UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Chapter 1

Mathematical Preliminaries and Error Analysis

What is scientific computing?

- Design and analysis of algorithms for solving mathematical problems arising in science and engineering numerically
- Also called numerical analysis or computational mathematics
- Distinguishing features of scientific computing
  - ✓ Deals with continuous quantities (e.g., time, distance, velocity, temperature, density, pressure) typically measured by real numbers
  - ✓ Considers effects of approximations

Why scientific computing?

- Predictive simulation of natural phenomena
- Virtual phototyping of engineering designs
- Analyzing data

Well-Posed Problems

- **Problem is *well-posed* (适定的) if solution**
  - exists
  - is unique
  - depends continuously on problem data

Otherwise, problem is *ill-posed.*

- **Even if problem is well posed, solution may still be *sensitive* to input data.**
- **Computational algorithm should not make sensitivity worse**

General Strategy

- **Replace difficult problem by easier one having same or closely related solution**
  - infinite → finite
  - differential → algebraic
  - nonlinear → linear
  - complicated → simple
- **Solution obtained may only *approximate* that of original problem**

Sources of Approximation

- **Before computation**
  - modeling
  - empirical measurements
  - previous computations
- **During computation**
  - truncation or discretization
  - rounding
- **Accuracy of final result reflects all these**
- **Uncertainty in input may be amplified by problem**
- **Perturbations during computation may be amplified by algorithm**

Example: Approximations

- **Computation surface area of Earth using formula $A = 4\pi r^2$ involves several approximations**
  - Earth is modeled as sphere, idealizing its true shape
  - Value for radius is based on empirical measurements and previous computations
  - Value for π requires truncating infinite process
  - Values for input data and results of arithmetic operations are rounded in computer

Numerical Analysis ⟶ Scientific Computing

- Pre-computer era (before~1940)
  - ✓ Foundations and basic methods established by Newton, Euler, Lagrange, Gauss, and many other mathematicians, scientists, and engineers
- Pre-integrated circuit era (1940-1970): *Numerical Analysis*
  - ✓ Programming languages developed for scientific applications
  - ✓ Numerical methods formalized in computer algorithms and software
  - ✓ Floating-point arithmetic developed

Numerical Analysis → Scientific Computing

- Integrated circuit era (since~1970): Scientific Computing
  - ✓ Application problem sizes explode as computing capacity grows exponentially
  - ✓ Computation becomes an essential component of modern scientific research and engineering practice, along with theory and experiment

Mathematical Problems

- Given mathematical relationship $y = f(x)$, typical problems include
  - ✓ Evaluate a function: compute output y for given input x
  - ✓ Solve an equation: find input x that produces given output y
  - ✓ Optimize: find x that yields extreme value of y over given domain
- Specific type of problem and best approach to solving it depend on whether variables and function involved are
  - ✓ discrete or continuous
  - ✓ linear or nonlinear
  - ✓ finite or infinite dimensional
  - ✓ purely algebraic or involve derivatives or integrals

General Problem-Solving Strategy

- Replace difficult problem by easier one having same or closely related solution
  - ✓ infinite dimensional → finite dimensional
  - ✓ differential → algebraic
  - ✓ nonlinear → linear
  - ✓ complicated → simple
- Solution obtained may only approximate that of original problem
- Our goal is to estimate accuracy

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 1-1

Review of Calculus

1. Limits and Continuity

Definition 1.1 A function f defined on a set X of real numbers has the limit L at x_0, written

$$\lim_{x \to x_0} f(x) = L$$

if, given any real number $\varepsilon > 0$, there exists a real number $\delta > 0$ such that $|f(x) - L| < \varepsilon$, whenever $x \in X$ and $0 < |x - x_0| < \delta$. (See Figure 1.1)

Figure 1.1

Definition 1.2 Let f be a function defined on a set X of real numbers and $x_0 \in X$. Then f is continuous at x_0 if

$$\lim_{x \to x_0} f(x) = f(x_0)$$

The function f is continuous on the set X if it is continuous at each number in X

- The set of all functions that are continuous on the set X is denoted $\boldsymbol{C(X)}$.
- The set of all functions continuous on the closed interval $[a, b]$ is denoted $\boldsymbol{C[a, b]}$.
- R denotes the set of all real numbers. $(-\infty, +\infty)$
- All functions that are continuous at every real number is denoted by $\boldsymbol{C(R)}$

Definition 1.3 Let $\{x_n\}_{n=1}^{\infty}$ be an infinite sequence of real numbers. This sequence has the limit x (converges to x) if, for any $\varepsilon > 0$ there exists a positive integer $N(\varepsilon)$ such that $|x_n - x| < \varepsilon$, whenever $n > N(\varepsilon)$. The notation

$$\lim_{n\to\infty} x_n = x \text{ or } x_n \to x, \text{ as } n \to \infty$$

means that the sequence $\{x_n\}_{n=1}^{\infty}$ converges to x.

Theorem 1.4 If f is a function defined on a set X of real numbers and $x_0 \in X$, then the following statements are equivalent:

a. f is continuous at x_0;

b. If $\{x_n\}_{n=1}^{\infty}$ is any sequence in X converging to x_0, then

$$\lim_{n\to\infty} f(x_n) = f(x_0)$$

2. Differentiability

Definition 1.5 Let f be a function defined in an open interval containing x_0. The function f is differentiable at x_0 if

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

exists. The number $f'(x_0)$ is called the derivative of f at x_0. A function that has a derivative at each number in a set X is differentiable on X.

Figure 1.2

Theorem 1.6 If the function f is differentiable at x_0, then f is continuous at x_0.

- the set of functions that have derivatives of all orders on X is denoted $C^{\infty}(X)$
- Polynomial, rational, trigonometric, exponential, and logarithmic functions are in $C^{\infty}(X)$

Theorem 1.7 (Rolle's Theorem)

Suppose $f \in C[a,b]$ and f is differentiable on (a,b). If $f(a) = f(b)$, then a number c in (a,b) exists with $f'(c) = 0$. (See Figure 1.3)

Figure 1.3

Theorem 1.8 (Mean Value Theorem)

If $f \in C[a,b]$ and f is differentiable on (a,b), then a number c in (a,b) exists with (See Figure 1.4)

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

Figure 1.4

Theorem 1.9 (Extreme Value Theorem)

If $f \in C[a,b]$, then $c_1, c_2 \in [a,b]$ exist with $f(c_1) \leq f(x) \leq f(c_2)$, for all $x \in [a,b]$. In addition, if f is differentiable on (a,b), then the numbers c_1 and c_2 occur either at the endpoints of $[a,b]$ or where f' is zero.

Theorem 1.10 (Generalized Rolle's Theorem)

Suppose $f \in C[a,b]$ is n times differentiable on (a,b). If $f(x) = 0$ at the $n+1$ distinct numbers $a \leq x_0 < x_1 < \cdots < x_n \leq b$, then a number c in (x_0, x_n), and hence in (a,b), exists with $f^{(n)}(c) = 0$. .

Theorem 1.11 (Intermediate Value Theorem)

If $f \in C[a,b]$ and K is any number between $f(a)$ and $f(b)$, then there exists a number c in (a,b) for which $f(c) = K$.

Figure 1.7

3.Integration

Definition 1.12 The **Riemann integral** of the function f on the interval $[a, b]$ is the following limit, provided it exists:

$$\int_a^b f(x)dx = \lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(z_i)\Delta x_i$$

where the numbers $x_0, x_1, \cdots, x_n$ satisfy $a = x_0 < x_1 < \cdots < x_n = b$, ,
where $\Delta x_i = x_i - x_{i-1}$, for each $i = 1,2, \cdots, n$, and z_i is arbitrarily chosen in the interval $[x_{i-1}, x_i]$.

