ENGINEERING MATHEMATICS II

010.141

Byeng Dong Youn (윤병동)

CHAP. 19
Numerical Methods in General

ABSTRACT OF CHAP. 19

- ***Part E: Numerical methods provide the transition from the mathematical model to an algorithm, which is a detailed stepwise recipe for solving a problem of the indicated kind to be programmed on your computer*.**

- ***Chapter 19 on numerics begins with an explanation of some general concepts, interpolations, numerical integration and differentiation.***
  - Methods for solving equations (19.2), interpolation methods including splines (19.3 and 19.4), and numerical integration and differentiation (19.5)

CHAP. 19.1 INTRODUCTION

Steps and important issues of numerical methods.

STEPS IN NUMERICAL METHODS

- Methods for solving problems numerically on a computer
- **Steps**:
  - Modeling
  - Choice of a numerical method, Programming
  - Doing the computation
  - Interpreting the results

Algorithm

Numeric methods can be formulated as algorithms. An **algorithm** is a step-by-step procedure that states a numeric method in a form (a **"pseudocode"**) understandable to humans. (Turn pages to see what algorithms look like.) The algorithm is then used to write a program in a programming language that the computer can understand so that it can execute the numeric method. Important algorithms follow in the next sections. For routine tasks your CAS or some other software system may contain programs that you can use or include as parts of larger programs of your own.

STABILITY

Stability

Stability. To be useful, an algorithm should be **stable;** that is, small changes in the initial data should cause only small changes in the final results. However, if small changes in the initial data can produce large changes in the final results, we call the algorithm **unstable**.

This *"numeric instability,"* which in most cases can be avoided by choosing a better algorithm, must be distinguished from *"mathematical instability"* of a problem, which is called *"ill-conditioning,"* a concept we discuss in the next section.

Some algorithms are stable only for certain initial data, so that one must be careful in such a case.

ACCURACY

Errors of Numeric Results

Final results of computations of unknown quantities generally are **approximations;** that is, they are not exact but involve errors. Such an error may result from a combination of the following effects. **Roundoff errors** result from rounding, as discussed on p. 782. **Experimental errors** are errors of given data (probably arising from measurements). **Truncating errors** result from truncating (prematurely breaking off), for instance, if we replace a Taylor series with the sum of its first few terms. These errors depend on the computational method used and must be dealt with individually for each method. ["Truncating" is sometimes used as a term for chopping off (see before), a terminology that is not recommended.]

Formulas for Errors. If $\tilde{a}$ is an approximate value of a quantity whose exact value is a, we call the difference

(4) $$\epsilon = a - \tilde{a}$$

the **error** of $\tilde{a}$. Hence

(4*) $$a = \tilde{a} + \epsilon, \qquad \text{True value} = \text{Approximation} + \text{Error}.$$

ACCURACY

The **relative error** ϵ_r of $\tilde{a}$ is defined by

(5)
$$\epsilon_r = \frac{\epsilon}{a} = \frac{a - \tilde{a}}{a} = \frac{\text{Error}}{\text{True value}} \qquad (a \neq 0) .$$

This looks useless because a is unknown. But if $|\epsilon|$ is much less than $|\tilde{a}|$, then we can use $\tilde{a}$ instead of a and get

(5′)
$$\epsilon_r \approx \frac{\epsilon}{\tilde{a}} .$$

This still looks problematic because ϵ is unknown—if it were known, we could get $a = \tilde{a} + \epsilon$ from (4) and we would be done. But what one often can obtain in practice is an **error bound** for $\tilde{a}$, that is, a number β such that

$$|\epsilon| \leqq \beta, \qquad \text{hence} \qquad |a - \tilde{a}| \leqq \beta .$$

This tells us how far away from our computed $\tilde{a}$ the unknown a can at most lie. Similarly, for the relative error, an error bound is a number β_r such that

$$|\epsilon_r| \leqq \beta_r, \qquad \text{hence} \qquad \left|\frac{a - \tilde{a}}{a}\right| \leqq \beta_r .$$

ERROR PROPAGATION

Error Propagation

This is an important matter. It refers to how errors at the beginning and in later steps (roundoff, for example) propagate into the computation and affect accuracy, sometimes very drastically. We state here what happens to error bounds. Namely, bounds for the *error* add under addition and subtraction, whereas bounds for the *relative error* add under multiplication and division. You do well to keep this in mind.

THEOREM 1

a. *In addition and subtraction, an error bound for the results is given by the sum of the error bounds for the terms.*

b. *In multiplication and division, an error bound for the relative error of the results is given (approximately) by the sum of the bounds for the relative errors of the given numbers.*

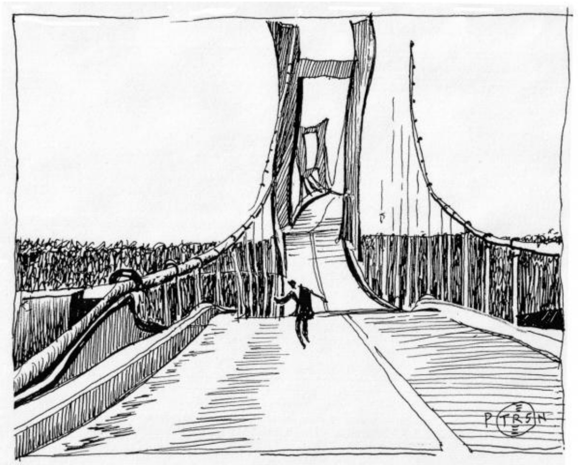
HOMEWORK IN 19.1

- HW1. Problems 14

CHAP. 19.2
SOLUTION OF EQUATIONS BY ITERATION

Finding solution of an equation using iterative steps.

Nonlinear Equations in Engineering Fields



$f(x) = \omega_n(x) - \omega \neq 0$
x: bridge design variables

To design a safe bridge , an excitation frequency must be different from its natural frequency.

$f(x) = T(x) - T^* < 0$
x: battery design variables

To design a safe battery, a temperature level must be smaller than a marginal temperature.

