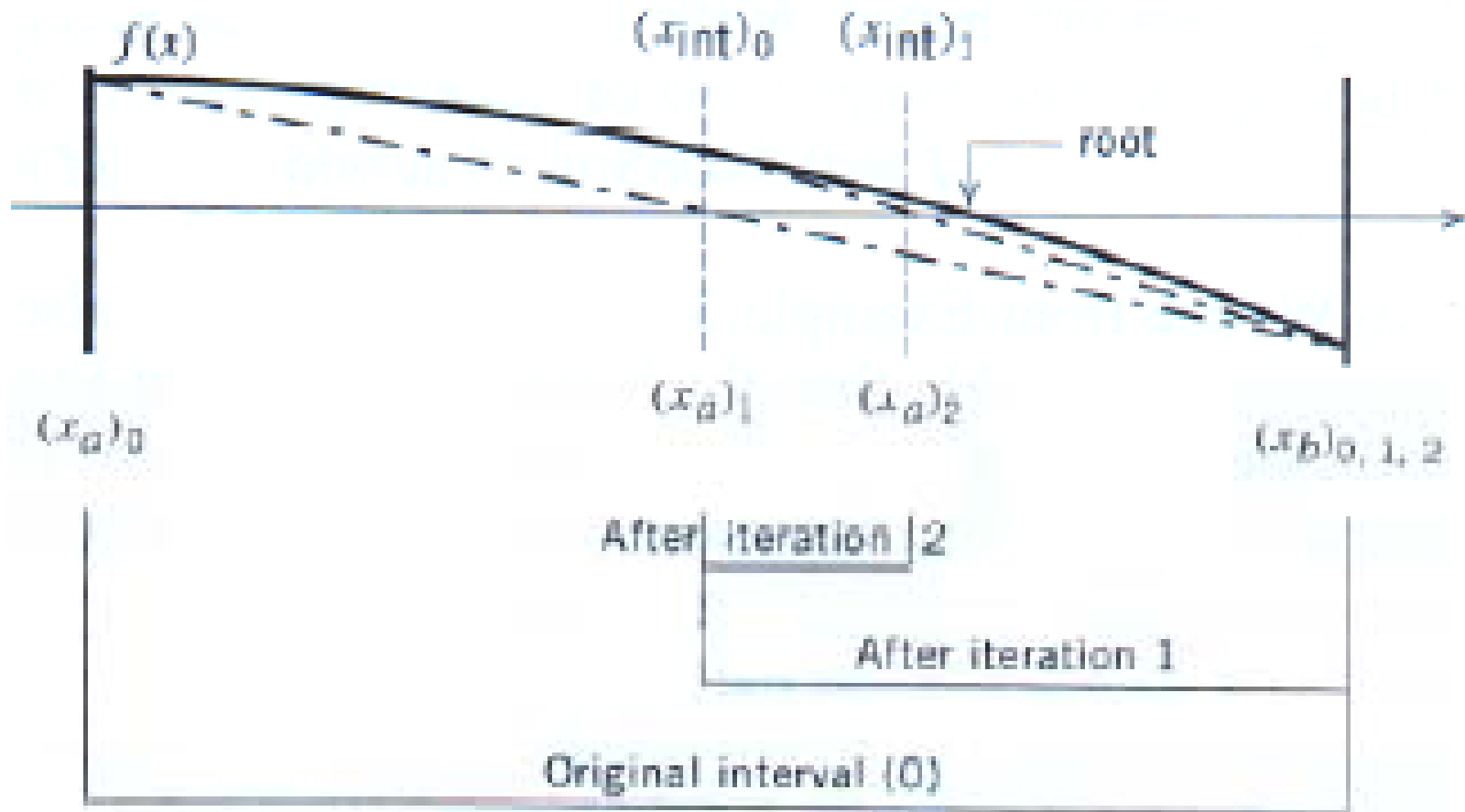


Illustration of the false-position method.



# Algorithm



- **Given two guesses  $x_0$ ,  $x_1$  that bracket the root,**
- Repeat
- Set  $x_2 = x_1 - f(x_1) * \frac{x_0 - x_1}{f(x_0) - f(x_1)}$
- If  $f(x_2)$  is of opposite sign to  $f(x_0)$  then
- Set  $x_1 = x_2$
- Else Set  $x_0 = x_1$
- End If
- Until  $|f(x_2)| < \text{tolerance value.}$

# Discussion of False Position Method



- This method achieves better convergence but a more complicated algorithm.
- May fail if the function is not continuous.

# Newton's Method



- The bisection method is useful up to a point.
- In order to get a good accuracy a large number of iterations must be carried out.
- A second inadequacy occurs when there are multiple roots to be found.

# Newton's Method



- The bisection method is useful up to a point.
- In order to get a good accuracy a large number of iterations must be carried out.
- A second inadequacy occurs when there are multiple roots to be found.
- Newton's method is a much better algorithm.

# Newton's Method

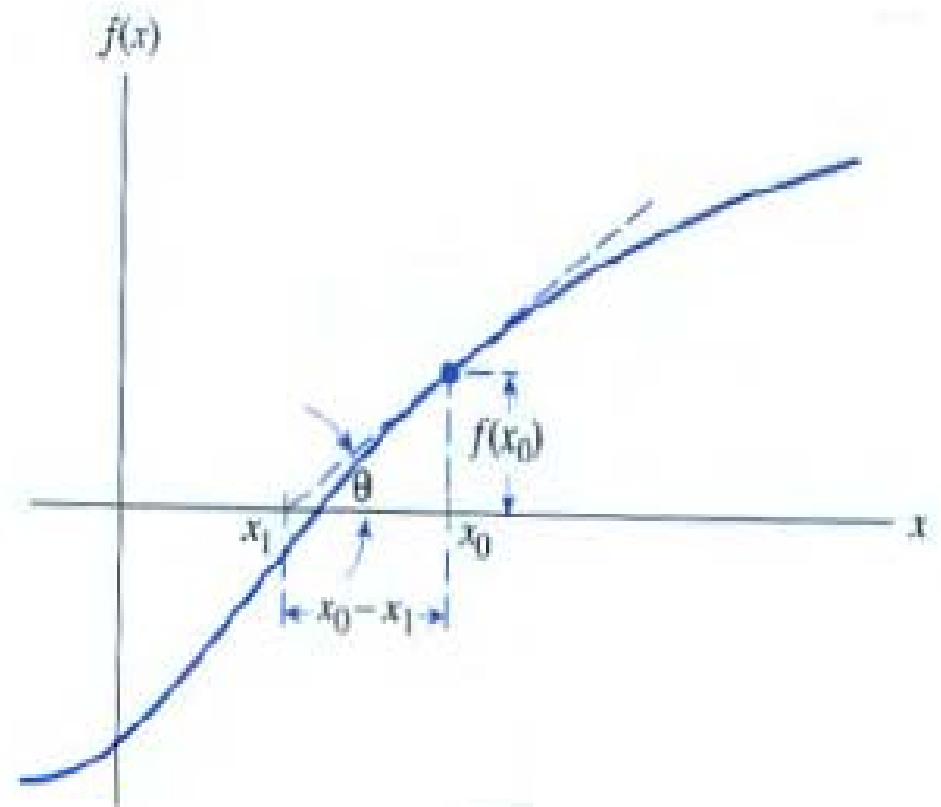


- Newton's method relies on calculus and uses linear approximation to the function by finding the tangent to the curve.

# Newton's Method



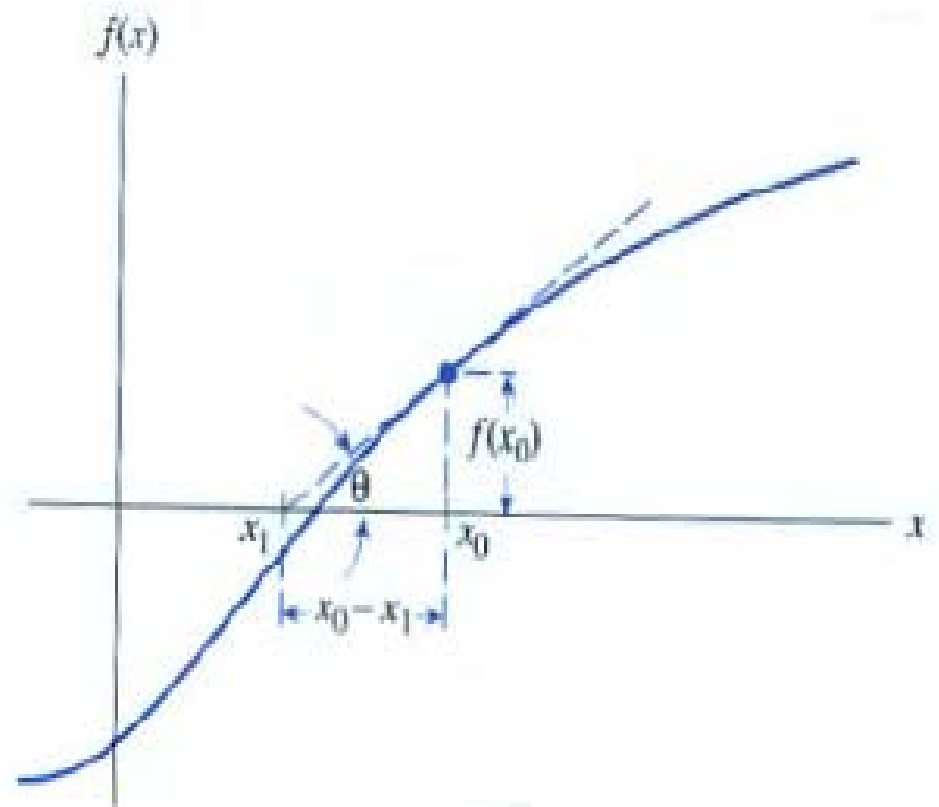
- Algorithm requires an initial guess,  $x_0$ , which is close to the root.
- The point where the tangent line to the function ( $f(x)$ ) meets the x-axis is the next approximation,  $x_1$ .
- This procedure is repeated until the value of 'x' is sufficiently close to zero.



# Newton's Method



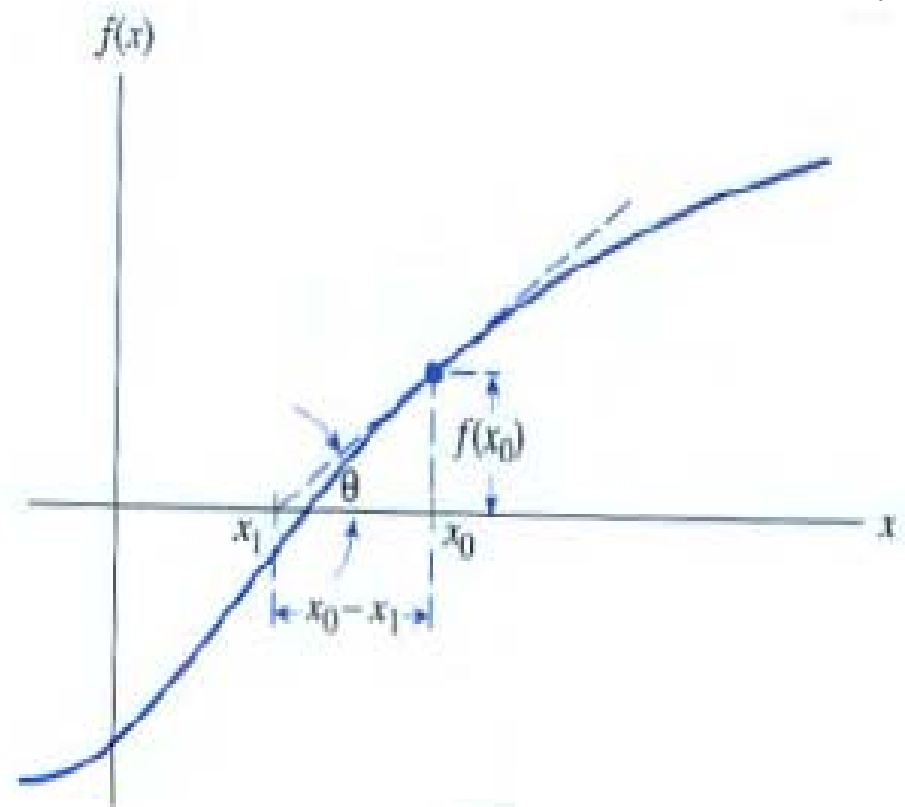
- The equation for Newton's Method can be determined graphically!



# Newton's Method



- The equation for Newton's Method can be determined graphically!
- From the diagram  $\tan \theta = f'(x_0) = f(x_0)/(x_0 - x_1)$

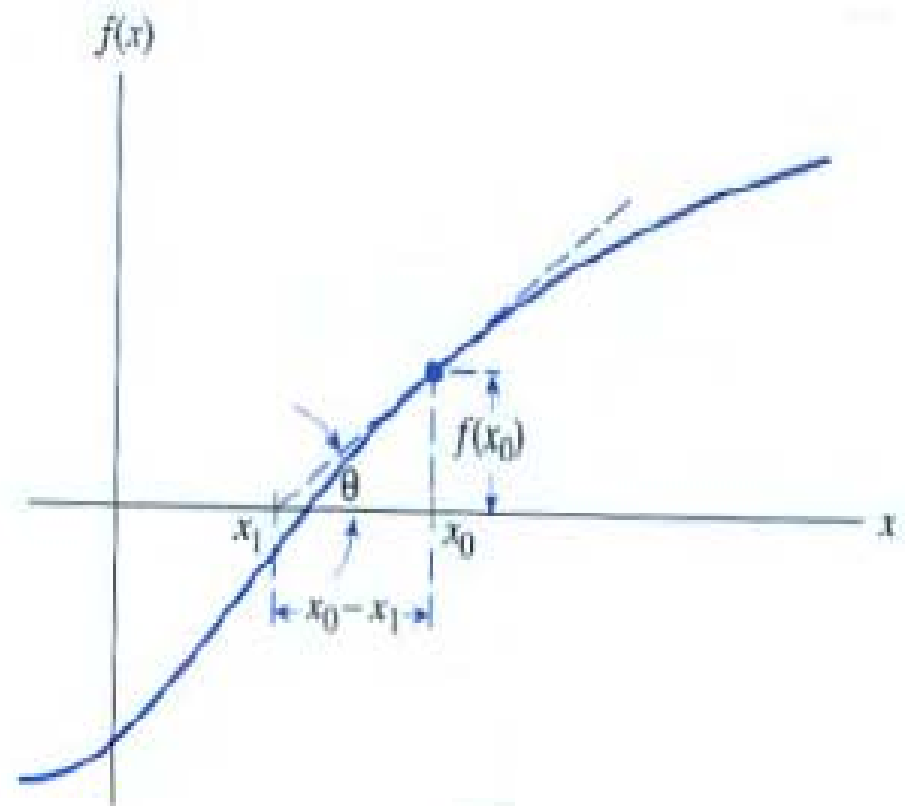




# Newton's Method



- The equation for Newton's Method can be determined graphically!
- From the diagram  $\tan \theta = f'(x_0) = f(x_0)/(x_0 - x_1)$
- Thus,  $x_1 = x_0 - f(x_0)/f'(x_0)$ .