We can choose x_i to be equally spaced in $[a, b]$, and for each $i = 1,2,\cdots,n$, to choose $z_i = x_i$. In this case,

$$\int_a^b f(x)dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

Figure 1.8

$$x_i = a + i(b-a)/n$$

Theorem 1.13 (Weighted Mean Value Theorem for Integrals)

Suppose $f \in C[a,b]$, the Riemann integral of g exists on $[a,b]$, and $g(x)$ does not change sign on $[a,b]$. Then there exists a number c in (a,b) with

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx$$

- When $g(x) \equiv 1$, Theorem 1.13 is the usual Mean Value Theorem for Integrals.

$$f(c) = \frac{1}{b-a}\int_a^b f(x)dx$$

Figure 1.9

4.Taylor Polynomials and Series

Theorem 1.14 (Taylor's Theorem)

Suppose $f \in C^n[a,b]$, that $f^{(n+1)}$ exists on $[a,b]$, and $x_0 \in [a,b]$. For every $x \in [a,b]$, there exists a number $\xi(x)$ between x_0 and x with

$$f(x) = P_n(x) + R_n(x)$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

$$= \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k, \quad R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}$$

nth Taylor polynomial **remainder term (truncation error)**

- The infinite series obtained by taking the limit of $P_n(x)$ as $n \to \infty$ is called the **Taylor series** for f about x_0.
- In the case $x_0 = 0$, the Taylor polynomial is often called a **Maclaurin polynomial**, and the Taylor series is often called a **Maclaurin series**.
- The term **truncation error** in the Taylor polynomial refers to the error involved in using a truncated, or finite, summation to approximate the sum of an infinite series.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 1-2

Round-off Errors and Computer Arithmetic

Comparison

(1) In our traditional mathematical world, $\sqrt{3}$ is a unique positive number, when multiplied by itself produces the integer 3.

(2) In the computational world, $\sqrt{3}$ is not rational, it is given an approximate representation, one whose square will not be precisely 3, although it will likely be sufficiently close to 3 to be acceptable in most situations.

The error that is produced when a calculator or computer is used to perform real number calculations is called **round-off error**.

Binary Machine Numbers

- In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called **Binary Floating Point Arithmetic Standard** 754–1985.
- A 64-bit (binary digit) representation is used for a real number. The first bit is a sign indicator, denoted s. This is followed by an 11-bit exponent, c, called the **characteristic(首数)**, and a 52-bit binary fraction, f, called the **mantissa(尾数)**. The base for the exponent is 2.
- a floating-point number of the form.

$$(-1)^s 2^{c-1023}(1+f)$$

Illustration:

0 10000000011 1011100100010000000000000000000000000000000000000000

equivalent to the decimal number

$$c = 1 \cdot 2^{10} + 0 \cdot 2^{9} + \cdots + 0 \cdot 2^{2} + 1 \cdot 2^{1} + 1 \cdot 2^{0} = 1024 + 2 + 1 = 1027$$

$$f = 1 \cdot \left(\frac{1}{2}\right)^{1} + 1 \cdot \left(\frac{1}{2}\right)^{3} + 1 \cdot \left(\frac{1}{2}\right)^{4} + 1 \cdot \left(\frac{1}{2}\right)^{5} + 1 \cdot \left(\frac{1}{2}\right)^{8} + 1 \cdot \left(\frac{1}{2}\right)^{12}$$

$$(-1)^{s} 2^{c-1023}(1+f) = (-1)^{0} \cdot 2^{1027-1023}\left(1 + \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{4096}\right)\right)$$

$$= 27.5664065$$

The smallest normalized positive number that can be represented has $s = 0$, $c = 1$and $f = 0$ is equivalent to

$$2^{-1022} \cdot (1 + 0) \approx 0.22251 \times 10^{-307}$$

and the largest has $s = 0, c = 2046$, and $f = 1 - 2^{-52}$ and is equivalent to

$$2^{1023} \cdot (2 - 2^{-52}) \approx 0.17977 \times 10^{309}$$

Numbers occurring in calculations that have a magnitude less than $2^{-1022} \cdot (1 + 0)$ result in **underflow** and are generally set to zero. Numbers greater than $2^{1023} \cdot (2 - 2^{-52})$ result in **overflow** and typically cause the computations to stop (unless the program has been designed to detect this occurrence).

Decimal Machine Numbers

- The use of binary digits tends to conceal the computational difficulties that occur when a finite collection of machine numbers is used to represent all the real numbers.
- We will use more familiar decimal numbers instead of binary representation. we assume that machine numbers are represented in the normalized decimal floating-point form

$$\pm 0.d_1 d_2 \dots d_k \times 10^n,\ 1 \le d_1 \le 9, \text{ and } 1 \le d_i \le 9, \text{ for each } i = 2, \dots, k$$

We call it ***k*-digit decimal machine numbers**.

Decimal Machine Numbers

Any positive real number within the numerical range of the machine can be normalized to the form

$$y = 0.d_1 d_2 \dots d_k d_{k+1} d_{k+2} \dots \times 10^n$$

The floating-point form of y, denoted $fl(y)$, is obtained by terminating the mantissa of y at k decimal digits.

There are two common ways of performing this termination.

(1)chopping: That is to simply chop off the digits $d_{k+1}d_{k+2}\cdots$. This produces the floating-point form

$$fl(y) = 0.d_1d_2 \dots d_k \times 10^n$$

(2) rounding: adds $5 \times 10^{n-(k+1)}$ to y and then chops the result to obtain a number of the form

$$fl(y) = 0.\delta_1\delta_2 \dots \delta_k \times 10^n$$

For rounding, $d_{k+1} \geq 5$, we add 1 to d_k to obtain $fl(y)$; that is **round up**. when $d_{k+1} < 5$, we chop off all but the first k digits; that is **round down**.

Example 1 Determine the five-digit (a) chopping and (b) rounding values of the irrational number π.

Solution: Written in normalized decimal form, we have $\pi = 0.314159265 \ldots \times 10^1$

(a) The floating-point form of π using five-digit chopping is

$$fl(\pi) = 0.31415 \times 10^1 = 3.1415$$

(b) The sixth digit of the decimal expansion of π is a 9, so the floating-point form of π using five-digit rounding is

$$fl(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416$$

Definition 1.15 Suppose that p^* is an approximation to p. The **absolute error** is $|p - p^*|$, and the **relative error** is $\frac{|p-p^*|}{|p|}$, provided that $p \neq 0$.

Example 2 Determine the absolute and relative errors when approximating p by p^*

when **(a)** $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$

(b) $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$

(c) $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$

Solution: **(a)** the absolute error is 0.1, and the relative error is $0.333\overline{3} \times 10^{-1}$

(b) the absolute error is 0.1×10^{-4}, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(c) the absolute error is 0.1×10^3, and the relative error is $0.333\overline{3} \times 10^{-1}$.

- This example shows that the same relative error, occurs for widely varying absolute errors.
- As a measure of accuracy, the absolute error can be misleading and **the relative error more meaningful**, because the relative error takes into consideration the size of the value.

In scientific measurements, it is almost always the relative error that is significant.

Significant digits (or figures) are the leading nonzero digit and all following digits.

Finite-Digit Arithmetic

Assume that the floating-point representations $fl(x)$ and $fl(y)$ are given for the real numbers x and y and that the symbols $\oplus, \ominus, \otimes, \oslash$ represent machine addition, subtraction,multiplication, and division operations, respectively. We will assume a finite-digit arithmetic given by

$$x \oplus y = fl(fl(x) + fl(y)) \qquad x \otimes y = fl(fl(x) \times fl(y))$$

$$x \ominus y = fl(fl(x) - fl(y)) \qquad x \oslash y = fl(fl(x) \div fl(y))$$

This arithmetic corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its finite-digit floating-point representation.

Loss of significance

One major problem that arises in many forms is the loss of significant digits that results from subtracting nearly equal numbers. Assume that through considerable effort, as part of a long calculation, we have determined two numbers correct to seven significant digits, and now need to subtract them:

$$\begin{array}{r} 123.4567 \\ -\ 123.4566 \\ \hline 000.0001 \end{array}$$

The two input numbers are seven-digit accuracy, and ended with a result that has only one-digit accuracy.