$f(x) = \sigma(x) - S < 0$
x: bridge gusset plate

To design a safe bridge, a stress level at a critical bridge element must be smaller than its strength.

SOLUTION OF EQUATIONS BY ITERATION

- Solving equation $f(x) = 0$
- **Methods**:
  - Fixed – Point Iteration
  - Newton's Method
  - Secant Method

FIXED-POINT ITERATION FOR SOLVING f(x) = 0

- **Idea**: transform $f(x) = 0$ into $x = g(x)$

- **Steps**:
  1. Choose x_0
  2. Compute $x_1 = g(x_0)$, $x_2 = g(x_1)$, …, $x_{n+1} = g(x_n)$

- A solution of $x = g(x)$ is called a **fixed point**

- Depending on the initial value chosen (x_0), the related sequences may converge or diverge

FIXED-POINT ITERATION FOR SOLVING f(x) = 0

- **Example**: $f(x) = x^2 - 3x + 1 = 0$

$$\text{Solutions} = \begin{cases} 2.618034 \\ 0.381966 \end{cases}$$

The equation may be written

(4a) $$x = g_1(x) = \frac{1}{3}(x^2 + 1), \qquad \text{thus} \qquad x_{n+1} = \frac{1}{3}(x_n^2 + 1).$$

If we choose $x_0 = 1$, we obtain the sequence (Fig. 423a; computed with 6S and then rounded)

$$x_0 = 1.000, \quad x_1 = 0.667, \quad x_2 = 0.481, \quad x_3 = 0.411, \quad x_4 = 0.390, \cdots$$

which seems to approach the smaller solution. If we choose $x_0 = 2$, the situation is similar. If we choose $x_0 = 3$, we obtain the sequence (Fig. 423a, upper part)

$$x_0 = 3.000, \quad x_1 = 3.333, \quad x_2 = 4.037, \quad x_3 = 5.766, \quad x_4 = 11.415, \cdots$$

which diverges.

FIXED-POINT ITERATION FOR SOLVING f(x) = 0

Our equation may also be written (divide by x)

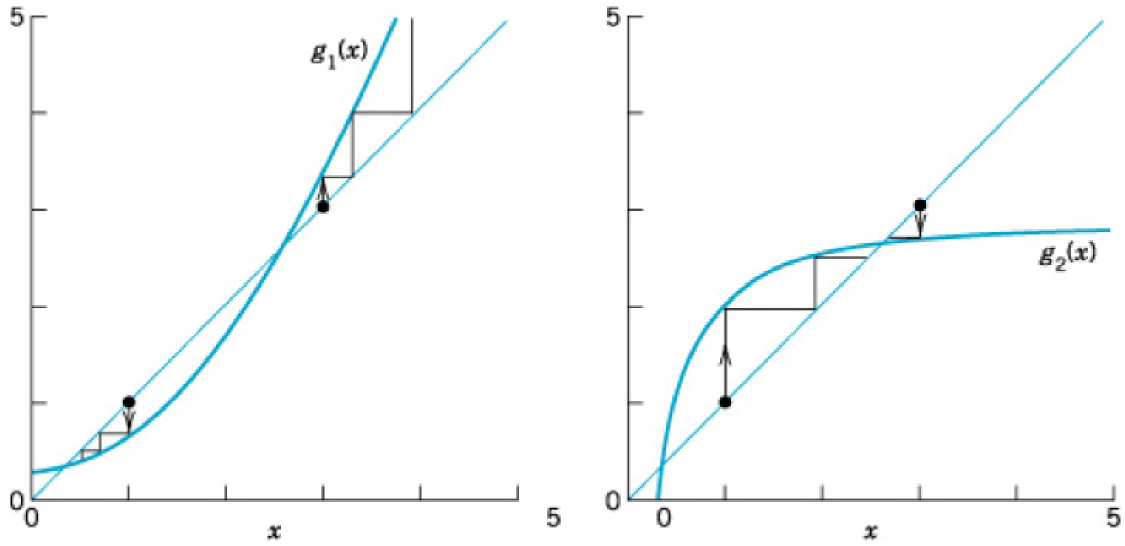
(4b) $$x = g_2(x) = 3 - \frac{1}{x}, \qquad \text{thus} \qquad x_{n+1} = 3 - \frac{1}{x_n},$$

and if we choose $x_0 = 1$, we obtain the sequence (Fig. 423b)

$$x_0 = 1.000, \quad x_1 = 2.000, \quad x_2 = 2.500, \quad x_3 = 2.600, \quad x_4 = 2.615, \cdots$$

which seems to approach the larger solution. Similarly, if we choose $x_0 = 3$, we obtain the sequence (Fig. 423b)

$$x_0 = 3.000, \quad x_1 = 2.667, \quad x_2 = 2.625, \quad x_3 = 2.619, \quad x_4 = 2.618, \cdots$$



NEWTON'S METHOD FOR SOLVING f(x) = 0

- f must have a continuous derivative f '
- The method is simple and fast

$$\tan\beta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1}$$

$$\Downarrow$$

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

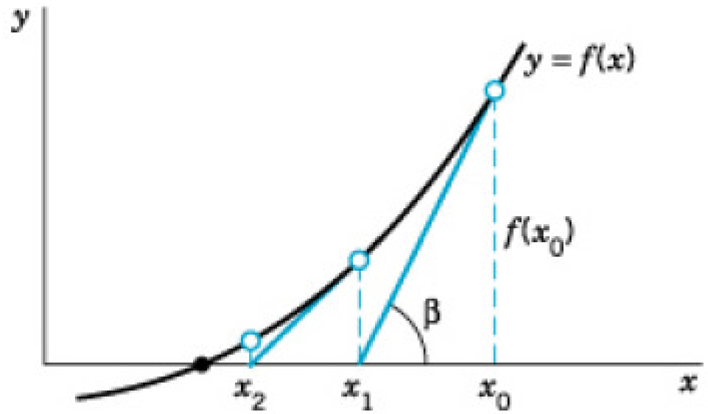


Fig. 425. Newton's method

NEWTON'S METHOD FOR SOLVING f(x) = 0

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

$$\vdots$$

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

ALGORITHM

ALGORITHM NEWTON (f, f', x_0, ϵ, N)

This algorithm computes a solution of $f(x) = 0$ given an initial approximation x_0 (starting value of the iteration). Here the function $f(x)$ is continuous and has a continuous derivative $f'(x)$.