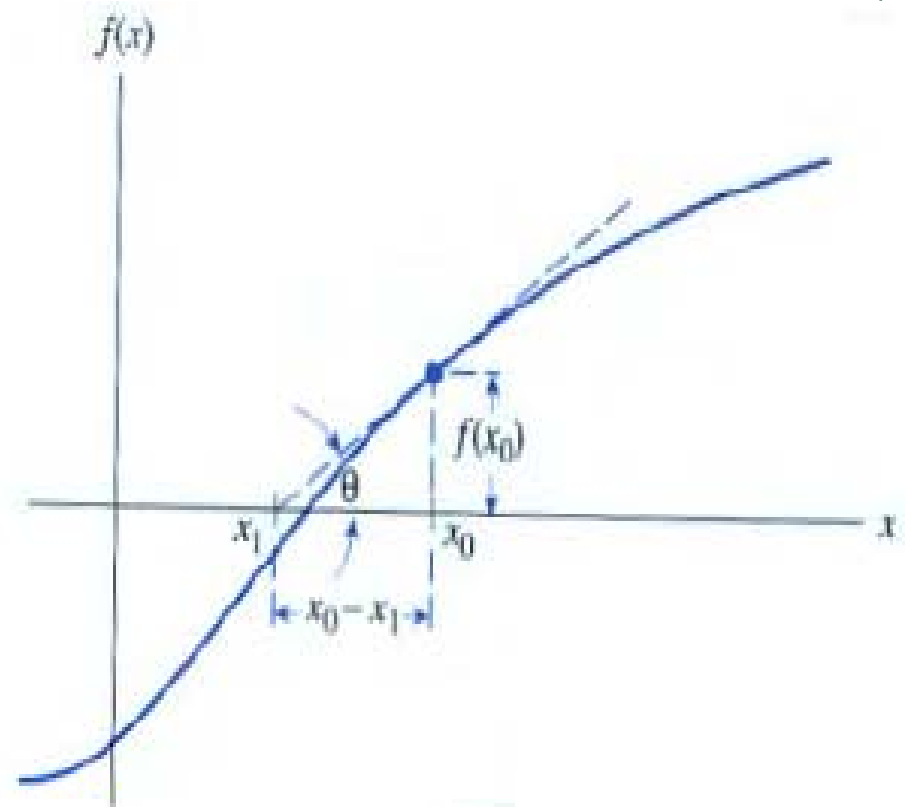


# Newton's Method



- The general form of Newton's Method is:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$



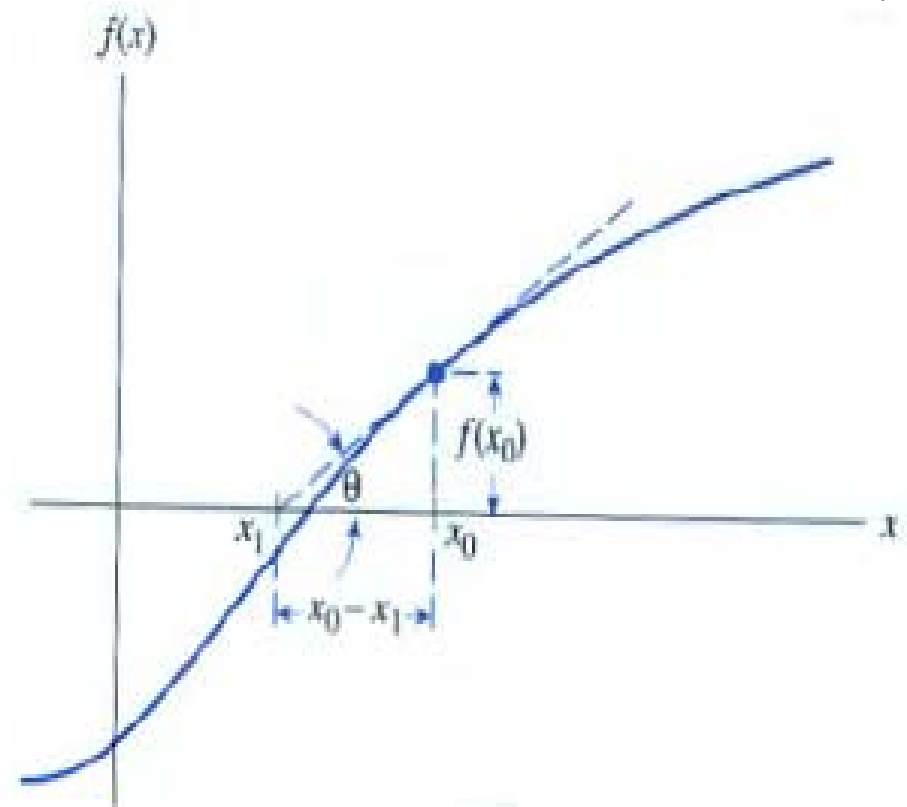
# Newton's Method



- The general form of Newton's Method is:  
$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

- **Algorithm**

- Pick a starting value for  $x$
- Repeat
  - $x := x - f(x)/f'(x)$
- Return  $x$



# Numerical Differentiation and Integration

- Standing in the heart of calculus are the mathematical concepts of *differentiation* and *integration*:

$$\frac{\Delta y}{\Delta x} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$
$$\frac{dy}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$
$$I = \int_a^b f(x) dx$$

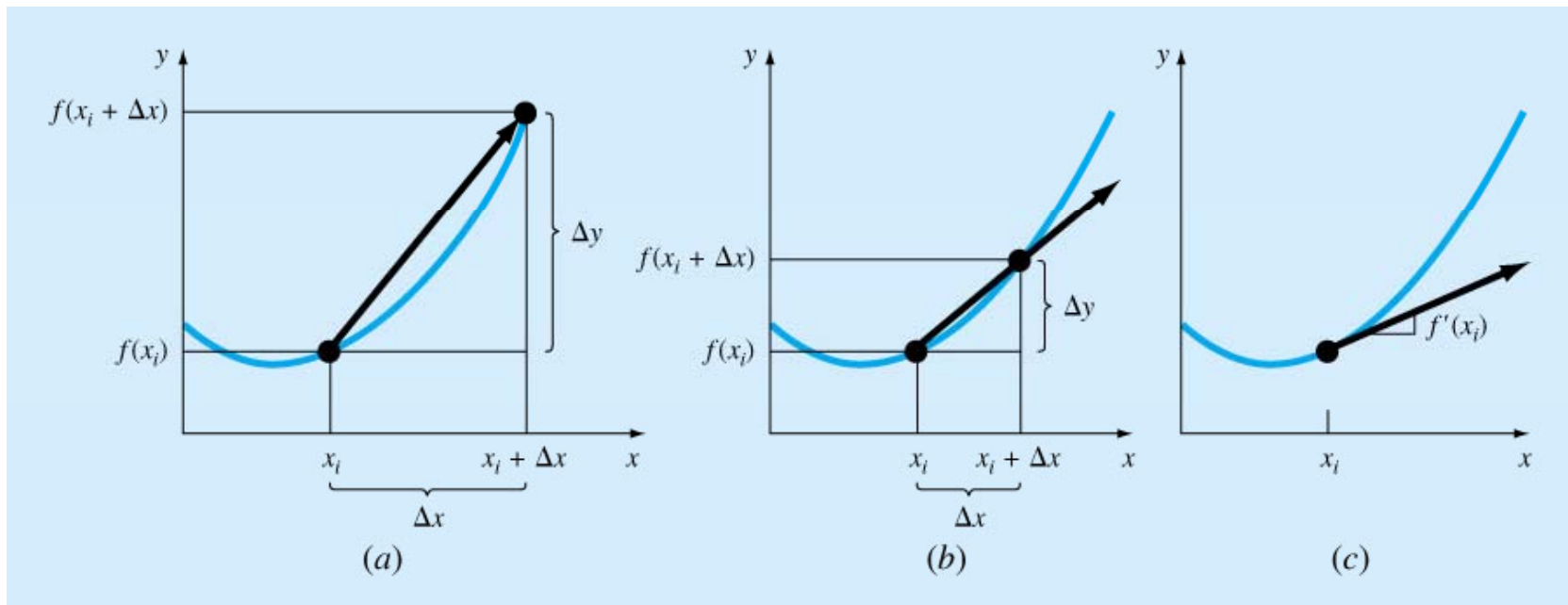
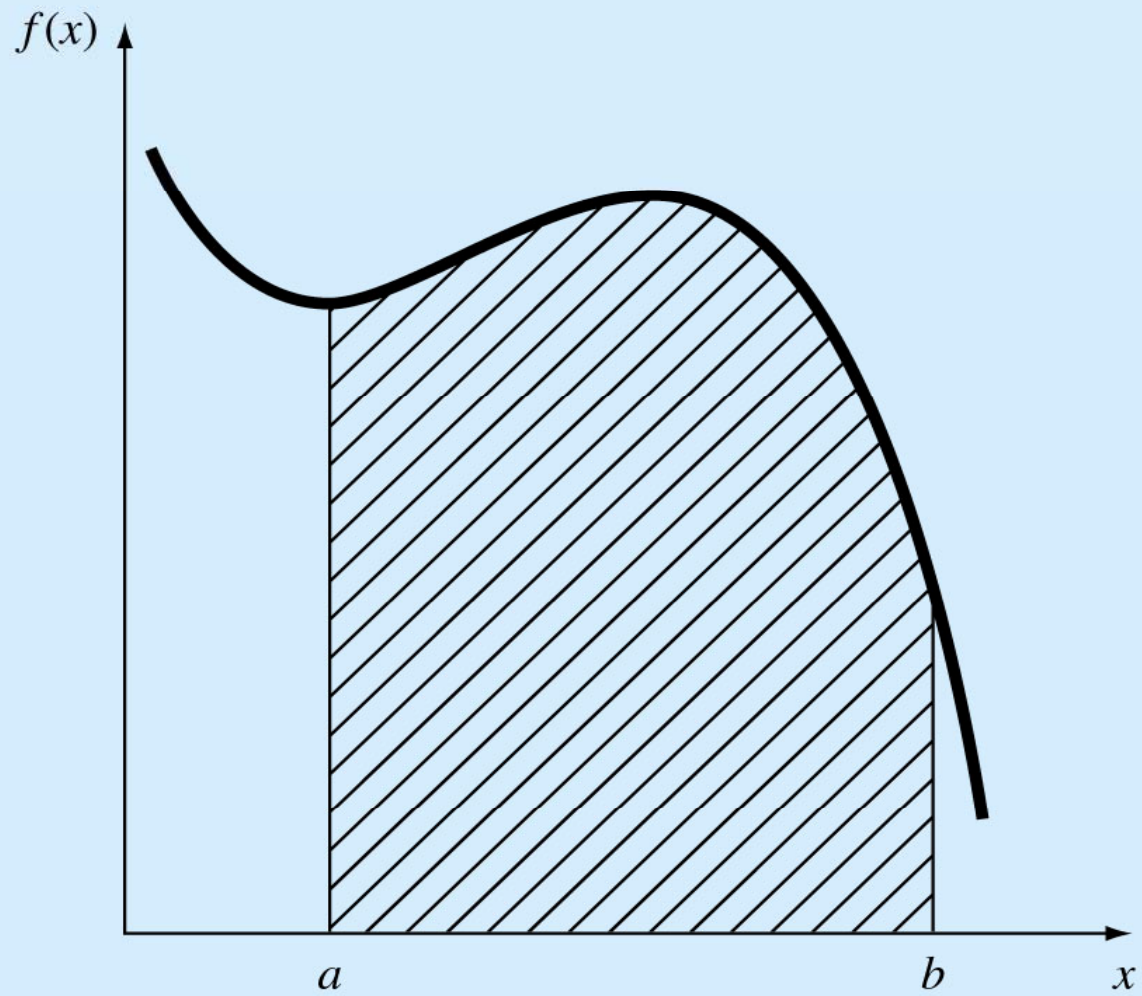


Figure 4.1

Figure 4.2



# Noncomputer Methods for Differentiation and Integration



- The function to be differentiated or integrated will typically be in one of the following three forms:
  - A simple continuous function such as polynomial, an exponential, or a trigonometric function.
  - A complicated continuous function that is difficult or impossible to differentiate or integrate directly.
  - A tabulated function where values of  $x$  and  $f(x)$  are given at a number of discrete points, as is often the case with experimental or field data.