Example

Calculate $\sqrt{9.01} - 3$ on a three-decimal-digit computer.

Suppose two nearly equal numbers x and y, with $x > y$, have the k-digit representations

$$fl(x) = 0.d_1 d_2 \dots d_p \alpha_{p+1} \alpha_{p+2} \dots \alpha_k \times 10^n,$$

and

$$fl(y) = 0.d_1 d_2 \dots d_p \beta_{p+1} \beta_{p+2} \dots \beta_k \times 10^n.$$

The floating-point form of $x - y$ is

$$fl\big(fl(x) - fl(y)\big) = 0.\sigma_{p+1} \sigma_{p+2} \dots \sigma_k \times 10^{n-p},$$

where

$$0.\sigma_{p+1} \sigma_{p+2} \dots \sigma_k = 0.\alpha_{p+1} \alpha_{p+2} \dots \alpha_k - 0.\beta_{p+1} \beta_{p+2} \dots \beta_k.$$

If a finite-digit representation or calculation introduces an error, further enlargement of the error occurs when dividing by a number with small magnitude (or, equivalently, when multiplying by a number with large magnitude).

Suppose that the number z has the finite-digit approximation $z + \delta$, error δ is introduced by representation or by previous calculation. Now divide by $\varepsilon = 10^{-n}$, where $n > 0$. Then

$$\frac{z}{\varepsilon} \approx fl\left(\frac{fl(z)}{fl(\varepsilon)}\right) = (z + \delta) \times 10^n$$

The absolute error in this approximation, $|\delta| \times 10^n$, is the original absolute error, $|\delta|$, multiplied by the factor 10^n.

Illustration

Consider this formula applied to the equation $x^2 + 62.10x + 1 = 0$, whose roots are approximately

$$x_1 = -0.01610723 \text{ and } x_2 = -62.08390.$$

We will again use four-digit rounding arithmetic in the calculations to determine the root. In this equation, b^2 is much larger than $4ac$, so the numerator in the calculation for x_1 involves the ***subtraction*** of nearly equal numbers.

To obtain a more accurate four-digit rounding approximation for x_1, we change the form of the quadratic formula by ***rationalizing the numerator***:

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}\left(\frac{-b-\sqrt{b^2-4ac}}{-b-\sqrt{b^2-4ac}}\right) = \frac{b^2-(b^2-4ac)}{2a(-b-\sqrt{b^2-4ac})}$$

which simplifies to an alternate quadratic formula

$$x_1 = \frac{-2c}{b+\sqrt{b^2-4ac}} \qquad (1.2)$$

Using (1.2) gives

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610$$

which has the small relative error 6.2×10^{-4}.

The rationalization technique can also be applied to give the following alternative quadratic formula for x_2:

$$x_2 = \frac{-2c}{b - \sqrt{b^2 - 4ac}}$$

This is the form to use if b is a negative number.

In the Illustration, however, the mistaken use of this formula for x_2 would result in not only the subtraction of nearly equal numbers, but also the division by the small result of this subtraction.

The inaccuracy that this combination produces,

$$fl(x_2) = \frac{-2c}{b - \sqrt{b^2 - 4ac}} = \frac{-2.000}{62.10 - 62.06} = \frac{-2.000}{0.04000} = -50.00$$

has the large relative error 1.9×10^{-1}.

Nested Arithmetics

Accuracy loss due to round-off error can also be reduced by rearranging calculations

Example 6 Evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ at $x = 4.71$ using three-digit arithmetic.

Solution: Table 1.4 gives the intermediate results in the calculations.

Table 1.4

	x	x^2	x^3	$6.1x^2$	$3.2x$
Exact	4.71	22.1841	104.487111	135.32301	15.072
Three-digit (chopping)	4.71	22.1	104.	134.	15.0
Three-digit (rounding)	4.71	22.2	105.	135.	15.1

To illustrate the calculations, let us look at those involved with finding x^3 using three-digit rounding arithmetic. First we find

$$x^2 = 4.71^2 = 22.1841 \text{ which rounds to } 22.2.$$

$$x^3 = x^2 \cdot x = 22.2 \cdot 4.71 = 104.562 \text{ which rounds to } 105.$$

$$6.1x^2 = 6.1 \cdot 22.2 = 135.42 \text{ which rounds to } 135,$$

$$3.2x = 3.2 \cdot 4.71 = 15.072 \text{ which rounds to } 15.1.$$

Exact: $f(4.71) = 104.487111 - 135.32301 + 15.072 + 1.5 = -14.263899.$

Using finite-digit arithmetic, the way in which we add the results can effect the final result. Suppose that we add left to right. Then for chopping arithmetic we have

Three-digit(chopping): $f(4.71) = \big((104. - 134.) + 15.0\big) + 1.5 = -13.5,$

and for rounding arithmetic we have

Three-digit(rounding): $f(4.71) = \big((105. - 135.) + 15.1\big) + 1.5 = -13.4.$

The relative errors for the three-digit methods are

Chopping: $\left|\frac{-14.263899+13.5}{-14.263899}\right| \approx 0.05$, and **Rounding**: $\left|\frac{-14.263899+13.4}{-14.263899}\right| \approx 0.06$.

Illustration As an alternative approach, the polynomial $f(x)$ in **Example 6** can be written in a **nested** manner as

$$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5 = \big((x - 6.1)x + 3.2\big)x + 1.5.$$

Using three-digit chopping arithmetic now produces

$$
\begin{aligned}
f(4.71) &= \big((4.71 - 6.1) \times 4.71 + 3.2\big) \times 4.71 + 1.5 \\
&= \big((-1.39) \times (4.71) + 3.2\big) \times 4.71 + 1.5 \\
&= (-6.54 + 3.2) \times 4.71 + 1.5 \\
&= (-3.34) \times 4.71 + 1.5 \\
&= -15.7 + 1.5 \\
&= -14.2
\end{aligned}
$$

$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$

The new relative errors ars

$$\textbf{Three-digit(chopping)}: \left|\frac{-14.263899+14.2}{-14.263899}\right| \approx 0.0045$$

$$\textbf{Three-digit(rounding)}: \left|\frac{-14.263899+14.3}{-14.263899}\right| \approx 0.0025$$

- The decreased error in the Illustration is due to the reduction in computations from four multiplications and three additions to two multiplications and three additions.
- Polynomials should always be expressed in nested form before performing an evaluation. One way to reduce round-off error is to reduce the number of computations.

ill - posed problem :

$$\begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{6} \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 = \frac{13}{12} \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = \frac{47}{60} \end{cases}$$

well - posed problem :

$$\begin{cases} 2x_1 - x_2 = 6 \\ x_1 + 2x_2 = -2 \end{cases}$$

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 1-3

Algorithms and Convergence

Introduction

- An **algorithm** is a procedure that describes, in an unambiguous manner, a finite sequence of steps to be performed in a specified order.
- The **object** of the algorithm is to implement a procedure to solve a problem or approximate a solution to the problem.

Illustration The following algorithm computes $x_1 + x_2 + \cdots + x_N = \sum_{i=1}^{N} x_i$, given N and the numbers $x_1, x_2, \ldots, x_N$.

INPUT $N, x_1, x_2, \ldots, x_N$.

OUTPUT $SUM = \sum_{i=1}^{N} x_i$.

Step 1 Set $SUM = 0$. (*Initialize accumulator.*)

Step 2 For $i = 1, 2, \ldots, N$ do set $SUM = SUM + x_i$. (*Add the next term.*)

Step 3 OUTPUT (SUM);

STOP.

Example 1 The Nth Taylor polynomial for $f(x) = \ln x$ expanded about $x_0 = 1$ is

$$P_N(x) = \sum_{i=1}^{N} \frac{(-1)^{i+1}}{i}(x-1)^i,$$

and the value of $\ln 1.5$ to eight decimal places is 0.40546511. Construct an algorithm to determine the minimal value of N required for

$$|\ln 1.5 - P_N(1.5)| < 10^{-5},$$

without using the Taylor polynomial remainder term.