INPUT: f, f', initial approximation x_0, tolerance $\epsilon > 0$, maximum number of iterations N.

OUTPUT: Approximate solution x_n ($n \leqq N$) or message of failure.

For $n = 0, 1, 2, \cdots, N - 1$ do:

1 Compute $f'(x_n)$.

2 If $f'(x_n) = 0$ then OUTPUT "Failure". Stop.

[Procedure completed unsuccessfully]

3 Else compute

(5)
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

4 If $|x_{n+1} - x_n| \leqq \epsilon |x_n|$ then OUTPUT x_{n+1}. Stop.

[Procedure completed successfully]

End

5 OUTPUT "Failure". Stop.

[Procedure completed unsuccessfully after N iterations]

End NEWTON

NEWTON'S METHOD FOR SOLVING f(x) = 0

Example: Find the positive solution of f(x) = 2 sin x – x = 0

Solution:

Setting $f(x) = x - 2\sin x$, we have $f'(x) = 1 - 2\cos x$, and (5) gives

$$x_{n+1} = x_n - \frac{x_n - 2\sin x_n}{1 - 2\cos x_n} = \frac{2(\sin x_n - x_n \cos x_n)}{1 - 2\cos x_n} = \frac{N_n}{D_n}.$$

From the graph of f we conclude that the solution is near $x_0 = 2$. We compute:

n	x_n	N_n	D_n	x_{n+1}
0	2.00000	3.48318	1.83229	1.90100
1	1.90100	3.12470	1.64847	1.89552
2	1.89552	3.10500	1.63809	1.89550
3	1.89550	3.10493	1.63806	1.89549

$x_4 = 1.89549$ is exact to 5D since the solution to 6D is 1.895 494.

SECANT METHOD FOR SOLVING f(x) = 0

Newton's method is powerful but disadvantageous because it is difficult to obtain f '. The secant method approximates f '.

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

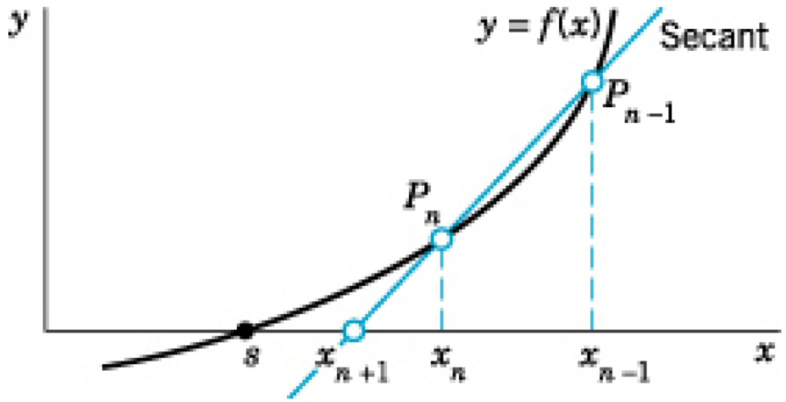


Fig. 426. Secant method

SECANT METHOD FOR SOLVING f(x) = 0

- Newton's Method

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

- Secant

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

$$\Downarrow$$

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

SECANT METHOD FOR SOLVING f(x) = 0

Find the positive solution of $f(x) = x - 2\sin x = 0$ by the secant method, starting from $x_0 = 2$, $x_1 = 1.9$.

Solution:

Here, (10) is

$$x_{n+1} = x_n - \frac{(x_n - 2\sin x_n)(x_n - x_{n-1})}{x_n - x_{n-1} + 2(\sin x_{n-1} - \sin x_n)} = x_n - \frac{N_n}{D_n}.$$

Numerical values are:

n	x_{n-1}	x_n	N_n	D_n	$x_{n+1} - x_n$
1	2.000 000	1.900 000	−0.000 740	−0.174 005	−0.004 253
2	1.900 000	1.895 747	−0.000 002	−0.006 986	−0.000 252
3	1.895 747	1.895 494	0		0

$x_3 = 1.895\,494$ is exact to 6D. See Example 4.

HOMEWORK IN 19.2

- HW1. Problems 2
- HW2. Problems 10
- HW3. Problems 11
- HW4. Problems 21

CHAP. 19.3
INTERPOLATION

Finding (approximate) values of a function f(x) for an x between different x-values x_0, x_1, … , x_n at which the values of f(x) are given.

INTERPOLATION

- Function f(x) is unknown
- Some values of f(x) are known (f_1, f_2, …, f_n)
- **Idea**: Find a polynomial $p_n(x)$ that is an approximation of f(x)

$$\boxed{p_n(x_0)=f_0,\ p_n(x_1)=f_1,\cdots,p_n(x_n)=f_n}$$

- **Lagrange interpolation**
  - Linear
  - Quadratic
  - General
- **Newton's interpolation**
  - Divided difference
  - Forward difference
  - Backward difference
- **Splines**

LINEAR LAGRANGE INTERPOLATION

- Use 2 known values of f(x) $\rightarrow$ f_0, f_1

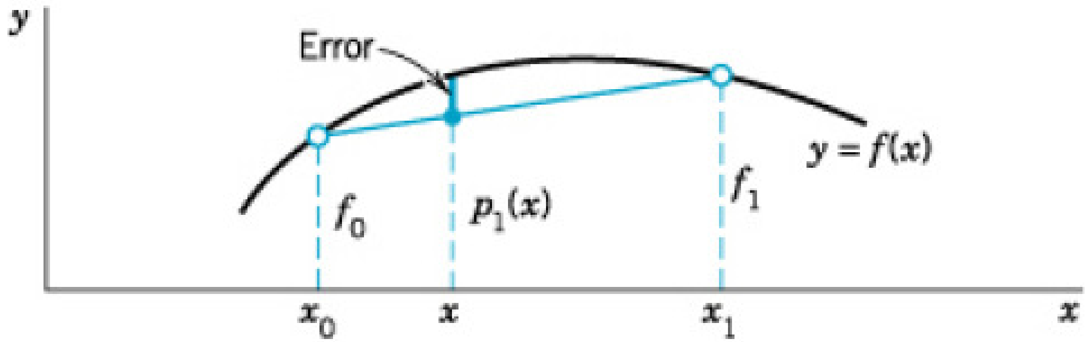


Fig. 428. Linear interpolation

p_1 is the linear Lagrange polynomial.