# Figure 4.3

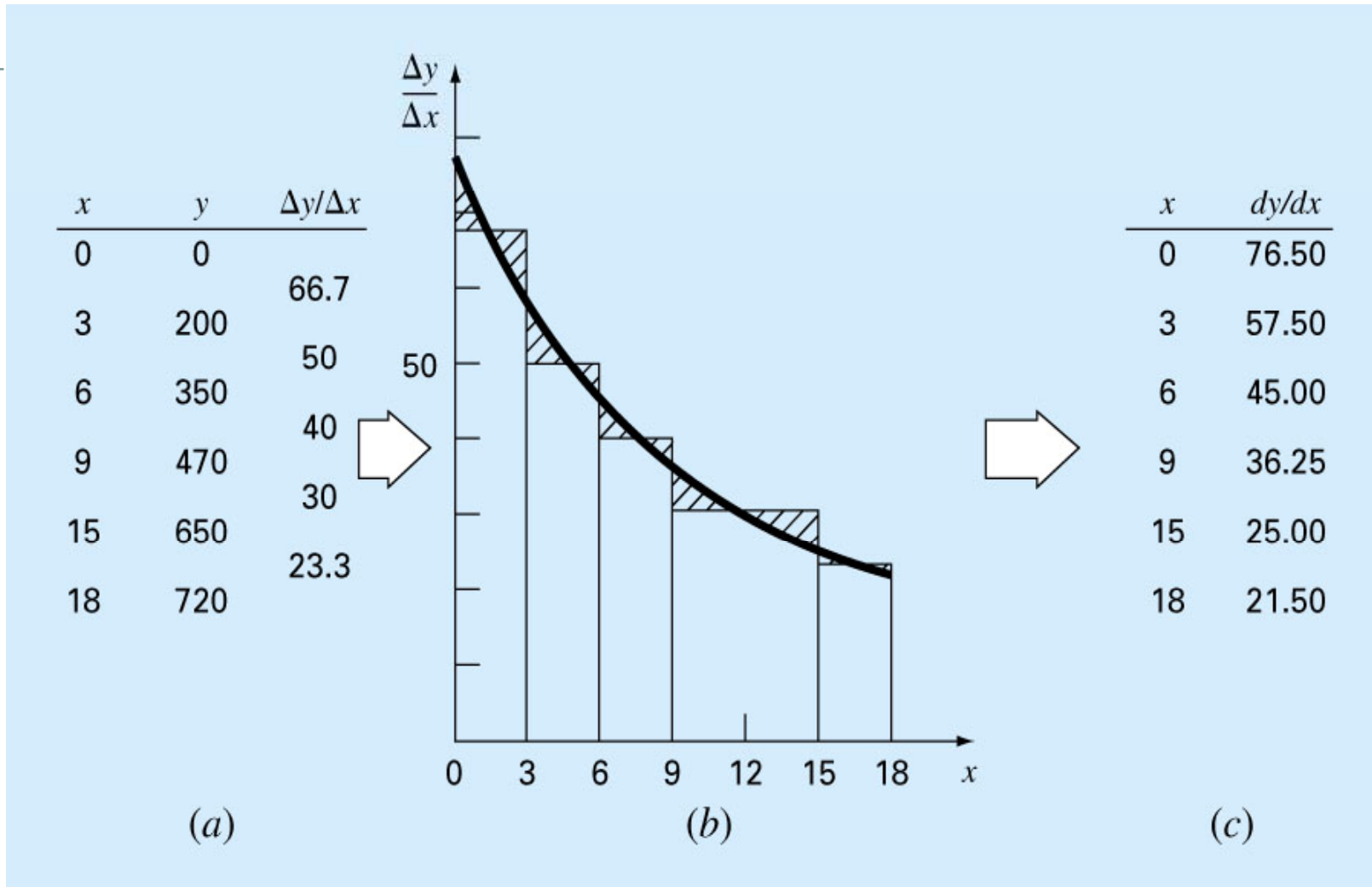




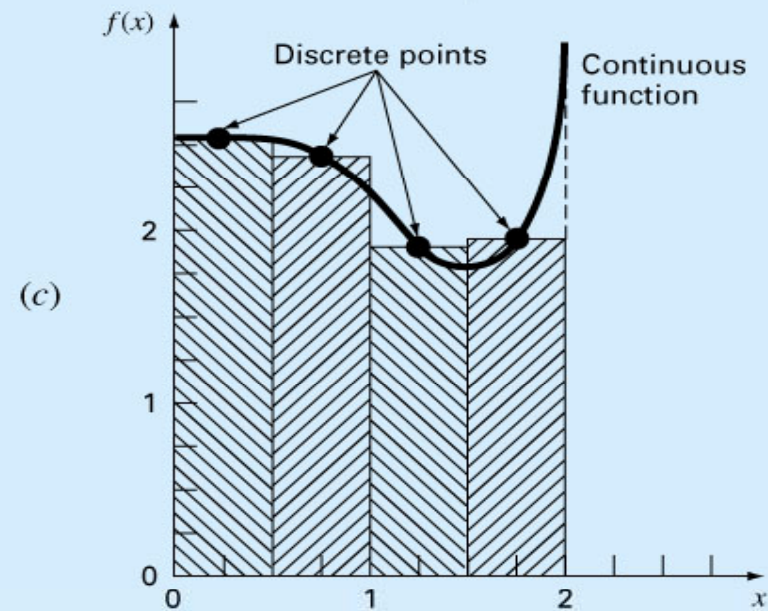
Figure 4.4

(a) 
$$\int_0^2 \frac{2 + \cos(1 + x^{3/2})}{\sqrt{1 + 0.5 \sin x}} e^{0.5x} dx$$



(b)

| $x$  | $f(x)$ |
|------|--------|
| 0.25 | 2.599  |
| 0.75 | 2.414  |
| 1.25 | 1.945  |
| 1.75 | 1.993  |



# Newton-Cotes Integration Formulas

- The *Newton-Cotes formulas* are the most common numerical integration schemes.
- They are based on the strategy of replacing a complicated function or tabulated data with an approximating function that is easy to integrate:

$$I = \int_a^b f(x)dx \cong \int_a^b f_n(x)dx$$

$$f_n(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + a_nx^n$$

Figure 4.5

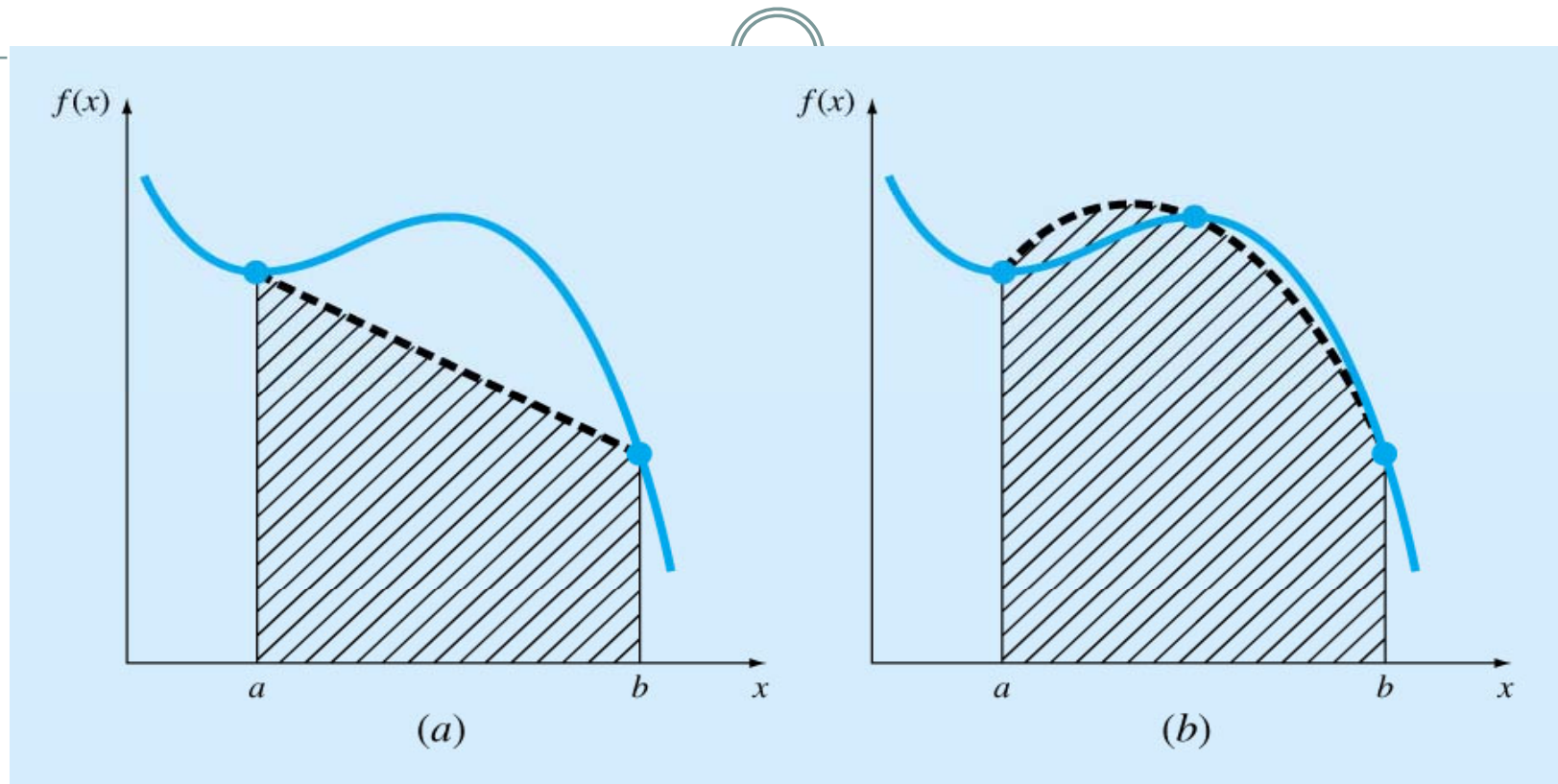
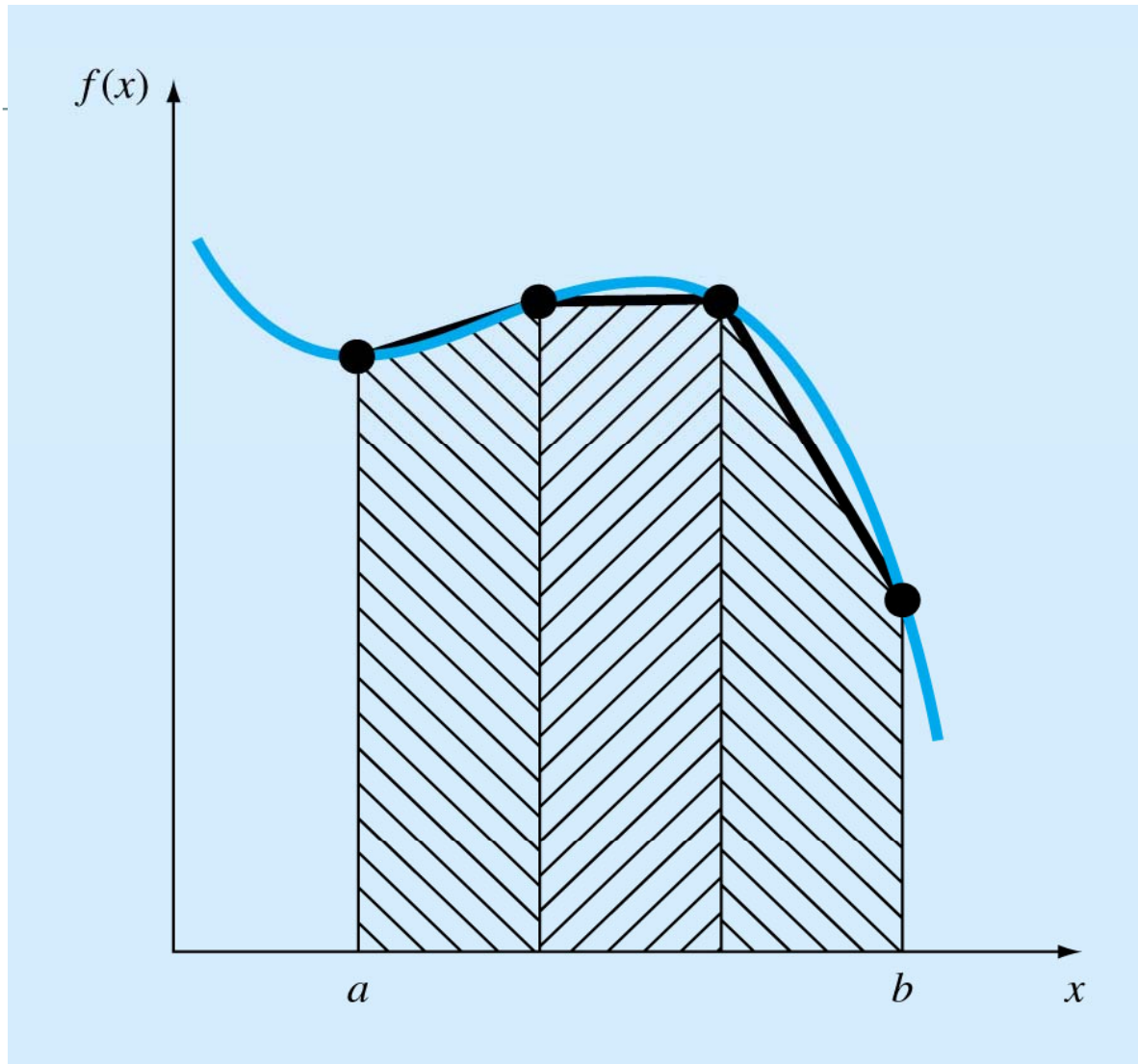


Figure 4.6



# The Trapezoidal Rule



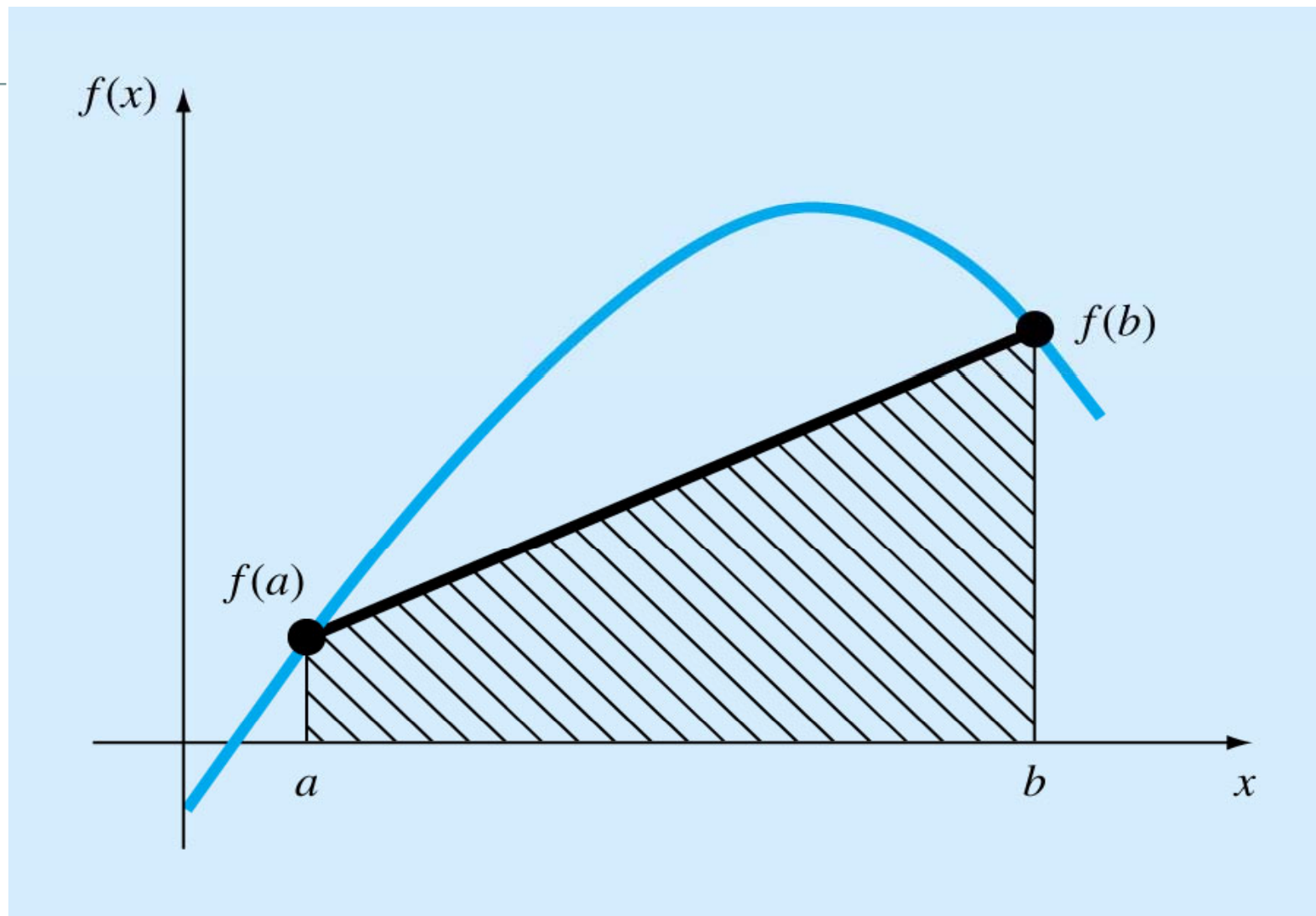
- The *Trapezoidal rule* is the first of the Newton-Cotes closed integration formulas, corresponding to the case where the polynomial is first order:

$$I = \int_a^b f(x)dx \cong \int_a^b f_1(x)dx$$

- The area under this first order polynomial is an estimate of the integral of  $f(x)$  between the limits of  $a$  and  $b$ :

$$I = (b - a) \frac{f(a) + f(b)}{2} \quad \left. \vphantom{I = (b - a) \frac{f(a) + f(b)}{2}} \right\} \textit{Trapezoidal rule}$$

Figure 4.7



## Error of the Trapezoidal Rule/

- When we employ the integral under a straight line segment to approximate the integral under a curve, error may be substantial:

$$E_t = -\frac{1}{12} f''(\xi)(b-a)^3$$

where  $\xi$  lies somewhere in the interval from  $a$  to  $b$ .