Solution:

INPUT value x, tolerance TOL, maximum number of iterations M.
OUTPUT degree N of the polynomial or a message of failure.
Step 1 Set $N = 1$;
$y = x - 1$;
$SUM = 0$;
$POWER = y$;
$TERM = y$;
$SIGN = -1$. (*Used to implement alternation of signs.*)

Step 2 While $N \leq M$ do Steps 3–5.

Step 3 Set $SIGN = -SIGN$; (*Alternate the signs.*)
$SUM = SUM + SIGN \cdot TERM$; (*Accumulate the terms.*)
$POWER = POWER \cdot y$;
$TERM = POWER/(N + 1)$. (*Calculate the next term.*)

Step 4 If $|TERM| < TOL$ then (*Test for accuracy.*)
$OUTPUT\ (N)$;
STOP. (*The procedure was successful.*)

Step 5 Set $N = N + 1$. (*Prepare for the next iteration.*)

Step 6 OUTPUT ('Method Failed'); (*The procedure was unsuccessful.*)
STOP.

Characterizing Algorithms

Criterion: small changes in the initial data produce correspondingly small changes in the final results

An algorithm that satisfies this property is called **stable**; otherwise it is unstable.

Some algorithms are stable only for certain choices of initial data, and are called **conditionally stable**.

Example Compute $I_n = e^{-1}\int_0^1 x^n e^x\, dx (n = 0,1,\cdots)$ and evaluate its error.

Example Compute $I_n = e^{-1}\int_0^1 x^n e^x\,dx (n = 0,1,\cdots)$ and evaluate its error.

n	$\tilde{I}_n$	I_n^*	n	$\tilde{I}_n$	I_n^*
0	0.6321	0.6321	5	0.1480	0.1455
1	0.3679	0.3679	6	0.1120	0.1268
2	0.2642	0.2643	7	0.2160	0.1121
3	0.2074	0.2073	8	-0.7280	0.1035
4	0.1704	0.1708	9	7.552	0.0684

Definition 1.17 Suppose that $E_0 > 0$ denotes an error introduced at some stage in the calculations and E_n represents the magnitude of the error after n subsequent operations.

- If $E_n \approx CnE_0$, where C is a constant independent of n, then the growth of error is said to be **linear**.
- If $E_n \approx C^n E^0$, for some $\boldsymbol{C > 1}$, then the growth of error is called **exponential**.

Linear growth of error is usually unavoidable, and when C and E_0 are small the results are generally acceptable. Exponential growth of error should be avoided, because the term C^n becomes large for even relatively small values of n. This leads to unacceptable inaccuracies, regardless of the size of E_0.

an algorithm that exhibits linear growth of error is stable, whereas an algorithm exhibiting exponential error growth is unstable.

Illustration For any constants c_1 and c_2,

$$p_n = c_1(\frac{1}{3})^n + c_2 3^n,$$

is a solution to the recursive equation

$$p_n = \frac{10}{3} p_{n-1} - p_{n-2}, \text{ for } n = 2,3, \ldots$$

Suppose that we are given $p_0 = 1$ and $p_1 = \frac{1}{3}$. As $c_1 = 1$ and $c_2 = 0$. So $p_n = (\frac{1}{3})^n$ for all n.

If five-digit rounding arithmetic is used, then $\hat{p}_0 = 1.0000$ and $\hat{p}_1 = 0.3333$, which requires modifying the constants to $\hat{c}_1 = 1.0000$ and $\hat{c}_2 = -0.12500 \times 10^{-5}$.

$$\hat{p}_n = 1.0000(\tfrac{1}{3})^n - 0.12500 \times 10^{-5}(3)^n,$$

which has round-off error,

$$p_n - \hat{p}_n = 0.12500 \times 10^{-5}(3)^n,$$

this procedure is unstable because the error grows exponentially with n.

Table 1.5

n	Computed $\hat{p}_n$	Correct p_n	Relative Error
0	0.10000×10^{1}	0.10000×10^{1}	
1	0.33333×10^{0}	0.33333×10^{0}	
2	0.11110×10^{0}	0.11111×10^{0}	9×10^{-5}
3	0.37000×10^{-1}	0.37037×10^{-1}	1×10^{-3}
4	0.12230×10^{-1}	0.12346×10^{-1}	9×10^{-3}
5	0.37660×10^{-2}	0.41152×10^{-2}	8×10^{-2}
6	0.32300×10^{-3}	0.13717×10^{-2}	8×10^{-1}
7	-0.26893×10^{-2}	0.45725×10^{-3}	7×10^{0}
8	-0.92872×10^{-2}	0.15242×10^{-3}	6×10^{1}

Now consider this recursive equation:

$$p_n = 2p_{n-1} - p_{n-2}, \text{ for } n = 2,3, \ldots$$

It has the solution $p_n = c_1 + c_2 n$ for any constants c_1 and c_2.

If we are given $p_0 = 1$ and $p_1 = \frac{1}{3}$, then constants in this equation are uniquely determined to be $c_1 = 1$ and $c_2 = -\frac{2}{3}$. This implies that $p_n = 1 - \frac{2}{3}n$.

If five-digit rounding arithmetic is used, then $\hat{p}_0 = 1.0000$ and $\hat{p}_1 = 0.33333$. The five-digit rounding constants are $\hat{c}_1 = 1.0000$ and $\hat{c}_2 = -0.66667$. Thus

$$\hat{p}_n = 1.0000 - 0.66667n,$$

which has round-off error

$$p_n - \hat{p}_n = \left(0.66667 - \frac{2}{3}\right)n.$$

This procedure is stable because the error grows ***linearly*** with n.

Table 1.6

n	Computed $\hat{p}_n$	Correct p_n	Relative Error
0	0.10000×10^1	0.10000×10^1	
1	0.33333×10^0	0.33333×10^0	
2	-0.33330×10^0	-0.33333×10^0	9×10^{-5}
3	-0.10000×10^1	-0.10000×10^1	0
4	-0.16667×10^1	-0.16667×10^1	0
5	-0.23334×10^1	-0.23333×10^1	4×10^{-5}
6	-0.30000×10^1	-0.30000×10^1	0
7	-0.36667×10^1	-0.36667×10^1	0
8	-0.43334×10^1	-0.43333×10^1	2×10^{-5}

- The effects of round-off error can be reduced by using high-order-digit arithmetic such as the double- or multiple-precision option available on most computers.
- Disadvantages in using double-precision arithmetic are that it takes more computation time and the growth of round-off error is not entirely eliminated.
- One approach to estimating round-off error is to use interval arithmetic (that is, to retain the largest and smallest possible values at each step), in the end, we obtain an interval that contains the true value.

Rates of Convergence

Definition 1.18 Suppose $\{\beta_n\}_{n=1}^{\infty}$ is a sequence known to converge to zero, and $\{\alpha_n\}_{n=1}^{\infty}$ converges to a number α. If a positive constant K exists with

$$|\alpha_n - \alpha| \leq K|\beta_n|, \text{ for large } n,$$

then we say that $\{\alpha_n\}_{n=1}^{\infty}$ converges to α with **rate, or order, of convergence** $O(\beta_n)$. It is indicated by writing $\alpha_n = \alpha + O(\beta_n)$.

We use $\boldsymbol{\beta_n = \frac{1}{n^p}}$, for some number $p > 0$. We are generally interested in the largest value of p with $\alpha_n = \alpha + O(\frac{1}{n^p})$.

Example 2 Suppose that, for $n \geq 1$,

$$\alpha_n = \frac{n+1}{n^2} \text{ and } \hat{\alpha}_n = \frac{n+3}{n^3}$$

Both $\lim_{n\to\infty} \alpha_n = 0$ and $\lim_{n\to\infty} \hat{\alpha}_n = 0$, but the sequence $\{\hat{\alpha}_n\}$ converges to this limit much faster than the sequence $\{\alpha_n\}$. Using five-digit rounding arithmetic we have the values shown in Table 1.7. Determine rates of convergence for these two sequences.