LINEAR LAGRANGE INTERPOLATION

- $p_1(x) = L_0(x)f_0 + L_1(x)f_1$
- L_0 and $L_1$are linear polynomials (weight functions).

$$L_0(x) = \begin{cases} 1 \text{ if } & x = x_0 \\ 0 \text{ if } & x = x_1 \end{cases} \qquad L_1(x) = \begin{cases} 0 \text{ if } & x = x_0 \\ 1 \text{ if } & x = x_1 \end{cases}$$

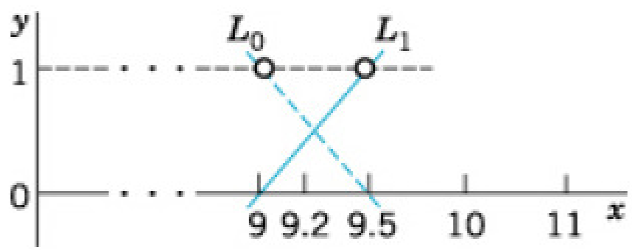


Fig. 429. L_0 and L_1 in Example 1

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

$$\boxed{p_1(x) = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1}$$

LINEAR LAGRANGE INTERPOLATION

Example: Compute ln 9.2 from ln 9.0 = 2.1972 and ln 9.5 = 2.2513 by linear Lagrange interpolation

Solution:

$$x_0 = 9.0, \quad x_1 = 9.5, \quad f_0 = \ln 9.0, \quad f_1 = \ln 9.5$$

$$L_0(9.2) = \frac{9.2 - 9.5}{9.0 - 9.5} = 0.6 \qquad L_1(9.2) = \frac{9.2 - 9.0}{9.5 - 9.0} = 0.4$$

$$\begin{aligned} \ln 9.2 &\approx p_1(9.2) \\ &= L_0(9.2) f_0 + L_1(9.2) f_1 \\ &= 0.6(2.1972) + 0.4(2.2513) \\ &= 2.2188 \end{aligned}$$

$$\text{Error}: \quad \ln 9.2 - p_1(9.2) = 2.2192 - 2.2188 = 0.0004$$

QUADRATIC LAGRANGE INTERPOLATION

- Use of 3 known values of f(x) $\rightarrow f_0, f_1, f_2$
- Approximation of f(x) by a second-degree polynomial

$$p_2(x) = L_0(x)f_0 + L_1(x)f_1 + L_2(x)f_2$$

$$L_0 = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$L_1 = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$L_2 = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

QUADRATIC LAGRANGE INTERPOLATION

Example: Compute ln 9.2 for f_0 (x_0 = 9.0) = ln 9.0, f_1 (x_1 = 9.5) = ln 9.5, f_2 (x_2 = 11.0) = ln 11.0

Solution:

$$L_0(x) = x^2 - 20.5x + 104.5$$

$$L_1(x) = -\frac{1}{0.75}\left(x^2 - 20x + 99\right)$$

$$L_2(x) = \frac{1}{3}\left(x^2 - 18.5x + 85.5\right)$$

$$\ln 9.2 \approx p_2(9.2) = 2.2192$$

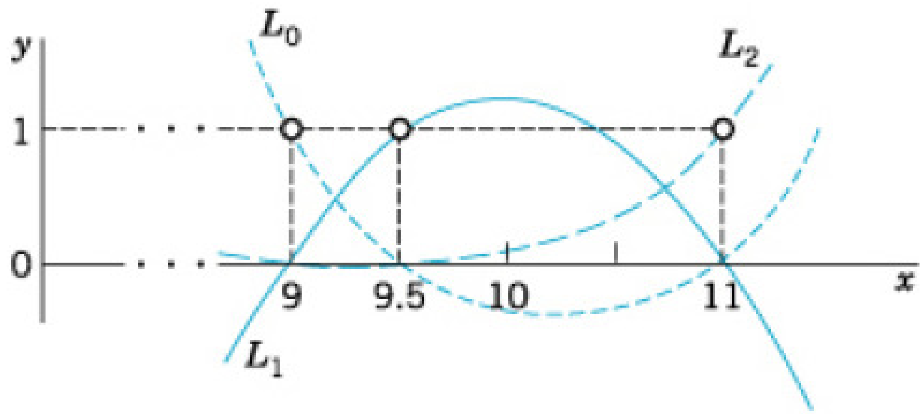


Fig. 430. L_0, L_1, L_2 in Example 2

GENERAL LAGRANGE INTERPOLATION

$$f(x) \approx p_n(x) = \sum_{\mathrm{k}=0}^{n} \mathrm{L_k}(x) f_k$$

$$= \sum_{\mathrm{k}=0}^{n} \frac{l_{\mathrm{k}}(x)}{l_{\mathrm{k}}(x_k)} f_k$$

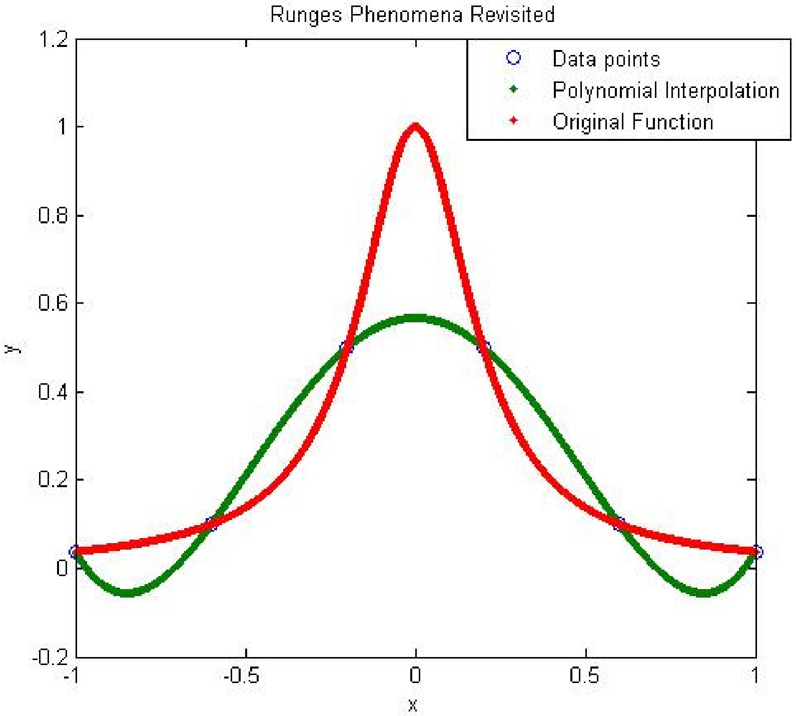
$$l_{\mathrm{k}}(x) = (x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)$$

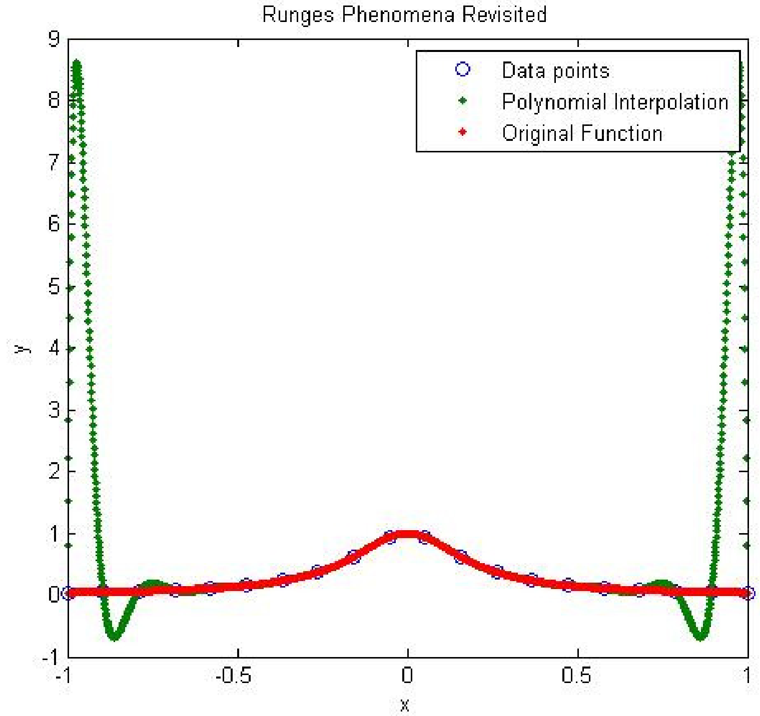
NEWTON'S INTERPOLATION

- More appropiate for computation
- Level of accuracy can be easily improved by adding new terms that increase the degree of the polynomial
- Divided difference
- Forward difference
- Backward difference

Is High-Order Polynomial a Good Idea?

f(x) = 1/(1+25x^2)





HOMEWORK IN 19.3

- HW1. Problems 3
- HW2. Problems 8

CHAP. 19.4
SPLINE INTERPOLATION

The undesirable oscillations are avoided by the method of splines initiated by I.J. Schoenberg in 1946 (Applied Mathematics 4, pp45-99, 112-141). This method is widely used in practice. It also laid the foundation for much of modern CAD (Computer Aided Design).

SPLINES

- Method of interpolation used to avoid numerical instability

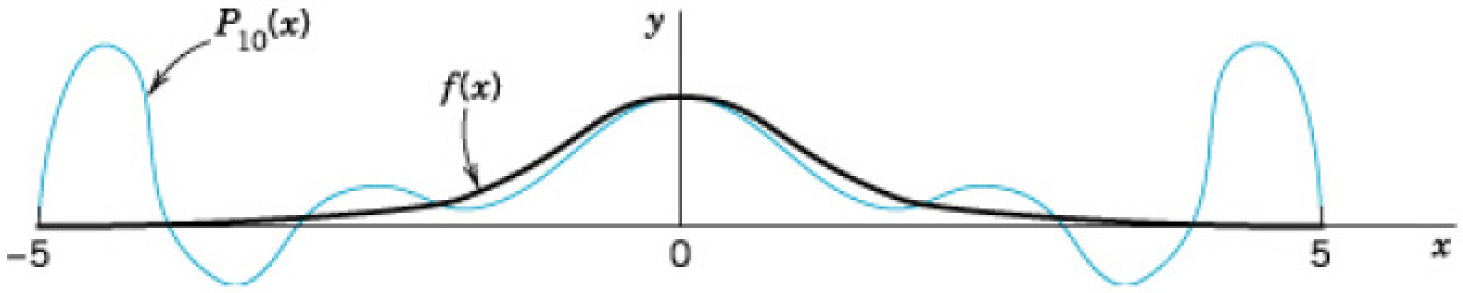


Fig. 431. Runge's example $f(x) = 1/(1 + x^2)$ and interpolating polynomial $P_{10}(x)$

- **Idea**: given an interval [a, b] where the high-degree polynomial can oscillate considerable, we subdivide [a, b] in several smaller intervals and use several low-degree polynomials (which cannot oscillate much)