## The Multiple Application Trapezoidal Rule



- One way to improve the accuracy of the trapezoidal rule is to divide the integration interval from  $a$  to  $b$  into a number of segments and apply the method to each segment.
- The areas of individual segments can then be added to yield the integral for the entire interval.

$$h = \frac{b-a}{n} \quad a = x_0 \quad b = x_n$$

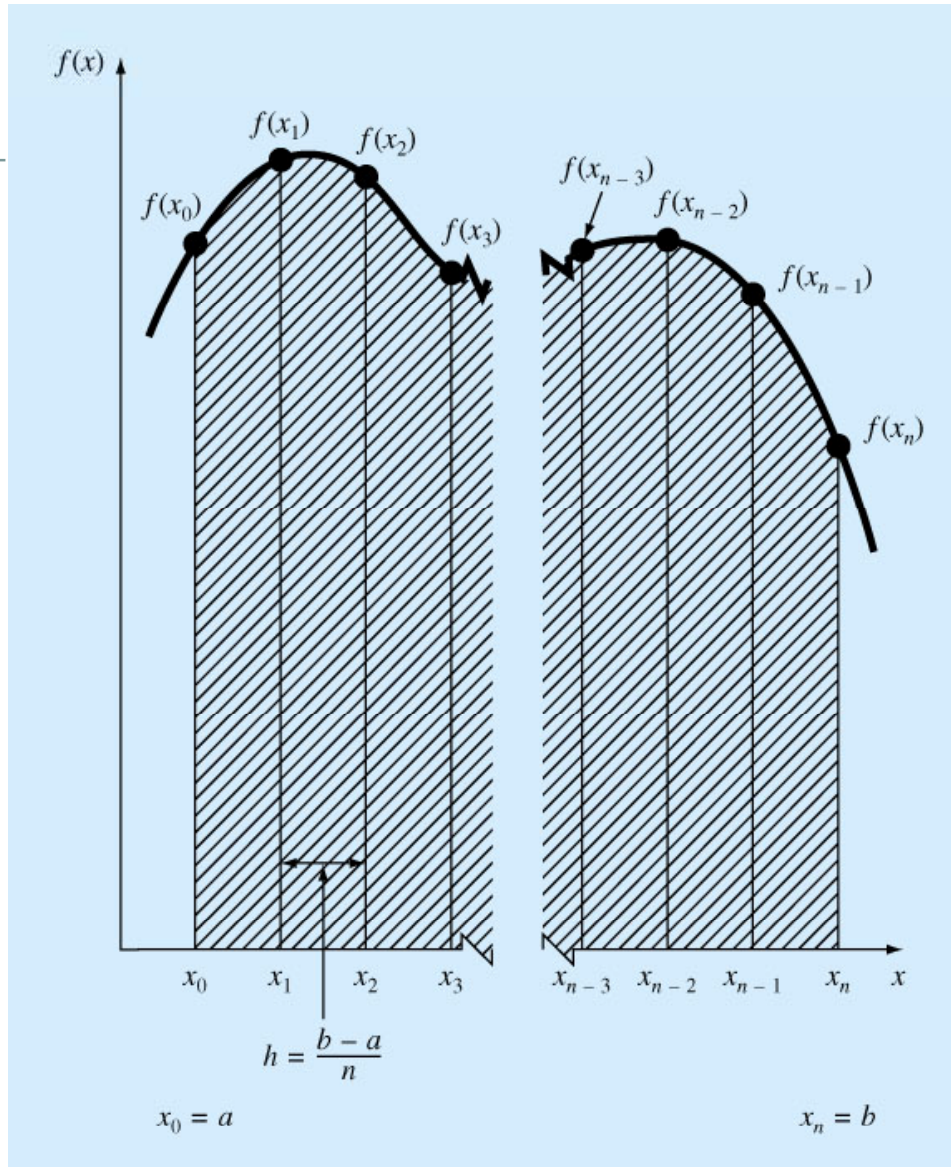
$$I = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$$

Substituting the trapezoidal rule for each integral yields:

$$I = h \frac{f(x_0) + f(x_1)}{2} + h \frac{f(x_1) + f(x_2)}{2} + \cdots + h \frac{f(x_{n-1}) + f(x_n)}{2}$$



Figure 4.8



# Simpson's Rules

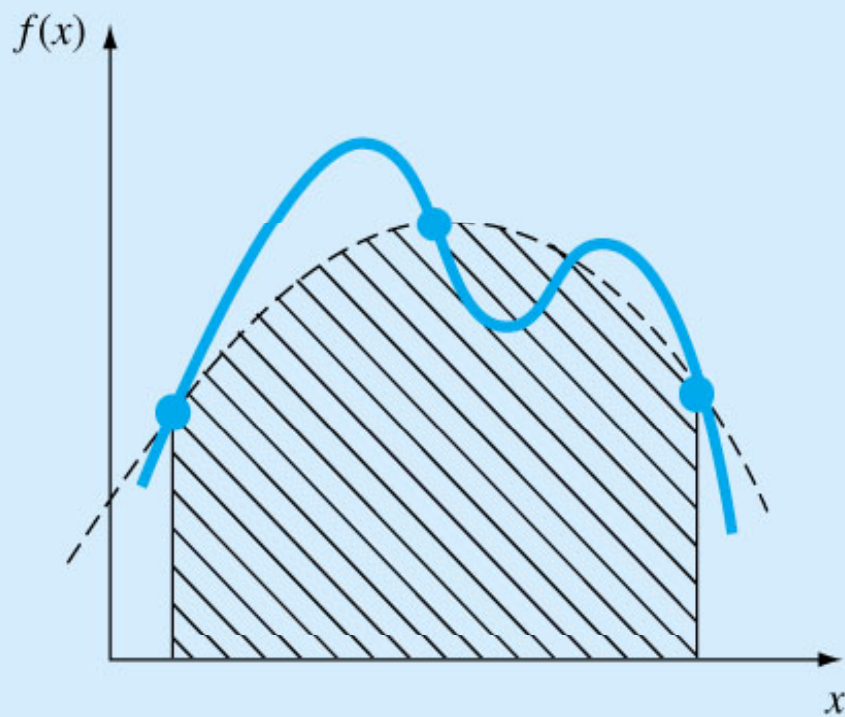


- More accurate estimate of an integral is obtained if a high-order polynomial is used to connect the points. The formulas that result from taking the integrals under such polynomials are called *Simpson's rules*.

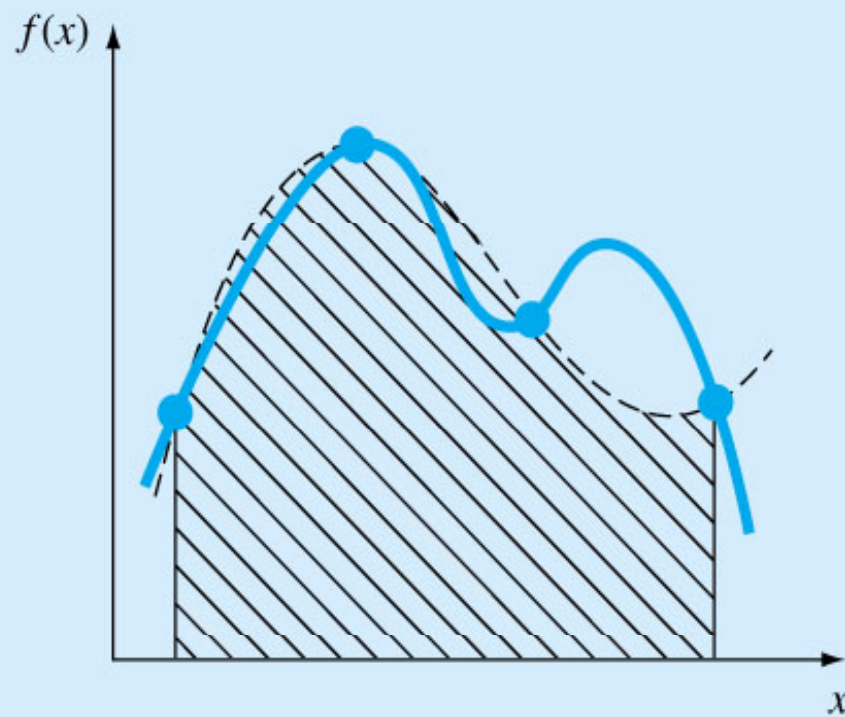
## **Simpson's 1/3 Rule/**

- Results when a second-order interpolating polynomial is used.

Figure 4.9



(a)



(b)

$$I = \int_a^b f(x)dx \cong \int_a^b f_2(x)dx$$

$$a = x_0 \quad b = x_2$$

$$I = \int_{x_0}^{x_2} \left[ \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \right] dx$$

$$I \cong \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] \quad h = \frac{b-a}{2}$$

Simpson's 1/3 Rule



The diagram shows a rectangular cross-section. The top portion is white, and the bottom portion is a light blue-grey gradient. A horizontal dashed line is positioned near the top, with a double-lined circle centered on it. A vertical dashed line extends downwards from the circle, passing through the center of the text 'Section C'. At the very bottom, there is a thin, solid dark teal horizontal bar.

Section C

# Solution of linear system of equations

126

- Circuit analysis (Mesh and node equations)
- Numerical solution of differential equations (Finite Difference Method)
- Numerical solution of integral equations (Finite Element Method, Method of Moments)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

# Consistency (Solvability)

127

- The linear system of equations  $Ax=b$  has a solution, or said to be **consistent** IFF

$$\text{Rank}\{A\} = \text{Rank}\{A|b\}$$

- A system is **inconsistent** when

$$\text{Rank}\{A\} < \text{Rank}\{A|b\}$$

$\text{Rank}\{A\}$  is the maximum number of linearly independent columns or rows of  $A$ . Rank can be found by using ERO (Elementary Row Operations) or ECO (Elementary column operations).

ERO  $\Rightarrow$  # of rows with at least one nonzero entry

ECO  $\Rightarrow$  # of columns with at least one nonzero entry

# Solution Techniques

128

- **Direct solution methods**
  - Finds a solution in a finite number of operations by transforming the system into an equivalent system that is 'easier' to solve.
  - Diagonal, upper or lower triangular systems are easier to solve
  - Number of operations is a function of system size  $n$ .
- **Iterative solution methods**
  - Computes successive approximations of the solution vector for a given  $A$  and  $b$ , starting from an initial point  $x_0$ .
  - Total number of operations is uncertain, may not converge.



# Direct solution Methods

129

- **Gaussian Elimination**

- By using ERO, matrix A is transformed into an upper triangular matrix (all elements below diagonal 0)
- Back substitution is used to solve the upper-triangular system

$$\begin{bmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{ni} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix} \xRightarrow{\text{ERO}} \begin{bmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \tilde{a}_{ii} & \cdots & \tilde{a}_{in} \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \tilde{b}_i \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$



Back substitution

# First step of elimination

130

**Pivotal element**

$$\begin{bmatrix} a_{11}^{(1)} \\ a_{21}^{(1)} \\ a_{31}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{bmatrix}
 \begin{bmatrix} a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}
 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
 =
 \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

$$\begin{matrix} m_{2,1} = a_{21}^{(1)} / a_{11}^{(1)} \\ m_{3,1} = a_{31}^{(1)} / a_{11}^{(1)} \\ \vdots \\ m_{n,1} = a_{n1}^{(1)} / a_{11}^{(1)} \end{matrix}
 \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}
 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
 =
 \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

# Second step of elimination

**Pivotal element**

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

$$\begin{matrix} m_{3,2} = a_{32}^{(2)} / a_{22}^{(2)} \\ \vdots \\ m_{n,2} = a_{n2}^{(2)} / a_{22}^{(2)} \end{matrix} \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \vdots \\ b_n^{(3)} \end{bmatrix}$$

# Gaussian elimination algorithm

132

$$m_{r,p} = a_{rp}^{(p)} / a_{pp}^{(p)}$$

$$a_{rp}^{(p)} = 0$$

$$b_r^{(p+1)} = b_r^{(p)} - m_{r,p} \times b_p^{(p)}$$

For c=p+1 to n

$$a_{rc}^{(p+1)} = a_{rc}^{(p)} - m_{r,p} \times a_{pc}^{(p)}$$

# Back substitution algorithm

133

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_{n-1,n-1}^{(n)} & a_{n-1,n}^{(n)} \\ 0 & 0 & 0 & 0 & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \vdots \\ b_{n-1}^{(n-1)} \\ b_n^{(n)} \end{bmatrix}$$

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}} \quad x_{n-1} = \frac{1}{a_{n-1,n-1}^{(n-1)}} \left[ b_{n-1}^{(n-1)} - a_{n-1,n}^{n-1} x_n \right]$$

$$x_i = \frac{1}{a_{ii}^{(i)}} \left[ b_i^{(i)} - \sum_{k=i+1}^n a_{ik}^{(i)} x_k \right] \quad i = n-1, n-2, \dots, 1$$

# Operation count

134

- Number of arithmetic operations required by the algorithm to complete its task.
- Generally only multiplications and divisions are counted
- Elimination process
- Back substitution
- Total

$$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$$
$$\frac{n^2 + n}{2}$$

Dominates  
Not efficient for  
different RHS vectors

$$\frac{n^3}{3} + n^2 - \frac{n}{3}$$

# LU Decomposition

135

$$A=LU$$

$$Ax=b \Rightarrow LUx=b$$

Define  $Ux=y$

$Ly=b$                   Solve  $y$  by forward substitution

ERO's must be performed on  $b$  as well as  $A$

The information about the ERO's are stored in  $L$

Indeed  $y$  is obtained by applying ERO's to  $b$  vector

$Ux=y$                   Solve  $x$  by backward substitution

# LU Decomposition by Gaussian elimination

136

There are infinitely many different ways to decompose A.  
 Most popular one: U=Gaussian eliminated matrix

L=Multipliers used for elimination

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ m_{2,1} & 1 & 0 & \cdots & 0 & 0 \\ m_{3,1} & m_{3,2} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & 0 \\ m_{n-1,1} & m_{n-1,2} & m_{n-1,3} & \cdots & 1 & \vdots \\ m_{n,1} & m_{n,2} & m_{n,3} & m_{n,4} & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_{n-1n-1}^{(n)} & a_{n-1n}^{(n)} \\ 0 & 0 & 0 & 0 & a_{nn}^{(n)} \end{bmatrix}$$

**Compact storage:** The diagonal entries of L matrix are all 1's, they don't need to be stored. LU is stored in a single matrix.