Table 1.7

n	1	2	3	4	5	6	7
α_n	2.00000	0.75000	0.44444	0.31250	0.24000	0.19444	0.16327
$\hat{\alpha}_n$	4.00000	0.62500	0.22222	0.10938	0.064000	0.041667	0.029155

Solution: Define the sequences $\beta_n = \frac{1}{n}$ and $\hat{\beta}_n = \frac{1}{n^2}$. Then

$$|\alpha_n - 0| = \frac{n+1}{n^2} \leq \frac{n+n}{n^2} = \frac{2}{n} = 2\beta_n$$

and

$$|\hat{\alpha}_n - 0| = \frac{n+3}{n^3} \leq \frac{n+3n}{n^3} = \frac{4}{n^2} = 4\hat{\beta}_n$$

Hence the rate of convergence of $\{\alpha_n\}$ to zero is similar to the convergence of $\left\{\frac{1}{n}\right\}$ to zero, whereas $\{\hat{\alpha}_n\}$ converges to zero at a rate similar to the more rapidly convergent sequence $\left\{\frac{1}{n^2}\right\}$.

$$\alpha_n = 0 + O\left(\frac{1}{n}\right) \text{ and } \hat{\alpha}_n = 0 + O\left(\frac{1}{n^2}\right).$$

Definition 1.19 Suppose that $\lim_{h\to 0} G(h) = 0$ and $\lim_{h\to 0} F(h) = L$. If a positive constant K exists with

$$|F(h) - L| \le K|G(h)|, \text{ for sufficiently small } h,$$

then we write $F(h) = L + O(G(h))$.

- The functions for comparison generally have the form $G(h) = h^p$, where $p > 0$. We are interested in the largest value of p for which $F(h) = L + O(h^p)$.

Example 3 Use the third Taylor polynomial about $h = 0$ to show that

$$\cos h + \frac{1}{2}h^2 = 1 + O(h^4).$$

Solution:

$$\cos h = 1 - \frac{1}{2}h^2 + \frac{1}{24}h^4 \cos \hat{\xi}(h),$$

for some number $\hat{\xi}(h)$ between zero and h. This implies that

$$\cos h + \frac{1}{2}h^2 = 1 + \frac{1}{24}h^4 \cos \hat{\xi}(h).$$

Hence

$$\left|\left(\cos h + \frac{1}{2}h^2\right) - 1\right| = \left|\frac{1}{24}\cos\hat{\xi}(h)\right| h^4 \leq \frac{1}{24}h^4$$

so as $h \to 0$, $\cos h + \frac{1}{2}h^2$ converges to its limit, 1, about as fast as h^4 converges to 0.

That is,

$$\cos h + \frac{1}{2}h^2 = 1 + O(h^4)$$

Maple uses the O notation to indicate the form of the error in Taylor polynomials and in other situations.

Some principles of avoiding loss of significant figures

- Avoid the case of $\frac{x}{y}$, if $0 < |y| \ll |x|$.
- Avoid the subtraction of nearly equal numbers
- Avoid the big number eats the small one
- Reduce the number of arithmetic calculations

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Chapter 2

Solutions of Equations in One Variable

introduction

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 2-1

The Bisection Method

二分法

Example

Given the equation $f(x) = x^3 - x - 1 = 0$, find the interval of the root.

In this chapter

- the root-finding problem--one of the most basic problems of numerical approximation.

Bisection Technique

1.Bisection

The first technique, based on the Intermediate Value Theorem, is called the **Bisection**, or **Binary-search**, method.

f--a continuous function defined on the interval $[a, b]$, $f(a)$and$f(b)$of opposite sign.

The Intermediate Value Theorem implies that a number p exists in(a, b) with$f(p) = 0$.

assume for simplicity that the root in this interval is unique.
The method calls for a repeated halving (or bisecting) of subintervals of $[a, b]$ and, at each step, locating the half containing p.

To begin, set $a_1 = a$ and $b_1 = b$, and let p_1 be the midpoint of $[a, b]$; that is,

$$p_1 = a_1 + \frac{b_1 - a_1}{2} = \frac{a_1 + b_1}{2}$$

- If $f(p_1) = 0$, then $p = p_1$, and we are done.
- If $f(p_1) \neq 0$, then $f(p_1)$ has as the same sign as either $f(a_1)$ or $f(b_1)$.

  If $f(p_1)$ and $f(a_1)$ have the same sign, $p \in (p_1, b_1)$. Set $a_2 = p_1$ and $b_2 = b_1$.
  If $f(p_1)$ and $f(a_1)$ have opposite signs, $p \in (a_1, p_1)$. Set $a_2 = a_1$ and $b_2 = p_1$.

Then reapply the process to the interval $[a_2, b_2]$. This produces the method described in Algorithm 2.1.

Figure 2.1

Bisection

To find a solution to $f(x) = 0$ given the continuous function f on the interval [a, b], where $f(a)$ and $f(b)$ have opposite signs

INPUT *endpoints a, b; tolerance TOL; maximum number of iterations* N_0.

OUTPUT *approximate solution p or message of failure.*

Step 1 *Set i = 1;*

FA = f (a).

Step 2 *While* $i \leq N_0$ *do Steps 3–6.*

Step 3 *Set* $p = a + (b - a)/2$; *(Compute pi.)*

$FP = f(p)$.

Step 4 *If* $FP = 0$ *or* $(b - a)/2 < TOL$ *then*

OUTPUT (p); (Procedure completed successfully.)

STOP.

Step 5 *Set* $i = i + 1$.

Step 6 *If* $FA \cdot FP > 0$ *then set* $a = p$; *(Compute* a_i, b_i.*)*

$FA = FP$

else set $b = p$. *(FA is unchanged.)*

***Step** 7 OUTPUT ('Method failed after N_0 iterations, N_0 =', N_0);*

(The procedure was unsuccessful.)

STOP.

Other stopping procedures can be applied in Step 4 of Algorithm 2.1 or in any of the iterative techniques in this chapter.

Example: we can select a tolerance ε > 0 and generate $p_1, \ldots, p_N$ until one of the following conditions is met:

$$|p_N - p_{N-1}| < \varepsilon \qquad (2.1)$$

$$\frac{|p_N - p_{N-1}|}{p_N} < \varepsilon, \qquad p_N \neq 0, \qquad \text{or} \qquad (2.2)$$

$$|f(p_N)| < \varepsilon \qquad (2.3)$$

Unfortunately, difficulties can arise using any of these stopping criteria.

- For example, there are sequences $\{p_n\}_{n=0}^{\infty}$ with the property that the differences $p_n - p_{n-1}$ converge to zero while the sequence itself diverges.

  for example: $p_n = \sum_{k=1}^{n} \frac{1}{k}$.

- It is also possible for $f(p_n)$ to be close to zero while p_n differs significantly from p.

  for example: $f(x) = (x-1)^{10}, p = 1$ and $p_n = 1 + \frac{1}{n}$. Show that $|f(p_n)| < 10^{-3}$ whenever $n > 1$ but that $|p - p_n| < 10^{-3}$ requires that $n > 1000$.

- Without additional knowledge about f or p, Inequality (2.2) is the best stopping criterion to apply because it comes closest to testing relative error.

$$\frac{|p_N - p_{N-1}|}{p_N} < \varepsilon, \qquad p_N \neq 0, \qquad (2.2)$$

Note: When using a computer to generate approximations, it is good practice to set an upper bound on the number of iterations.
This eliminates the possibility of entering an infinite loop, a situation that can arise when the sequence diverges (and also when the program is incorrectly coded).

Note that to start the Bisection Algorithm, an interval $[a, b]$ must be found with $f(a) \cdot f(b) < 0$. At each step the length of the interval known to contain a zero of f is reduced by a factor of 2; hence it is advantageous to choose the initial interval $[a, b]$ as small as possible.