SPLINES (cont)

- In an interval $x \in [x_j, x_{j+1}]$, $j = 0, \ldots, n-1$

$$\boxed{p_j(x) = a_{j_0} + a_{j_1}(x - x_j) + a_{j_2}(x - x_j)^2 + a_{j_3}(x - x_j)^3}$$

where

$$a_{j0} = p_j(x_j) = f_j$$

$$a_{j1} = p'_j(x_j) = k_j$$

$$a_{j2} = \frac{1}{2} p''_j(x_j) = \frac{3}{h^2}(f_{j+1} - f_j) - \frac{1}{h}(k_{j+1} + 2k_j)$$

$$a_{j3} = \frac{2}{h^3}(f_j - f_{j+1}) + \frac{1}{h^2}(k_{j+1} + k_j)$$

SPLINES (cont)

- Let $(x_0,f_0), (x_1,f_1), \ldots , (x_n,f_n)$.
- k_0, k_n are two given numbers.
- $k_1, k_2, \ldots, k_{n-1}$ are determined by a linear system of n-1 equations:

$$k_{j-1} + 4k_j + k_{j+1} = \frac{3}{h}\left(f_{j+1} - f_{j-1}\right)$$

$$p'_j\left(x_j\right) = k_j, \quad p'_j\left(x_j + 1\right) = k_{j+1} \quad \left(\mathrm{j} = 0, 1, \ldots, \mathrm{n} - 1\right)$$

- h is the distance between nodes

$$x_n = x_0 + nh$$

SPLINES (cont)

- Clamped conditions

$$g'(x_0) = f'(x_0), \quad g'(x_n) = f'(x_n)$$

- Free/natural conditions

$$g''(x_0) = 0, \quad g''(x_n) = 0$$

- **Example**:

  Interpolate $f(x) = x^4$ on interval $x \in [-1, 1]$ by cubic spline in partitions $x_0 = -1$, $x_1 = 0$, $x_2 = 1$, satisfying clamped conditions

$$g'(-1) = f'(-1), \quad g'(1) = f'(1)$$

SPLINES (cont)

the given data are $f_0 = f(-1) = 1, f_1 = f(0) = 0, f_2 = f(1) = 1.$

$$q_0(x) = a_{00} + a_{01}(x+1) + a_{02}(x+1)^2 + a_{03}(x+1)^3 \quad (-1 \leqq x \leqq 0)$$

$$q_1(x) = a_{10} + a_{11}x + a_{12}x^2 + a_{13}x^3 \quad (0 \leqq x \leqq 1)$$

$$k_0 + 4k_1 + k_2 = 3(f_2 - f_0).$$

Here $f_0 = f_2 = 1$ (the value of x^4 at the ends) and $k_0 = -4, k_2 = 4.$

$$-4 + 4k_1 + 4 = 3(1-1) = 0, \qquad k_1 = 0$$

$$a_{00} = f_0 = 1, a_{01} = k_0 = -4$$

$$a_{02} = \frac{3}{1^2}(f_1 - f_0) - \frac{1}{1}(k_1 + 2k_0) = 3(0-1) - (0-8) = \qquad 5$$

$$a_{03} = \frac{2}{1^3}(f_0 - f_1) + \frac{1}{1^2}(k_1 + k_0) = 2(1-0) + (0-4) = -2.$$

SPLINES (cont)

Similarly, for the coefficients of q_1 we obtain from (13) the values $a_{10} = f_1 = 0$, $a_{11} = k_1 = 0$, and

$$a_{12} = 3(f_2 - f_1) - (k_2 + 2k_1) = 3(1 - 0) - (4 + 0) = -1$$

$$a_{13} = 2(f_1 - f_2) + (k_2 + k_1) = 2(0 - 1) + (4 + 0) = 2\,.$$

This gives the polynomials of which the spline $g(x)$ consists, namely,

$$g(x) = \begin{cases} q_0(x) = 1 - 4(x+1) + 5(x+1)^2 - 2(x+1)^3 = -x^2 - 2x^3 & \text{if} \quad -1 \leqq x \leqq 0 \\ q_1(x) = -x^2 + 2x^3 & \text{if} \quad 0 \leqq x \leqq 1\,. \end{cases}$$

Figure 433 shows $f(x)$ and this spline. Do you see that we could have saved over half of our work by using symmetry?

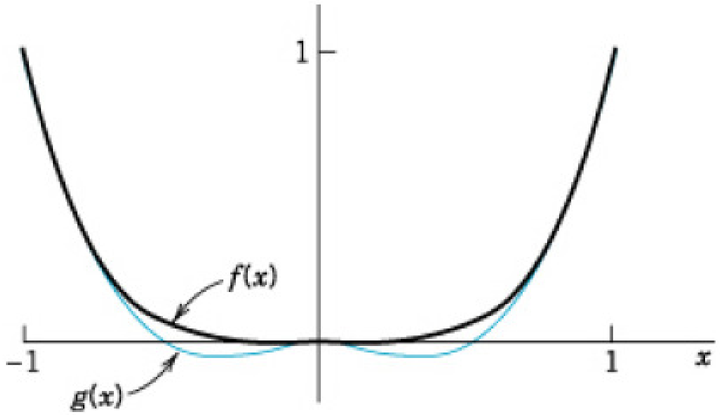


Fig. 433. Function $f(x) = x^4$ and cubic spline $g(x)$ in Example 1

HOMEWORK IN 19.4

- HW1. Problems 11
- HW2. Problems 12

CHAP. 19.5
NUMERICAL INTEGRATION AND DIFFERENTIATION

Evaluating the numerical integration.