# Operation count

137

- A=LU Decomposition

$$\frac{n^3}{3} - \frac{n}{3}$$

Done only once

- Ly=b forward substitution

$$\frac{n^2 - n}{2}$$

- Ux=y backward substitution

$$\frac{n^2 + n}{2}$$

- Total  $\frac{n^3}{3} + n^2 - \frac{n}{3}$

- For different RHS vectors, the system can be efficiently solved.

# Pivoting

138

- Computer uses *finite-precision* arithmetic
- A small error is introduced in each arithmetic operation, *error propagates*
- When the pivotal element is very small, the multipliers will be large.
- Adding numbers of widely differening magnitude can lead to *loss of significance*.
- To reduce error, row interchanges are made to *maximise* the magnitude of *the pivotal element*

# Example: Without Pivoting

139

4-digit arithmetic

$$\begin{bmatrix} 1.133 & 5.281 \\ 24.14 & -1.210 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.414 \\ 22.93 \end{bmatrix}$$

$$m_{21} = \frac{24.14}{1.133} = 21.31 \quad \begin{bmatrix} 1.133 & 5.281 \\ 0.000 & -113.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.414 \\ -113.8 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.9956 \\ 1.001 \end{bmatrix}$$

Loss of significance

## Example: With Pivoting

140

$$\begin{bmatrix} 24.14 & -1.210 \\ 1.133 & 5.281 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 22.93 \\ 6.414 \end{bmatrix}$$

$$m_{21} = \frac{1.133}{24.14} = 0.04693 \quad \begin{bmatrix} 24.14 & -1.210 \\ 0.000 & 5.338 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 22.93 \\ 5.338 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.000 \\ 1.000 \end{bmatrix}$$

# Pivoting procedures

Eliminated part

$$\begin{bmatrix}
 a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1i}^{(1)} & \dots & a_{1j}^{(1)} & \dots & a_{1n}^{(1)} \\
 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2i}^{(2)} & \dots & a_{2j}^{(2)} & \dots & a_{2n}^{(2)} \\
 0 & 0 & a_{33}^{(3)} & \dots & a_{3i}^{(3)} & \dots & a_{3j}^{(3)} & \dots & a_{3n}^{(3)} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 0 & 0 & 0 & \dots & a_{ii}^{(i)} & \dots & a_{ij}^{(i)} & \dots & a_{in}^{(i)} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 0 & 0 & 0 & \dots & a_{ji}^{(i)} & \dots & a_{jj}^{(i)} & \dots & a_{jn}^{(i)} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 0 & 0 & 0 & \dots & a_{ni}^{(i)} & \dots & a_{nj}^{(i)} & \dots & a_{nn}^{(i)}
 \end{bmatrix}$$

Pivotal row

Pivotal column

# Row pivoting

142

- Most commonly used *partial pivoting* procedure
- Search the pivotal column
- Find the largest element in magnitude
- Then switch this row with the pivotal row

# Row pivoting

143

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1i}^{(1)} & \cdots & a_{1j}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2i}^{(2)} & \cdots & a_{2j}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3i}^{(3)} & \cdots & a_{3j}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ii}^{(i)} & \cdots & a_{ij}^{(i)} & \cdots & a_{in}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ji}^{(i)} & \cdots & a_{jj}^{(i)} & \cdots & a_{jn}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ni}^{(i)} & \cdots & a_{nj}^{(i)} & \cdots & a_{nn}^{(i)} \end{bmatrix}$$

Interchange  
these rows



Largest in magnitude

# Column pivoting

144

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1i}^{(1)} & \cdots & a_{1j}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2i}^{(2)} & \cdots & a_{2j}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3i}^{(3)} & \cdots & a_{3j}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ii}^{(i)} & \cdots & a_{ij}^{(i)} & \cdots & a_{in}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ji}^{(i)} & \cdots & a_{jj}^{(i)} & \cdots & a_{jn}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ni}^{(i)} & \cdots & a_{nj}^{(i)} & \cdots & a_{nn}^{(i)} \end{bmatrix}$$

Largest in magnitude

Interchange these columns

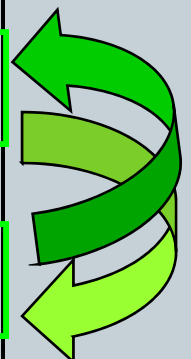




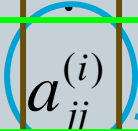
# Complete pivoting

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1i}^{(1)} & \cdots & a_{1j}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2i}^{(2)} & \cdots & a_{2j}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3i}^{(3)} & \cdots & a_{3j}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ii}^{(i)} & \cdots & a_{ij}^{(i)} & \cdots & a_{in}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ji}^{(i)} & \cdots & a_{jj}^{(i)} & \cdots & a_{jn}^{(i)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{ni}^{(i)} & \cdots & a_{nj}^{(i)} & \cdots & a_{nn}^{(i)} \end{bmatrix}$$

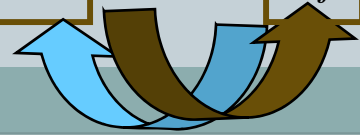
Interchange these rows



Largest in magnitude



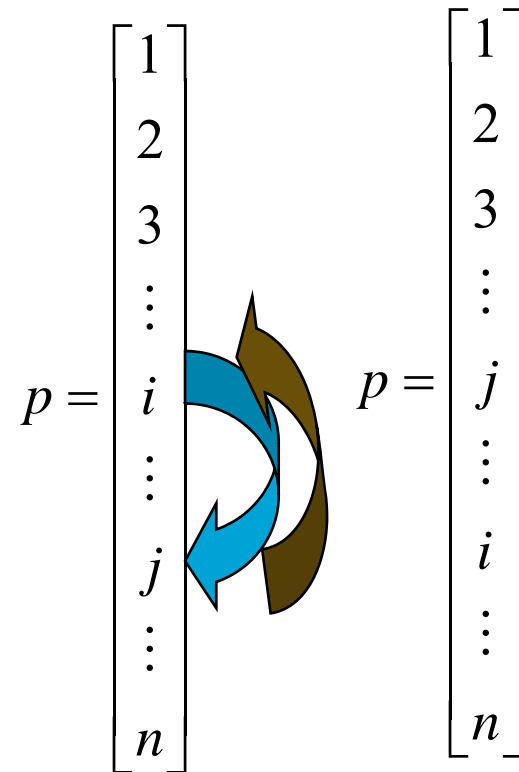
Interchange these columns



# Row Pivoting in LU Decomposition

146

- When two rows of  $A$  are interchanged, those rows of  $b$  should also be interchanged.
- Use a pivot vector. Initial pivot vector is integers from 1 to  $n$ .
- When two rows ( $i$  and  $j$ ) of  $A$  are interchanged, apply that to pivot vector.



# Example

147

$$A = \begin{bmatrix} 0 & 3 & 2 \\ -4 & -2 & 1 \\ 1 & 4 & -2 \end{bmatrix} \quad b = \begin{bmatrix} 12 \\ -5 \\ 3 \end{bmatrix} \quad p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Column search: Maximum magnitude second row  
Interchange 1<sup>st</sup> and 2<sup>nd</sup> rows

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ 0 & 3 & 2 \\ 1 & 4 & -2 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

## Example continued...

148

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ 0 & 3 & 2 \\ 1 & 4 & -2 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

Eliminate  $a_{21}$  and  $a_{31}$  by using  $a_{11}$  as pivotal element  
 $A=LU$  in compact form (in a single matrix)

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ 0 & 3 & 2 \\ -0.25 & 3.5 & -1.75 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

Multipliers (L matrix)

## Example continued...

149

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ 0 & 3 & 2 \\ -0.25 & 3.5 & -1.75 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

Column search: Maximum magnitude at the third row  
Interchange 2<sup>nd</sup> and 3<sup>rd</sup> rows

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ -0.25 & 3.5 & -1.75 \\ 0 & 3 & 2 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

## Example continued...

150

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ -0.25 & 3.5 & -1.75 \\ 0 & 3 & 2 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Eliminate  $a_{32}$  by using  $a_{22}$  as pivotal element

$$A' = \begin{bmatrix} -4 & -2 & 1 \\ -0.25 & 3.5 & -1.75 \\ 0 & 3/3.5 & 3.5 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Multipliers (L matrix)

## Example continued...

151

$$A' = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0 & 3/3.5 & 1 \end{bmatrix} \begin{bmatrix} -4 & -2 & 1 \\ 0 & 3.5 & -1.75 \\ 0 & 0 & 3.5 \end{bmatrix} \quad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

$$p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \quad b = \begin{bmatrix} 12 \\ -5 \\ 3 \end{bmatrix} \Rightarrow b' = \begin{bmatrix} -5 \\ 3 \\ 12 \end{bmatrix}$$

$$A'x = b' \quad LUx = b'$$

$$Ux = y$$

$$Ly = b'$$

# Example continued...

152

$$Ly=b'$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0 & 3/3.5 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -5 \\ 3 \\ 12 \end{bmatrix}$$



Forward  
substitution

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -5 \\ 1.75 \\ 10.5 \end{bmatrix}$$

$$Ux=y$$

$$\begin{bmatrix} -4 & -2 & 1 \\ 0 & 3.5 & -1.75 \\ 0 & 0 & 3.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -5 \\ 1.75 \\ 10.5 \end{bmatrix}$$



Backward  
substitution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$



# Gauss-Jordan elimination

153

- The elements above the diagonal are made zero at the same time that zeros are created below the diagonal

$$\begin{array}{ccc}
 \left[ \begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array} \right] & \xrightarrow{\quad} & \left[ \begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array} \right] \\
 & & \searrow & & \\
 \left[ \begin{array}{cccc|c} a_{11}^{(1)} & 0 & \cdots & a_{nn}^{(2)} & b_1^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{nn}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{array} \right] & \xrightarrow{\quad} & \left[ \begin{array}{cccc|c} a_{11}^{(1)} & 0 & \cdots & 0 & b_1^{(n-1)} \\ 0 & a_{22}^{(2)} & \cdots & 0 & b_2^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{array} \right]
 \end{array}$$

# Gauss-Jordan Elimination

154

- Almost 50% more arithmetic operations than Gaussian elimination
- Gauss-Jordan (GJ) Elimination is preferred when the inverse of a matrix is required.

- Apply GJ elimination to convert  $A$  into an identity matrix.

$$\left[ A \mid I \right]$$
$$\left[ I \mid A^{-1} \right]$$



# Eigen values and Eigenvectors

Suppose we have some vector  $A$ , in the equation  $Ax=b$  and we want to find the which vectors  $x$  are pointing the same direction after the transformation. These vectors are called Eigenvectors.

The vector  $b$  must be a scalar multiple of  $x$ . The scalar that multiplies  $x$  is called the Eigen value

The main equation for this section is

$$Ax = \lambda x$$

Any vector  $x$  that satisfies this equation is an Eigenvector, the corresponding  $\lambda$  is the Eigen value

Note: for this section we are only considering square matrices.

## Example A



Let's examine some vectors that we are already familiar with and determine the Eigenvectors and Eigenvalues.

Consider a Projection matrix  $P$  in  $\mathbb{R}^3$ , that projects vectors on to a plane. What are the Eigenvectors and Eigenvalues?

## Answer to Example A



Some Eigenvectors are the vectors that are already in the plane that is being projected on. In that case the vector does not change so the Eigenvalue for these vectors is 1

Other Eigenvectors are those orthogonal to the plane that is being projected on. Those vectors become the zero vector (which is considered parallel to all vectors). The Eigen value for these vectors is zero.

Look at the case  $\lambda = 0$



If  $A$  is a singular matrix, then we can solve

$$Ax = \lambda x$$

What did we previously call these values?

# Answer



If  $\lambda = 0$  then we are solving  $Ax=0$  which is the null space (Kernel)



# The following statements are equivalent

## A is invertible



The linear system  $Ax=b$  has a unique solution  $x$  for all  $b$

$$\text{rref}(A) = I_n$$

$$\text{Rank}(A) = n$$

$$\text{Im}(A) = \mathbb{R}^n$$

$$\ker(A) = \{0\}$$

The column vectors of  $A$  form a basis of  $\mathbb{R}^n$

The column vectors of  $A$  span  $\mathbb{R}^n$

The column vectors of  $A$  are linearly independent

$$\det A \neq 0$$

0 fails to be an eigenvalue of  $A$

## How can I solve $Ax = \lambda x$

Bring everything on one side

$$Ax - \lambda x = 0$$

$$(A - \lambda I)x = 0$$

If this can be solved then the matrix

$(A - \lambda I)$  must be singular

Which means that  $\det(A - \lambda I) = 0$

This equation is called the characteristic equation.

There should be  $n$  values to this equation (although some could be repeated)

Once we find  $\lambda$  find the nullspace of  $(A - \lambda I)x = 0$   
to find the  $x$ 's (Eigenvectors)

# Find the Eigenvalues



$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

# Find the Eigenvalues

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Find  $\det (A - \lambda I) = 0$

$$(3 - \lambda)^2 - 1$$

$$\begin{bmatrix} 3 - \lambda & 1 \\ 1 & 3 - \lambda \end{bmatrix}$$

$$\lambda^2 - 6\lambda + 8 = 0$$

$$(\lambda - 4)(\lambda - 2) = 0$$

$$\lambda = 4 \quad \lambda = 2$$

Note: this equation is called the characteristic equation

Plug in

$\lambda = 2$  and find

a basis for

kernel

Plug in  $\lambda = 4$  to find the Eigenvectors

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

find a basis for the null space (kernel)

# Eigenvalues of triangular matrices



Find the Eigenvalues of

$$\begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$$

# Triangular matrices slide 1 of solutions

- Find the Eigenvalues

$$A = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$$

- $A - \lambda I = \begin{bmatrix} 3 - \lambda & 1 \\ 0 & 3 - \lambda \end{bmatrix}$

$$\text{Det}(A) = (3 - \lambda)^2 = 0 \quad \lambda = 3$$

This matrix has a repeated Eigenvalue.

Note: for triangular matrices, the values on the diagonal of the matrix are the Eigenvalues

# Triangular matrices

Find the Eigenvectors  $A - \lambda I =$

Replace  $\lambda$  by 3

Find the null space

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 3 - \lambda & 1 \\ 0 & 3 - \lambda \end{bmatrix}$$

This matrix has only 1 Eigenvector!

A repeated  $\lambda$  gives the possibility of a lack of Eigenvectors



# Section D



# Ordinary Differential Equations

- Equations which are composed of an unknown function and its derivatives are called *differential equations*.
- Differential equations play a fundamental role in engineering because many physical phenomena are best formulated mathematically in terms of their rate of change.

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

$v$ - dependent variable

$t$ - independent variable



- When a function involves one dependent variable, the equation is called an *ordinary differential equation (or ODE)*. A *partial differential equation (or PDE)* involves two or more independent variables.
- Differential equations are also classified as to their order.
  - *A first order equation* includes a first derivative as its highest derivative.
  - *A second order equation* includes a second derivative.
- Higher order equations can be reduced to a system of first order equations, by redefining a variable.