Example: $f(x) = 2x^3 - x^2 + x - 1$, we have both

$$f(-4) \cdot \mathrm{f}(4) < 0 \quad \text{and} \quad \mathrm{f}(0) \cdot \mathrm{f}(1) < 0$$

so the Bisection Algorithm could be used on $[-4, 4]$ or on $[0, 1]$.

- Starting the Bisection Algorithm on $[0, 1]$ instead of $[-4, 4]$ will reduce by 3 the number of iterations required to achieve a specified accuracy.

Example 1

Show that $f(x) = x^3 + 4x^2 - 10 = 0$ has a root in [1,2], and use the Bisection method to determine an approximation to the root that is accurate to at least within 10^{-4}.

Solution 1.$f(1) = -5, f(2) = 14$,so this continuous function has a root in [1,2].For the first iteration of the Bisection method, $f(1.5) = 2.375 > 0$;

2.We should select the interval [1,1.5] for our second iteration, $f(1.25) = -1.796875$;

3.new interval becomes [1.25,1.5],whose midpoint is 1.375.

.....(Table 2.1)

After 13 iterations, p13 = 1.365112305 approximates the root p with an error

$$|p - p_{13}| < |b_{14} - a_{14}| = |1.365234375 - 1.365112305| = 0.000122070$$

Since $|a_{14}| < |p|$, we have

$$\frac{|p - p_{13}|}{|p|} < \frac{|b_{14} - a_{14}|}{|a_{14}|} \leq 9.0 \times 10^{-5}$$

Table 2.1

n	a_n	b_n	p_n	$f(p_n)$
1	1.0	2.0	1.5	2.375
2	1.0	1.5	1.25	−1.79687
3	1.25	1.5	1.375	0.16211
4	1.25	1.375	1.3125	−0.84839
5	1.3125	1.375	1.34375	−0.35098
6	1.34375	1.375	1.359375	−0.09641
7	1.359375	1.375	1.3671875	0.03236
8	1.359375	1.3671875	1.36328125	−0.03215
9	1.36328125	1.3671875	$\boxed{1.365234375}$	0.000072
10	1.36328125	1.365234375	1.364257813	−0.01605
11	1.364257813	1.365234375	1.364746094	−0.00799
12	1.364746094	1.365234375	1.364990235	−0.00396
13	1.364990235	1.365234375	$\boxed{1.365112305}$	−0.00194

so the approximation is correct to at least within 10^{-4}

Disadvantages of The Bisection method

The Bisection method has significant drawbacks.

- It is relatively slow to converge.
- a good intermediate approximation might be inadvertently discarded.

Advantages of dichotomy

- conceptually clear
- it always converges to a solution, it is often used to find the original value of the other methods.

Theorem 2.1

Suppose that $f \in C[a,b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \leq \frac{b-a}{2^n}, n \geq 1.$$

Note: Theorem 2.1 gives only a bound for approximation error and that this bound might be quite conservative.

In example 1:

$$|p - p_9| \leq \frac{2-1}{2^9} \approx 2 \times 10^{-3}$$

But the actual error is

$$|p - p_9| = |1.365230013 - 1.365234375| \approx 4.4 \times 10^{-6}$$

Example 2

Determine the number of iterations necessary to solve $f(x) = x^3 + 4x^2 - 10 = 0$ with accuracy 10^{-3} using $a_1 = 1$ and $b_1 = 2$.

When implementing the method on a computer, we need to consider the effects of round-off error.

For example the computation of the midpoint of the interval [an, bn] should be found from the equation

$$p_n = a_n + \frac{b_n - a_n}{2} \text{ instead of } p_n = \frac{a_n + b_n}{2}$$

- The first equation adds a small correction, $(b_n - a_n)/2$, to the known value a_n.
- When $b_n - a_n$ is near the maximum precision of the machine, this correction might be in error.
- when $b_n - a_n$ is near the maximum precision of the machine, it is possible for $(a_n + b_n)/2$ to return a midpoint that is not even in the interval $[a_n, b_n]$

As a final remark, to determine which subinterval of $[a_n, b_n]$ contains a root of f The test

$$\mathrm{sgn}(f(a_n))\mathrm{sgn}(f(p_n)) < 0 \quad \text{instead of} \quad f(a_n)f(p_n) < 0$$

gives the same result but avoids the possibility of overflow or underflow in the multipliation of $f(a_n)$ and $f(p_n)$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 2-2

Fixed-Point Iteration

- Finding solutions to fixed-point problems
- Finding the connection between the fixed-point problems and the root-finding problems

Iteration: $x_{k+1} = \varphi(x_k)$

Equivalence Problem: $f(x) = 0 \Leftrightarrow x = \varphi(x)$

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

$$x = \varphi_1(x) = \sqrt[3]{x+1}$$

$$x = \varphi_2(x) = \frac{1}{x^2 - 1}$$

$$x = \varphi_3(x) = x^3 - 1$$

Which one is better?

Definition 2.2 The number p is a **fixed point** for a given function g if $g(p) = p$.

Root-finding problems and fixed-point problems are equivalent classes in the following sense:

- Given a root-finding problem $f(p) = 0$, we can define functions g with a fixed point at p in a number of ways, for example, as

$$g(x) = x - f(x) \text{ or as } g(x) = x + 3f(x)$$

- Conversely, if the function g has a fixed point at p, then the function defined by

$$f(x) = x - g(x)$$

has a zero at p.

Example 1: Determine any fixed points of the function $g(x) = x^2 - 2$.

Figure 2.3

Existence and uniqueness of a fixed point

Theorem 2.3 (i) If $g \in C[a,b]$ and for all $g(x) \in [a,b]$, then g has at least one fixed point in $[a,b]$.

(ii) If, in addition, $g'(x)$exists on (a,b) and a positive constant $k < 1$ exists with

$$|g'(x)| \leq k < 1, \qquad \text{for all } x \in (a,b)$$

then there is exactly one fixed point in $[a,b]$.

Note: The theorem gives **sufficient conditions** for the existence and uniqueness of a fixed point.

Figure 2.4

Proof **(i)** If $g(a) = a$ or $g(b) = b$.then g has a fixed point at an endpoint.If not, then $g(a) > a$ and $g(b) < b$.

The function $h(x) = g(x) - x$ is continuous on $[a, b]$, with $h(a) = g(a) - a > 0$ and $h(b) = g(b) - b < 0$.

The Intermediate Value Theorem implies that there exists $p \in (a, b)$ for which $h(p) = 0$. This number p is a fixed point for g because

$$0 = h(p) = g(p) - p \quad \text{implies that } g(p) = p$$

Proof **(ii)** Suppose that $|g'(x)| \leq k < 1$ and that p and q are both fixed points in $[a, b]$.If $p \neq q$,then the Mean Value Theorem implies that a number ξ exists between p and q, and hence in $[a, b]$, with

$$\frac{g(p) - g(q)}{p - q} = g'(\xi)$$

Thus

$$|p - q| = |g(p) - g(q)| = |g'(\xi)||p - q| \leq k|p - q| < |p - q|$$

which is a contradiction. This contradiction must come from the only supposition, $p \neq q$.Hence,$p = q$ and the fixed point in $[a, b]$ is unique.

Example 2: Show that $g(x) = (x^2 - 1)/3$ has a unique fixed point on the interval $[-1,1]$.

Solution Since $g'(x) = 2x/3$, the function g is continuous and $g'(x)$ exists on $[-1, 1]$.

So The maximum and minimum values of $g(x)$ occur at $x = -1$, $x = 0$, or $x = 1$. But $g(-1) = 0$, $g(1) = 0$, and $g(0) = -1/3$,
so an absolute maximum for $g(x)$ on $[-1, 1]$ occurs at $x = -1$ and $x = 1$, and an absolute minimum at $x = 0$.

Moreover

$$|g'(x)| = \left|\frac{2x}{3}\right| \leq \frac{2}{3}, \qquad for\ all\ x \in (-1,1)$$

So g satisfies all the hypotheses of Theorem 2.3 and has a unique fixed point in $[-1, 1]$.