$$J = \int_a^b f(x)dx$$

NUMERICAL INTEGRATION

- Numerical evaluation of integrals whose analytical evaluation is too complicated or impossible, or that are given by recorded numerical values

$$\int_a^b f(x)dx$$

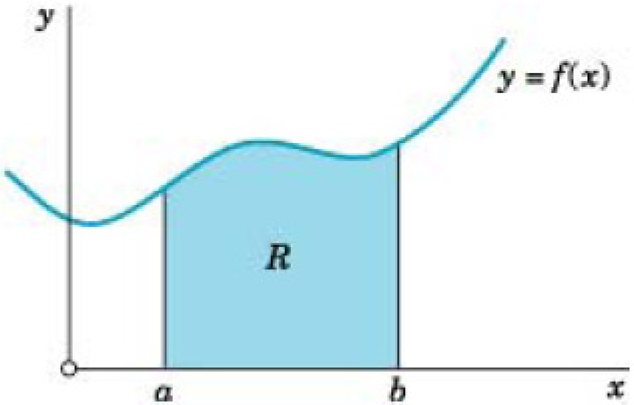


Fig. 437. Geometric interpretation of a definite integral

- Rectangular rule
- Trapezoidal rule
- Simpson's rule
- Gauss integration

RECTANGULAR RULE

- Approximation by n rectangular areas

$$\boxed{J = \int_a^b f(x)dx \approx h\left[f(x^*_1) + f(x^*_2) + \cdots + f(x^*_n)\right]}$$

where

$$h = \frac{b-a}{n}$$

$$x_1 = x_0 + h$$

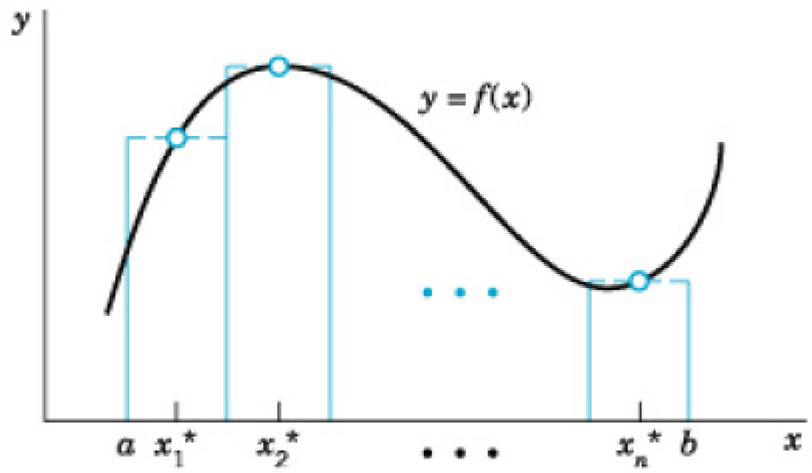


Fig. 438. Rectangular rule

TRAPEZOIDAL RULE

- Approximation by n trapezoidal areas

$$J = \int_a^b f(x)dx \approx h\left[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right]$$

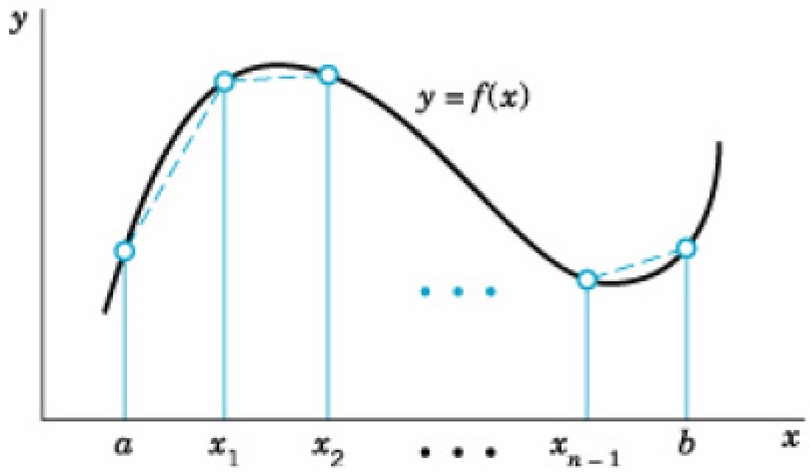


Fig. 439. Trapezoidal rule

TRAPEZOIDAL RULE

Evaluate $J = \int_0^1 e^{-x^2} dx$ by means of (2) with $n = 10$.

Solution:

$J \approx 0.1(0.5 \cdot 1.367\,879 + 6.778\,167) = 0.746\,211$ from Table 19.3.

TABLE 19.3 Computations in Example 1

j	x_j	x_j^2	$e^{-x_j^2}$	
0	0	0	1.000 000	
1	0.1	0.01		0.990 050
2	0.2	0.04		0.960 789
3	0.3	0.09		0.913 931
4	0.4	0.16		0.852 144
5	0.5	0.25		0.778 801
6	0.6	0.36		0.697 676
7	0.7	0.49		0.612 626
8	0.8	0.64		0.527 292
9	0.9	0.81		0.444 858
10	1.0	1.00	0.367 879	
Sums			1.367 879	6.778 167

SIMPSON'S RULE

- Approximation by parabolas using Lagrange polynomials $p_2(x)$
- Interval of integration [a, b] divided into an even number of subintervals

$$h = \frac{b-a}{2m} \quad f_0 = f(x_0) \quad x_1 = a + h \quad x_2 = x_1 + h$$

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m}\right)$$

$$f_j = f(x_j)$$

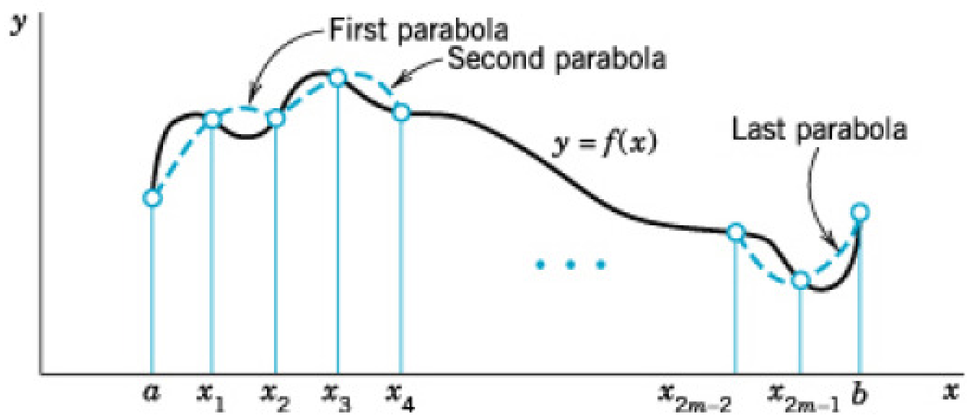


Fig. 440. Simpson's rule

SIMPSON'S RULE

Evaluate $J = \int_0^1 e^{-x^2} dx$ by Simpson's rule with $2m = 10$ and estimate the error.