# ODEs and Engineering Practice

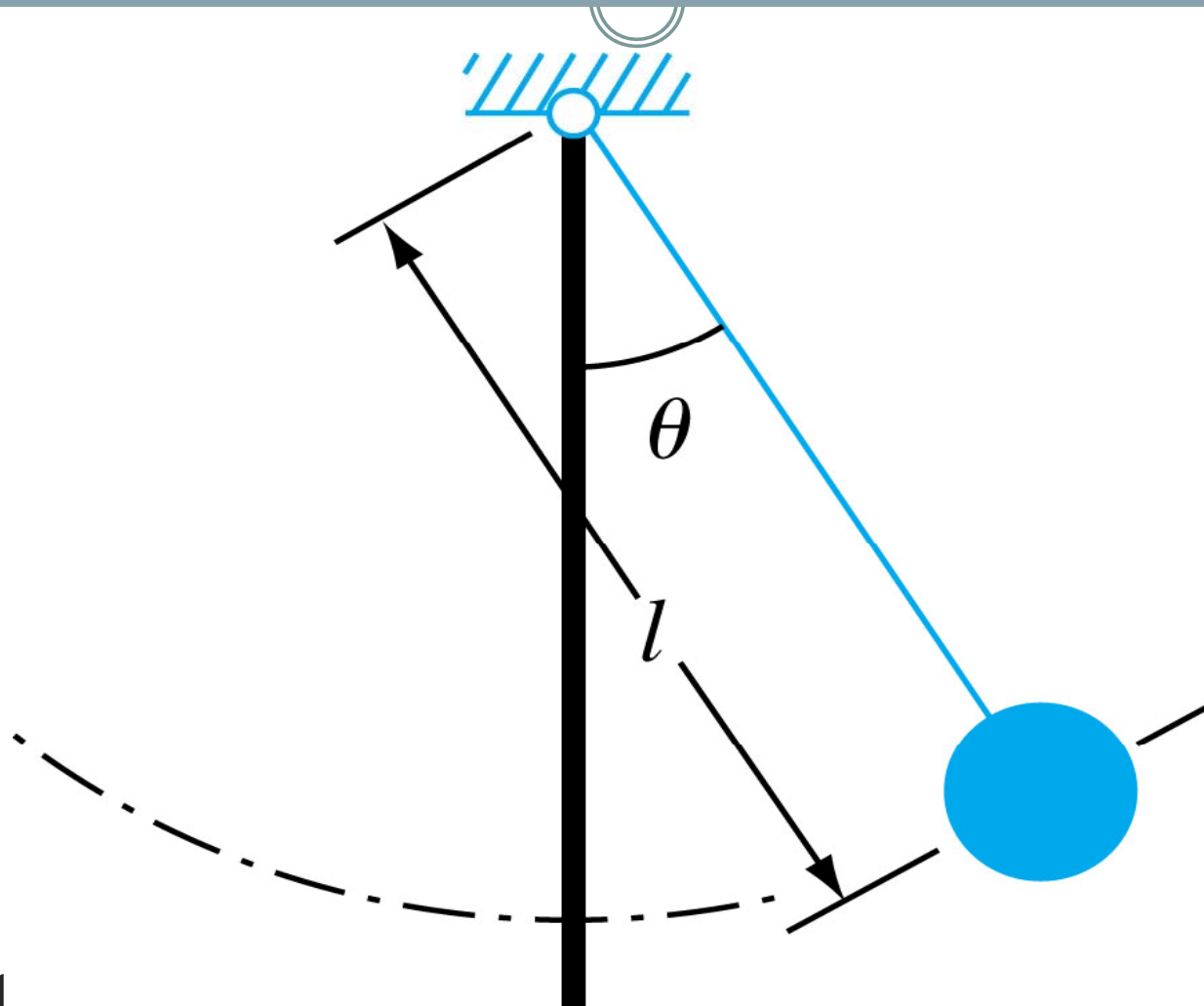


Figure 7.1

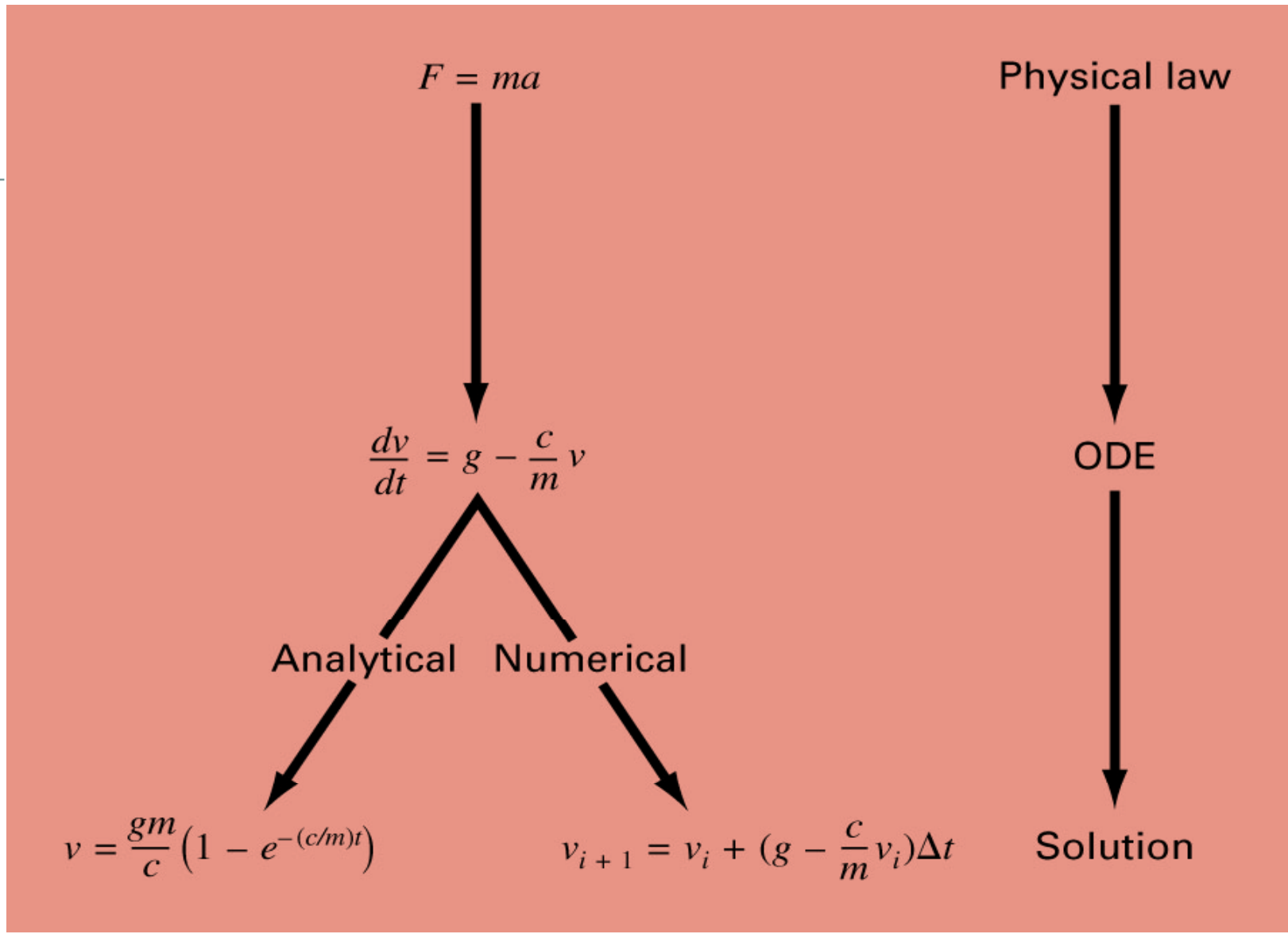


Figure 7.2

# Runga-Kutta Methods

- This chapter is devoted to solving ordinary differential equations of the form

$$\frac{dy}{dx} = f(x, y)$$

Euler's Method

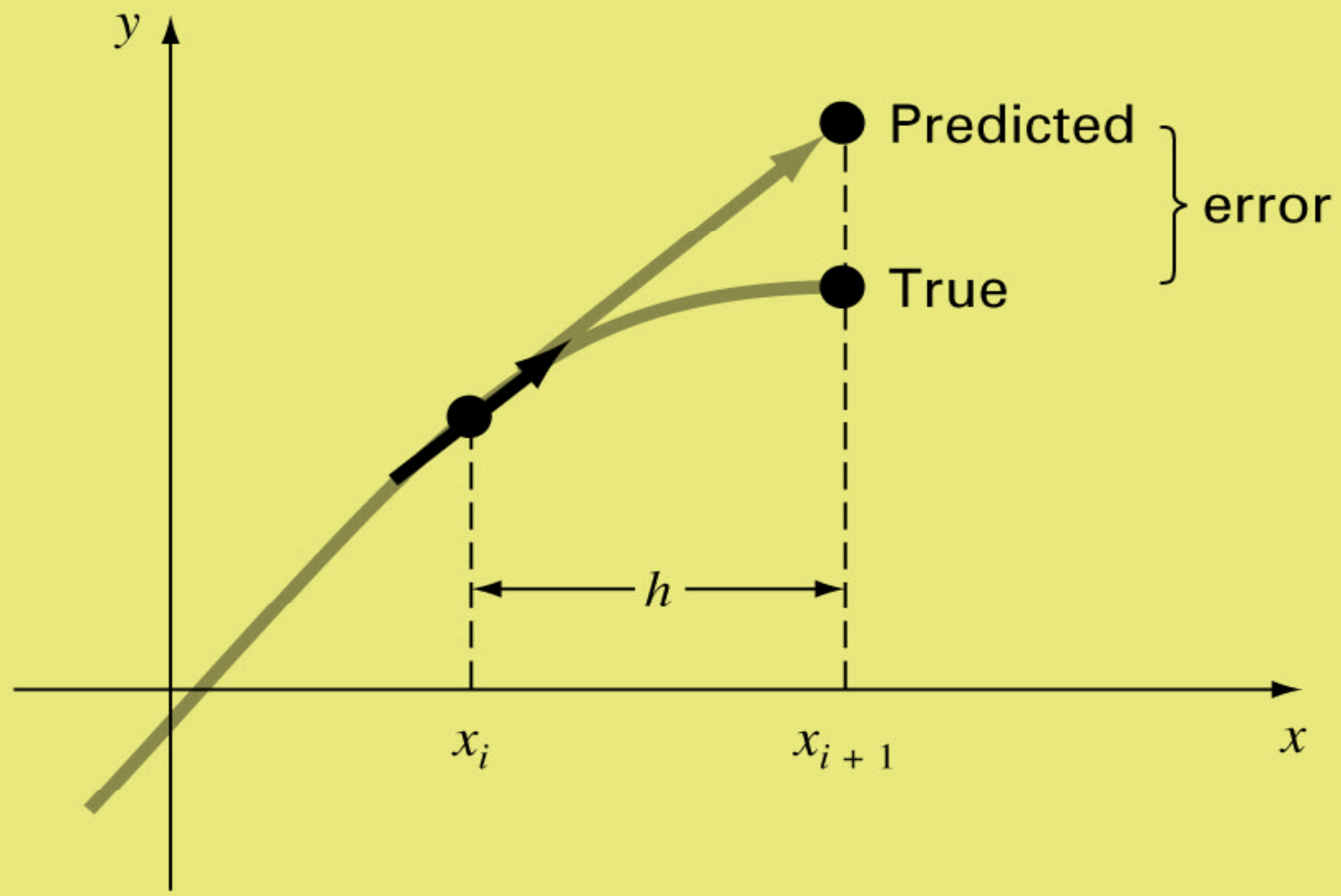


Figure 7.3

- The first derivative provides a direct estimate of the slope at  $x_i$

$$\phi = f(x_i, y_i)$$

where  $f(x_i, y_i)$  is the differential equation evaluated at  $x_i$  and  $y_i$ . This estimate can be substituted into the equation:

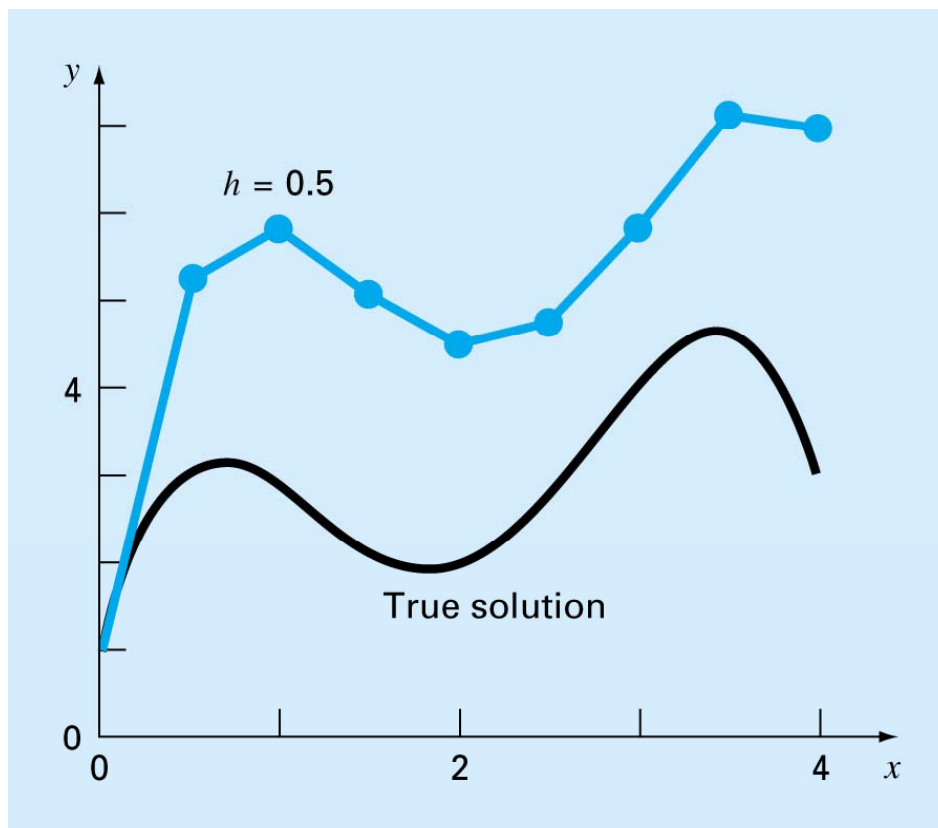
$$y_{i+1} = y_i + f(x_i, y_i)h$$

- A new value of  $y$  is predicted using the slope to extrapolate linearly over the step size  $h$ .

$$\frac{dy}{dx} = f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$$

Starting point  $x_0 = 0, y_0 = 1$

$$y_{i+1} = y_i + f(x_i, y_i)h = 1 + 8.5 * 0.5 = 5.25$$



Not good



## Error Analysis for Euler's Method/

- Numerical solutions of ODEs involves two types of error:

- *Truncation error*

- ✦ *Local truncation error*

$$E_a = \frac{f'(x_i, y_i)}{2!} h^2$$

$$E_a = O(h^2)$$

- ✦ *Propagated truncation error*

- The sum of the two is the *total or global truncation error*
- *Round-off errors*



- The Taylor series provides a means of quantifying the error in Euler's method. However;
  - The Taylor series provides only an estimate of the local truncation error-that is, the error created during a single step of the method.
  - In actual problems, the functions are more complicated than simple polynomials. Consequently, the derivatives needed to evaluate the Taylor series expansion would not always be easy to obtain.
- In conclusion,
  - the error can be reduced by reducing the step size
  - If the solution to the differential equation is linear, the method will provide error free predictions as for a straight line the 2<sup>nd</sup> derivative would be zero.