Supplement In Example 2, we can solve the unique fixed point p. If

$$p = g(p) = \frac{p^2 - 1}{3}$$

which implies that

$$p = \frac{1}{2}(3 - \sqrt{13})$$

Supplement

g also has a unique fixed point $p = \frac{1}{2}(3 + \sqrt{13})$ for the interval [3, 4].

However, $g(4) = 5$ and $g'(4) = \frac{8}{3} > 1$.This demonstrates that the hypotheses of Theorem 2.3 are **sufficient** to guarantee a unique fixed point but are **not necessary**.

Example 3 Show that Theorem 2.3 does not ensure a unique fixed point of $g(x) = 3^{-x}$ on the interval[0,1], even though a unique fixed point on this interval does exist.

Solution

$g'(x) = -3^{-x}ln3 < 0$ on[0,1],so

$$g(1) = \frac{1}{3} \leq g(x) \leq 1 = g(0), \qquad for\ 0 \leq x \leq 1.$$

Thus, for $x \in [0,1]$, we have $g(x) \in [0,1]$.The first part of Theorem 2.3 ensures that there is at least one fixed point in [0,1].However,

$$g'(x) = -ln3 = -1.098612289$$

so $|g'(x)| \nleq 1$on (0,1),and Theorem 2.3 cannot be used to determine uniqueness

Figure 2.6

Fixed-Point Iteration

To approximate the fixed point of a function g

- we choose an initial approximation p_0 and generate the sequence $\{p_n\}_{n=0}^{\infty}$ by letting $\boldsymbol{p_n = g(p_{n-1})}$, for each n ≥ 1.
- If the sequence **converges** to p and g is **continuous**, then

$$p = \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g(\lim_{n\to\infty} p_{n-1}) = g(p)$$

a solution to $x = g(x)$ is obtained.

This technique is called **fixed-point**, or **functional iteration**.

Fixed-point iteration process

Figure 2.7

Fixed-point iterative algorithm

To find a solution to $p = g(p)$ given an initial approximation p_0.

INPUT initial approximation p_0; tolerance TOL; maximum number of iterations N_0.

OUTPUT approximate solution p or message of failure

Step 1 Set $i = 1$.

Step 2 While $i \leq N_0$ do Steps 3–6.

Step 3 Set $p = g(p_0)$. (Compute p_i.)

Step 4 If $|p - p_0| < TOL$ then

OUTPUT (p); (The procedure was successful.)

STOP

Step 5 Set i = i + 1.

Step 6 Set $p_0 = p$.(Update p_0.)

Step 7 OUTPUT ('The method failed after N_0 iterations, $N_0 =', N_0$);

(The procedure was unsuccessful.).)

STOP.

Illustration

The equation $x^3 + 4x^2 - 10 = 0$ has a unique root in [1,2].

$(a)\ x = g_1(x) = x - x^3 - 4x^2 + 10,$

$(b)\ x = g_2(x) = \left(\frac{10}{x} - 4x\right)^{\frac{1}{2}}$

$(c)\ x = g_3(x) = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$

$(d)\ x = g_4(x) = \left(\frac{10}{4+x}\right)^{\frac{1}{2}}$

$(e)\ x = g_5(x) = x - \frac{x^3+4x^2-10}{3x^2+8x}$

With $p_0 = 1.5$

Table 2.2

n	(a)	(b)	(c)	(d)	(e)
0	1.5	1.5	1.5	1.5	1.5
1	−0.875	0.8165	1.286953768	1.348399725	1.373333333
2	6.732	2.9969	1.402540804	1.367376372	1.365262015
3	−469.7	$(-8.65)^{1/2}$	1.345458374	1.364957015	1.365230014
4	1.03×10^8		1.375170253	1.365264748	1.365230013
5			1.360094193	1.365225594	
6			1.367846968	1.365230576	
7			1.363887004	1.365229942	
8			1.365916734	1.365230022	
9			1.364878217	1.365230012	
10			1.365410062	1.365230014	
15			1.365223680	1.365230013	
20			1.365230236		
25			1.365230006		
30			1.365230013		

Question: How can we find a fixed-point problem that produces a sequence that reliably and rapidly converges to a solution to a given root-finding problem?

Theorem 2.4 (Fixed-Point Theorem) Let $g \in C[a,b]$ be such that $g(x) \in [a,b]$, for all x in $[a,b]$. Suppose, in addition, that g' exists on (a,b) and that a constant $0 < k < 1$ exists with

$$|g'(x)| \leq k < 1, \qquad \text{for all } x \in (a,b)$$

Then for any number p_0 in $[a,b]$, the sequence defined by

$$p_n = g(p_{n-1}), \qquad n \geq 1$$

converges to the unique fixed point p in $[a,b]$.

Proof We know a unique point p exists in $[a, b]$ with $g(p) = p$.Using the fact that $|g'(x)| \leq k$ and the Mean Value Theorem 1.8, we have, for each n,

$$|p_n - p| = |g(p_{n-1}) - g(p)| = |g'(\xi_n)||p_{n-1} - p| \leq k|p_{n-1} - p|,$$
$$\xi_n \in (a, b)$$

Applying this inequality inductively gives

$$|p_n - p| \leq k|p_{n-1} - p| \leq k^2|p_{n-2} - p| \leq \ldots \leq k^n|p_0 - p| \qquad (2.4)$$

Since 0 < k < 1, we have $lim_{n\to\infty} k^n = 0$ and

$$\lim_{n\to\infty} |p_n - p| \leq \lim_{n\to\infty} k^n|p_0 - p| = 0$$

Hence$\{p_n\}_{n=0}^{\infty}$ converges to p.

Corollary 2.5 If g satisfies the hypotheses of Theorem 2.4, then bounds for the error involved in using p_n to approximate p are given by

$$|p_n - p| \leq k^n max\{p_0 - a, b - p_0\} \qquad (2.5)$$

and

$$|p_n - p| \leq \frac{k^n}{1-k}|p_1 - p_0|, \qquad for\ all\ \ n \geq 1 \qquad (2.6)$$

Proof $\boldsymbol{p} \in [\boldsymbol{a}, \boldsymbol{b}]$, the first bound follows from Inequality (2.4)

$$|p_n - p| \leq k^n|p_0 - p| \leq k^n max\{p_0 - a, b - p_0\}$$

For n ≥ 1, the procedure used in the proof of Theorem 2.4 implies that

$$|p_{n+1} - p_n| = |g(p_n) - g(p_{n-1})| \leq k|p_n - p_{n-1}| \leq \ldots \leq k^n|p_1 - p_0|$$

Continued

Thus for m > n ≥ 1,

$$\begin{aligned}|p_m - p_n| &= |p_m - p_{m-1} + p_{m-1} - \cdots + p_{n+1} - p_n| \\ &\le |p_m - p_{m-1}| + |p_{m-1} - p_{m-2}| + \cdots + |p_{n+1} - p_n| \\ &\le k^{m-1}|p_1 - p_0| + k^{m-2}|p_1 - p_0| + \cdots + k^n|p_1 - p_0| \\ &= k^n|p_1 - p_0|\left(1 + k + k^2 + \cdots + k^{m-n-1}\right).\end{aligned}$$

By Theorem 2.3,

$$|p - p_n| = \lim_{m\to\infty} |p_m - p_n| \le \lim_{m\to\infty} k^n|p_1 - p_0| \sum_{i=0}^{m-n-1} k^i \le k^n|p_1 - p_0| \sum_{i=0}^{\infty} k^i.$$

Continued

But $\sum_{i=0}^{\infty} k^i$ is a geometric series. This sequence converges to 1/(1 − k), which gives the second bound:

$$|p - p_n| \leq \frac{k^n}{1-k}|p_1 - p_0|$$

- Both inequalities in the corollary relate the rate at which $\{p_n\}_{n=0}^{\infty}$ converges to the bound k on the first derivative. The rate of convergence depends on the factor k^n. The smaller the value of k, the faster the convergence, which may be very slow if k is close to 1.