Solution:

Since $h = 0.1$, Table 19.5 gives

$$J \approx \frac{0.1}{3}(1.367\ 879 + 4 \cdot 3.740\ 266 + 2 \cdot 3.037\ 901) = 0.746\ 825\ .$$

TABLE 19.5 Computations in Example 3

j	x_j	x_j^2	$e^{-x_j^2}$		
0	0	0	1.000 000		
1	0.1	0.01		0.990 050	
2	0.2	0.04			0.960 789
3	0.3	0.09		0.913 931	
4	0.4	0.16			0.852 144
5	0.5	0.25		0.778 801	
6	0.6	0.36			0.697 676
7	0.7	0.49		0.612 626	
8	0.8	0.64			0.527 292
9	0.9	0.81		0.444 858	
10	1.0	1.00	0.367 879		
Sums			1.367 879	3.740 266	3.037 901

SIMPSON'S RULE

ALGORITHM SIMPSON ($a, b, m, f_0, f_1, \cdots, f_{2m}$)

This algorithm computes the integral $J = \int_a^b f(x)\,dx$ from given values $f_j = f(x_j)$ at equidistant $x_0 = a, x_1 = x_0 + h, \cdots, x_{2m} = x_0 + 2mh = b$ by Simpson's rule (7), where $h = (b - a)/(2m)$.

INPUT: $a, b, m, f_0, \cdots, f_{2m}$

OUTPUT: Approximate value $\tilde{J}$ of J

Compute $s_0 = f_0 + f_{2m}$

$$s_1 = f_1 + f_3 + \cdots + f_{2m-1}$$

$$s_2 = f_2 + f_4 + \cdots + f_{2m-2}$$

$$h = (b - a)/2m$$

$$\tilde{J} = \frac{h}{3}(s_0 + 4s_1 + 2s_2)$$

OUTPUT $\tilde{J}$. Stop.

End SIMPSON

ADAPTIVE INTEGRATION

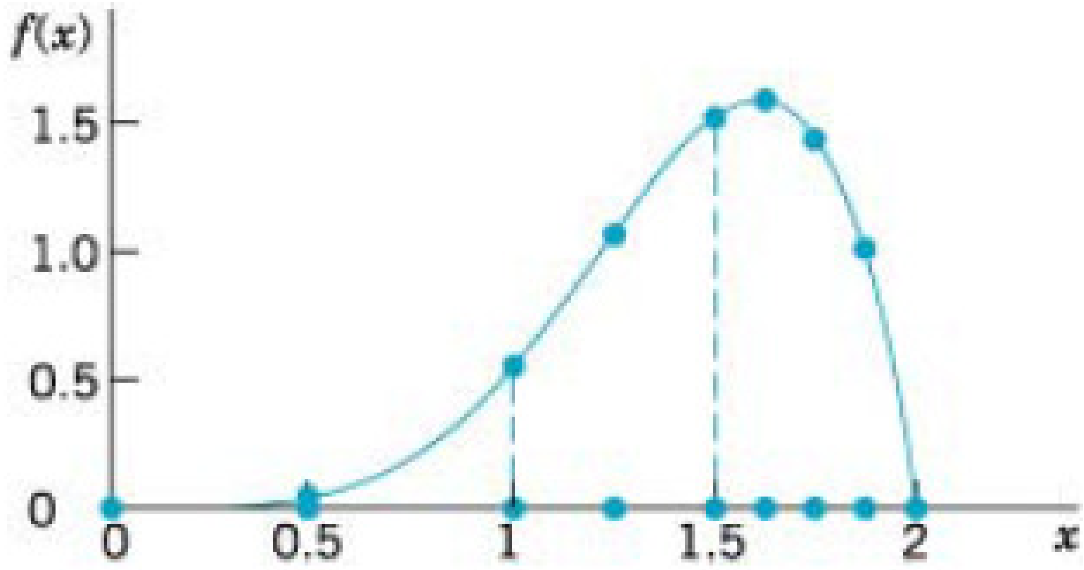


Fig. 441. Adaptive integration in Example 6

GAUSS INTEGRATION

$$\int_a^b f(x)dx = \int_{-1}^{1} f(t)dt \approx \sum_{j=1}^{\infty} A_j f_j$$

where

$$x = \frac{1}{2}[a(t-1)+b(t+1)]$$

$$A_1, \ldots, A_n \quad \Rightarrow \quad \text{coefficients}$$

$$f_j = f(t_j)$$

TABLE 19.7 **Gauss Integration: Nodes t_j and Coefficients A_j**

n	Nodes t_j	Coefficients A_j	Degree of Precision
2	−0.57735 02692	1	3
	0.57735 02692	1	
3	−0.77459 66692	0.55555 55556	5
	0	0.88888 88889	
	0.77459 66692	0.55555 55556	
4	−0.86113 63116	0.34785 48451	7
	−0.33998 10436	0.65214 51549	
	0.33998 10436	0.65214 51549	
	0.86113 63116	0.34785 48451	
5	−0.90617 98459	0.23692 68851	9
	−0.53846 93101	0.47862 86705	
	0	0.56888 88889	
	0.53846 93101	0.47862 86705	
	0.90617 98459	0.23692 68851	

HOMEWORK IN 19.5

- HW1. Problem 5
- HW2. Problem 6
- HW3. Problem 21