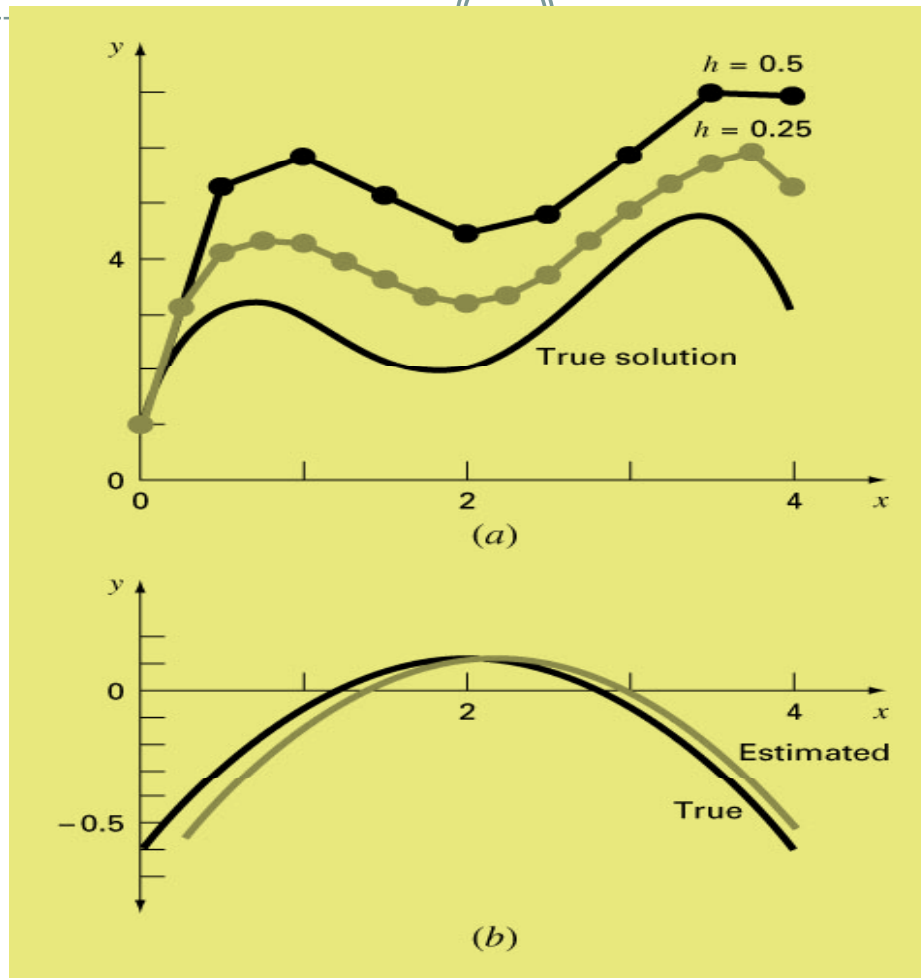


Figure 7.4

# Improvements of Euler's method



- A fundamental source of error in Euler's method is that the derivative at the beginning of the interval is assumed to apply across the entire interval.
- Two simple modifications are available to circumvent this shortcoming:
  - Heun's Method
  - The Midpoint (or Improved Polygon) Method

## Heun's Method/

- One method to improve the estimate of the slope involves the determination of two derivatives for the interval:
  - At the initial point
  - At the end point
- The two derivatives are then averaged to obtain an improved estimate of the slope for the entire interval.

$$\text{Predictor : } y_{i+1}^0 = y_i + f(x_i, y_i)h$$

$$\text{Corrector : } y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2} h$$

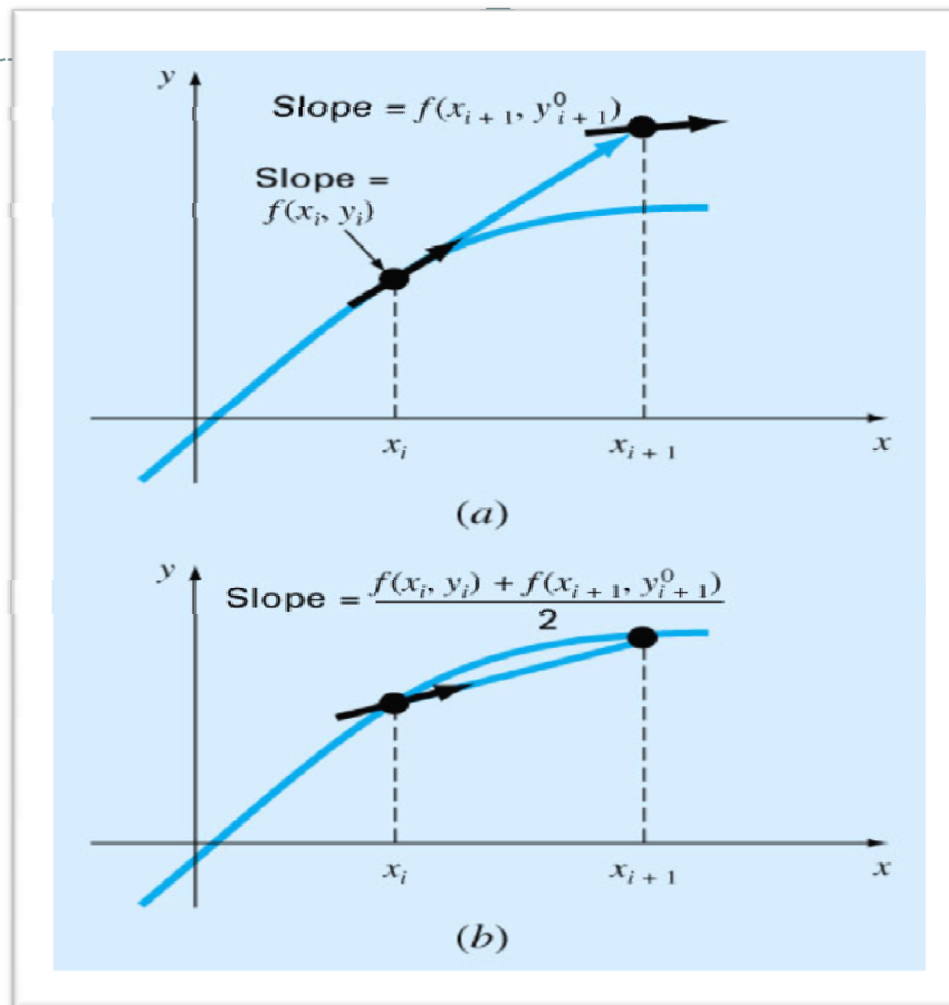


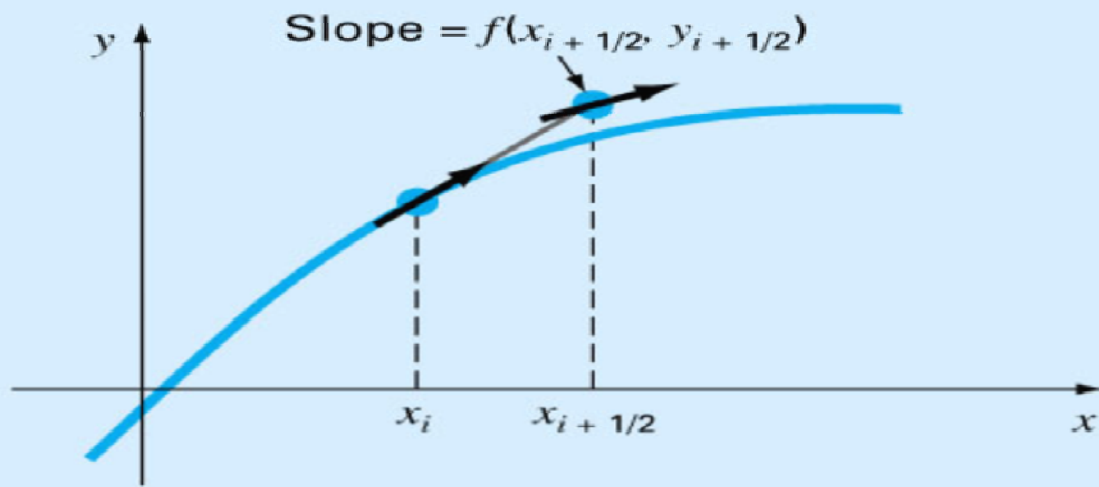
Figure 7.5



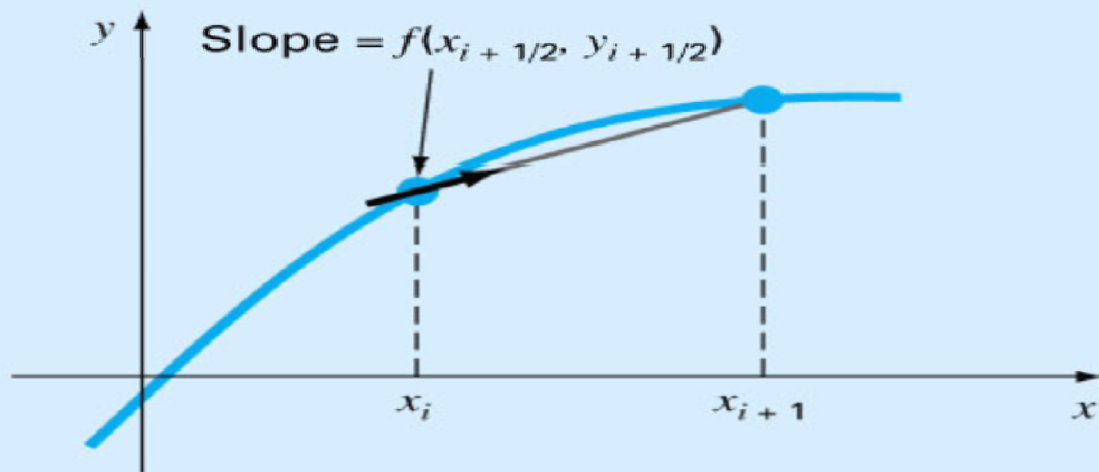
## The Midpoint (or Improved Polygon) Method/

- Uses Euler's method to predict a value of  $y$  at the midpoint of the interval:

$$y_{i+1} = y_i + f(x_{i+1/2}, y_{i+1/2})h$$



(a)



(b)

Figure 7.6



# Runge-Kutta Methods (RK)

- Runge-Kutta methods achieve the accuracy of a Taylor series approach without requiring the calculation of higher derivatives.

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h$$

$$\phi = a_1k_1 + a_2k_2 + \cdots + a_nk_n \quad \text{Increment function}$$

$a$ 's = constants

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h) \quad \text{p's and q's are constants}$$

$$k_3 = f(x_i + p_3h, y_i + q_{21}k_1h + q_{22}k_2h)$$

⋮

$$k_n = f(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + \cdots + q_{n-1,n-1}k_{n-1}h)$$

- $k$ 's are recurrence functions. Because each  $k$  is a functional evaluation, this recurrence makes RK methods efficient for computer calculations.
- Various types of RK methods can be devised by employing different number of terms in the increment function as specified by  $n$ .
- First order RK method with  $n=1$  is in fact Euler's method.
- Once  $n$  is chosen, values of  $a$ 's,  $p$ 's, and  $q$ 's are evaluated by setting general equation equal to terms in a Taylor series expansion.

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

- Values of  $a_1$ ,  $a_2$ ,  $p_1$ , and  $q_{11}$  are evaluated by setting the second order equation to Taylor series expansion to the second order term. Three equations to evaluate four unknowns constants are derived.

$$\text{We have : } y_{i+1} = y_i + (a_1k_1 + a_2k_2)h$$

$$\text{However } y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2$$

$$\text{But } f'(x_i, y_i) = \frac{\partial f(x_i, y_i)}{\partial x} + \frac{\partial f(x_i, y_i)}{\partial y} \frac{dy}{dx}$$

$$\text{Then } y_{i+1} = y_i + f(x_i, y_i)h + \left[ \frac{\partial f(x_i, y_i)}{\partial x} + \frac{\partial f(x_i, y_i)}{\partial y} \frac{dy}{dx} \right] \frac{h^2}{2!}$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$$

$$\text{We now expand } k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$$

$$k_2 = f(x_i, y_i) + \frac{\partial f(x_i, y_i)}{\partial x} p_1h + \frac{\partial f(x_i, y_i)}{\partial y} q_{11}k_1h$$

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

$$y_{i+1} = y_i + \left\{ a_1 f(x_i, y_i) + a_2 \left( f(x_i, y_i) + \frac{\partial f(x_i, y_i)}{\partial x} p_1 h + \frac{\partial f(x_i, y_i)}{\partial y} q_{11} k_1 h \right) \right\} h$$

$$y_{i+1} = y_i + \underbrace{a_1 h f(x_i, y_i) + a_2 h f(x_i, y_i)}_{\substack{\text{---} \\ \searrow \\ \text{---}}} + a_2 p_1 h^2 \frac{\partial f(x_i, y_i)}{\partial x} + a_2 q_{11} f(x_i, y_i) h^2 \frac{\partial f(x_i, y_i)}{\partial y}$$

$$y_{i+1} = y_i + f(x_i, y_i)h + \left[ \frac{\partial f(x_i, y_i)}{\partial x} + \frac{\partial f(x_i, y_i)}{\partial y} f(x_i, y_i) \right] \frac{h^2}{2!}$$

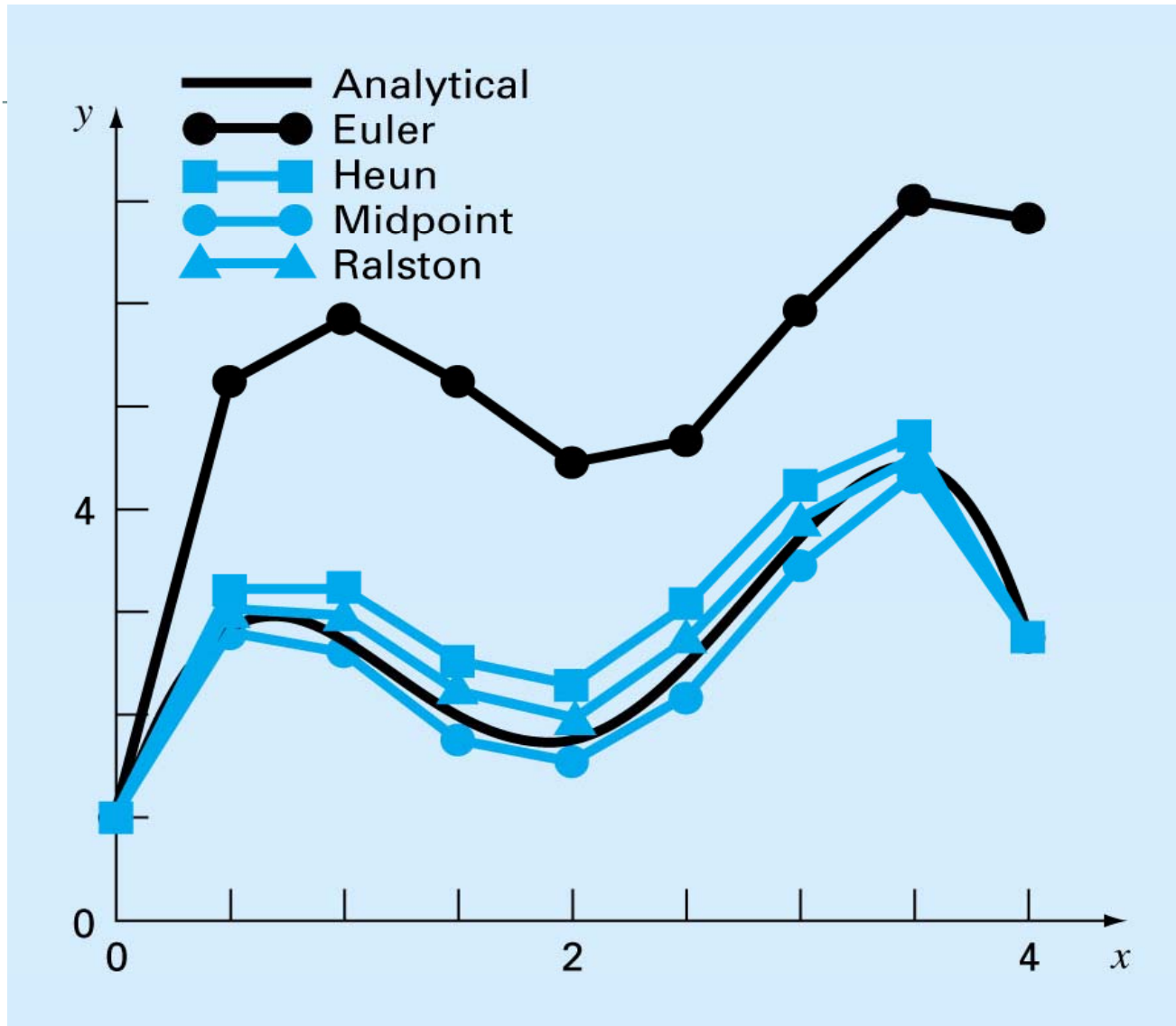
$$a_1 + a_2 = 1$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

- Because we can choose an infinite number of values for  $a_2$ , there are an infinite number of second-order RK methods.
- Every version would yield exactly the same results if the solution to ODE were quadratic, linear, or a constant.
- However, they yield different results if the solution is more complicated (typically the case).
- Three of the most commonly used methods are:
  - Huen Method with a Single Corrector ( $a_2=1/2$ )
  - The Midpoint Method ( $a_2=1$ )
  - Raltson's Method ( $a_2=2/3$ )

Figure 7.7



# Partial Differential Equations

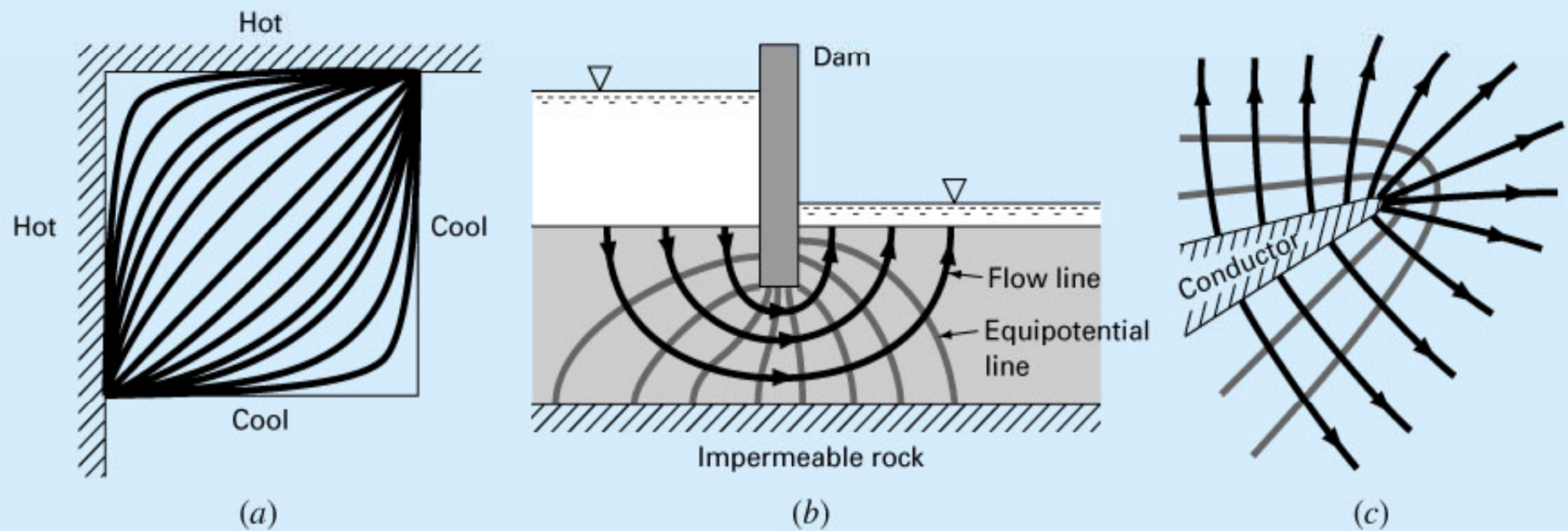


Figure 8.1

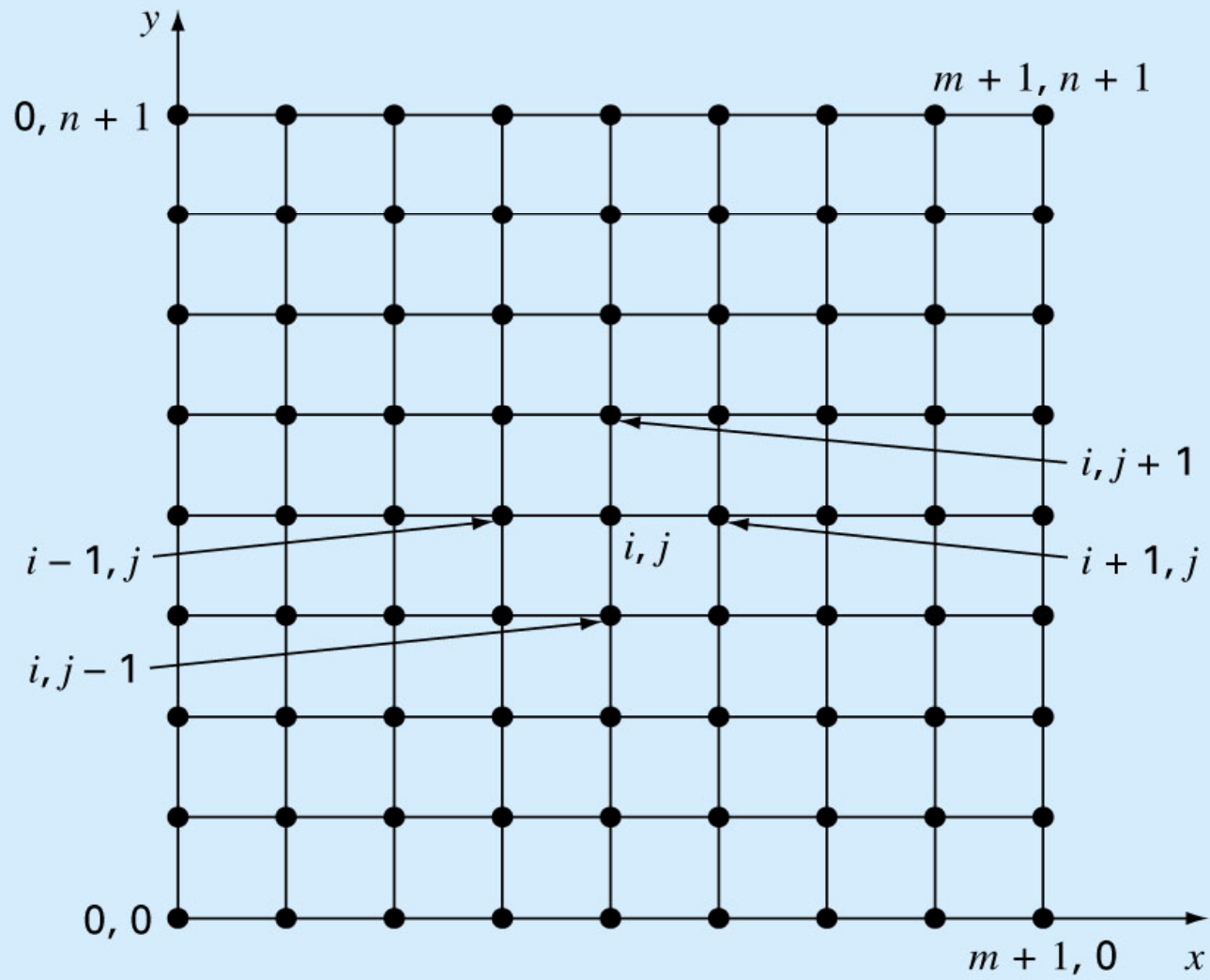
# Finite Difference: Elliptic Equations

## Solution Technique

- Elliptic equations in engineering are typically used to characterize steady-state, boundary value problems.
- For numerical solution of elliptic PDEs, the PDE is transformed into an algebraic difference equation.
- Because of its simplicity and general relevance to most areas of engineering, we will use a heated plate as an example for solving elliptic PDEs.



Figure 8.2



## The Laplacian Difference Equations/

194

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

*Laplace Equation*

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \quad O[\Delta(x)^2]$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} \quad O[\Delta(y)^2]$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0$$

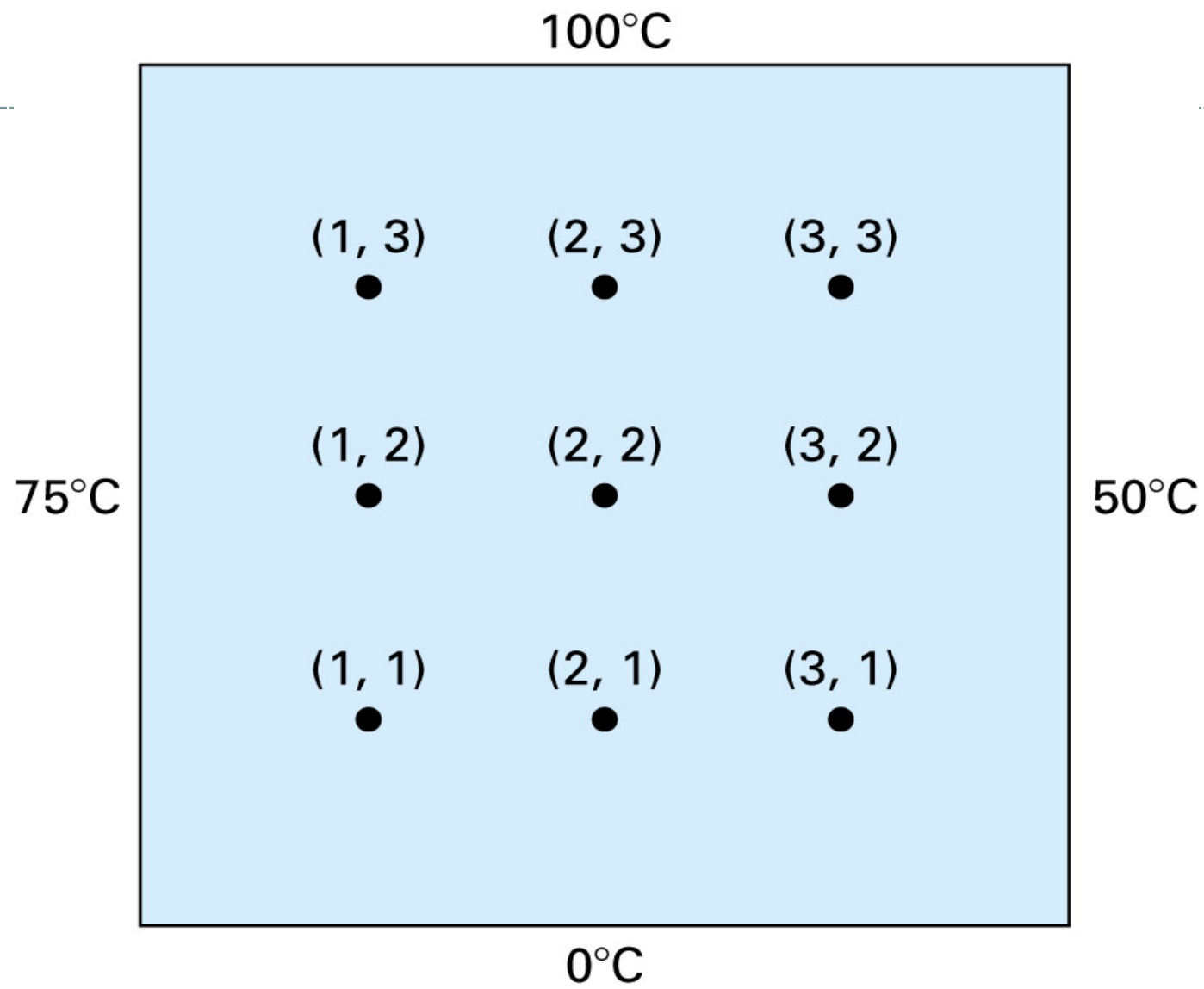
$$\Delta x = \Delta y$$

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

*Laplacian difference equation.*

**Holds for all interior points**

Figure 8.3



- In addition, boundary conditions along the edges must be specified to obtain a unique solution.
- The simplest case is where the temperature at the boundary is set at a fixed value, *Dirichlet boundary condition*.
- A balance for node (1,1) is:

$$T_{21} + T_{01} + T_{12} + T_{10} - 4T_{11} = 0$$

$$T_{01} = 75$$

$$T_{10} = 0$$

$$-4T_{11} + T_{12} + T_{21} = 0$$

- Similar equations can be developed for other interior points to result a set of simultaneous equations.

- The result is a set of nine simultaneous equations with nine unknowns:

$$\begin{array}{rcccccccc}
 4T_{11} & -T_{21} & & -T_{12} & & & & & =75 \\
 -T_{11} & +4T_{21} & -T_{13} & & -T_{22} & & & & =0 \\
 & -T_{21} & +4T_{31} & & & -T_{32} & & & =50 \\
 -T_{11} & & & +4T_{12} & -T_{22} & & -T_{13} & & =75 \\
 & -T_{21} & & -T_{12} & +4T_{22} & -T_{32} & & -T_{23} & =0 \\
 & & -T_{31} & & -T_{22} & +4T_{32} & & -T_{33} & =50 \\
 & & & -T_{12} & & & +4T_{13} & -T_{23} & =175 \\
 & & & & -T_{22} & & -T_{13} & +4T_{23} & -T_{33} & =100 \\
 & & & & & -T_{32} & & -T_{23} & +4T_{33} & =150
 \end{array}$$