Illustration Let us reconsider the various fixed-point schemes described in the preceding illustration

(a) For $g_1(x) = x - x^3 - 4x^2 + 10 \rightarrow g_1(1)$=6,$g_1(2) = -12$,so g_1 does not map [1, 2] into itself.

Moreover,$g'_1(x) = 1 - 3x^2 - 8x$, so$|g'_1(x)| > 1$

Although Theorem 2.4 does not guarantee that the method must fail for this choice of g, there is no reason to expect convergence.

Illustration

(b)

- $g_2(x) = [\frac{10}{x} - 4x]^{1/2} \rightarrow g_2$ does not map [1, 2] into [1, 2]
- the sequence $\{p_n\}_{n=0}^{\infty}$ is not defined when $p_0 = 1.5$
- there is no interval containing p ≈ 1.365 such that $|g'_2(x)| < 1$

→ There is no reason to expect that this method will converge.

Illustration

(c) For $g_3(x) = \frac{1}{2}(10 - x^3)^{1/2}$

- $g'_3(x) = -\frac{3}{4}x^2(10 - x^3)^{1/2} < 0, on[1,2] \rightarrow g_3$ is strictly decreasing on [1, 2].
- $|g'_3(2)| \approx 2.12$, so the condition $|g'_3(x)| \leq k < 1$ fails on [1, 2].
- A closer examination of the sequence $\{p_n\}_{n=0}^{\infty}$ starting with $p_0 = 1.5$ shows that it suffices to consider the interval [1, 1.5] instead of [1, 2].
- $g'_3(x)<0$ and g_3 is strictly decreasing,$1 < 1.28 \approx g_3(1.5) \leq g_3(x) \leq g_3(1) = 1.5$

- This shows that g_3 maps the interval [1, 1.5] into itself.
- $|g'_3(x)| \leq |g'_3(1.5)| \approx 0.66$

so Theorem 2.4 confirms the convergence of which we were already aware.

Illustration

(d)

- $g_4(x) = (10/(4+x))^{1/2} \rightarrow |g'_4(x)| = \left|\frac{-5}{\sqrt{10}(4+x)^{3/2}}\right| \leq \frac{5}{\sqrt{10}(5)^{3/2}} < 0.15$
- The bound on the magnitude of $g'_4(x)$ is much smaller than the bound (found in (c)) on the magnitude of $g'_3(x)$

This explains the more rapid convergence using g_4.

(e) $g_5(x) = x - \frac{x^3+4x^2-10}{3x^2+8x}$

- converges much more rapidly than our other choices.

Question: How can we find a fixed-point problem that produces a sequence that reliably and rapidly converges to a solution to a given root-finding problem?

Answer: Manipulate the root-finding problem into a fixed point problem that satisfies the conditions of Fixed-Point Theorem 2.4 and has a derivative that is as small as possible near the fixed point.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 2-3

Newton's Method and Its Extensions

- numerical methods for solving a root-finding problem.

Newton's Method

Newton's method, is based on Taylor polynomials.

- Suppose that $f \in C^2[a,b]$.
- Let $p_0 \in [a,b]$be an approximation to p such that $f'(p_o) \neq 0$ and $|p - p_0|$ is "small."

Consider the first Taylor polynomial for $f(x)$ expanded about p_0 and evaluated at $x = p$.

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''(\xi(p))$$

where $\xi(p)$ lies between p and p_0 . Since $f(p) = 0$, this equation gives

$$0 = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''(\xi(p))$$

assuming that since $|p - p_0|$ is small, the term involving $(p - p_0)^2$ is much smaller, so

$$0 \approx f(p_0) + (p - p_0)f'(p_0)$$

$$p \approx p_0 - \frac{f(p_0)}{f'(p_0)} \equiv p_1$$

This sets the stage for Newton's method, which starts with an initial approximation p_0 and generates the sequence $\{p_n\}_{n=0}^{\infty}$, by

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \qquad n \geq 1 \qquad (2.7)$$

Figure 2.8 illustrates how the approximations are obtained using successive tangents.

Figure 2.8

ALGORITHM 2.3

To find a solution to $f(x) = 0$ given an initial approximation p_0:

INPUT *initial approximation* p_0*; tolerance TOL; maximum number of iterations* N_0.

OUTPUT *approximate solution p or message of failure.*

Step 1 *Set i = 1;*

Step 2 *While* $i \leq N_0$ *do Steps 3–6.*

Step 3 *Set* $p_0 - f(p_0)/f'(p_0)$ *(Compute pi.)*

Step 4 *If* $|p - p_0| < TOL$ *then*

OUTPUT (p); (The procedure was successful.)

STOP.

Step 5 *Set* $i = i + 1$.

Step 6 *Set* $p_0 = p$.(Update p_0)

Step 7 OUTPUT ('The method failed after N_0 iterations, N_0 =',N_0);

(The procedure was unsuccessful.)

STOP

The stopping-technique inequalities given with the Bisection method are applicable to Newton's method.

- select a tolerance $\varepsilon > 0$, and construct $p_1, \ldots, p_N$

$$|p_N - p_{N-1}| < \varepsilon \qquad (2.8)$$

$$\frac{|p_N - p_{N-1}|}{|p_N|} < \varepsilon, p_N \neq 0 \qquad (2.9)$$

or

$$|f(p_N)| < \varepsilon \qquad (2.10)$$

A form of Inequality (2.8) is used in Step 4 of Algorithm 2.3

Newton's method is a functional iteration technique with $p_n = g(p_{n-1})$

$$g(p_{n-1}) = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, n \geq 1$$

Illustration

The equation $x^3 + 4x^2 - 10 = 0$ has a unique root in [1,2].

$(a)\ x = g_1(x) = x - x^3 - 4x^2 + 10,$

$(b)\ x = g_2(x) = \left(\frac{10}{x} - 4x\right)^{\frac{1}{2}}$

$(c)\ x = g_3(x) = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$

$(d)\ x = g_4(x) = \left(\frac{10}{4+x}\right)^{\frac{1}{2}}$

$(e)\ x = g_5(x) = x - \frac{x^3+4x^2-10}{3x^2+8x}$

With $p_0 = 1.5$

Table 2.2

n	(a)	(b)	(c)	(d)	(e)
0	1.5	1.5	1.5	1.5	1.5
1	−0.875	0.8165	1.286953768	1.348399725	1.373333333
2	6.732	2.9969	1.402540804	1.367376372	1.365262015
3	−469.7	$(-8.65)^{1/2}$	1.345458374	1.364957015	1.365230014
4	1.03×10^8		1.375170253	1.365264748	1.365230013
5			1.360094193	1.365225594	
6			1.367846968	1.365230576	
7			1.363887004	1.365229942	
8			1.365916734	1.365230022	
9			1.364878217	1.365230012	
10			1.365410062	1.365230014	
15			1.365223680	1.365230013	
20			1.365230236		
25			1.365230006		
30			1.365230013		

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \qquad n \geq 1 \qquad (2.7)$$

- It is clear from Equation (2.7) that Newton's method cannot be continued if $f'(p_{n-1})$ for some n.

- In fact, the method is most effective when f' is bounded away from zero near p.

Example 1

Consider the function $f(x) = \cos x - x = 0$. Approximate a root of f using **(a)** a fixed-point method, and **(b)** Newton's Method

Solution

(a) a solution to this root-finding problem is also a solution to the fixed-point problem $x = \cos x$

Table 2.3 shows the results of fixed point iteration with $p_0 = \pi/4$. The best we could conclude from these results is that p ≈ 0.74.

Table 2.3

n	p_n
0	0.7853981635
1	0.7071067810
2	0.7602445972
3	0.7246674808
4	0.7487198858
5	0.7325608446
6	0.7434642113
7	0.7361282565

A single fixed-point p lies in $[0, \pi/2]$

Solution

(b) $f'(x) = -\sin x - 1$, Starting again with $p_0 = \pi/4$. we generate the sequence defined, for n ≥ 1, by

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} = p_{n-1} - \frac{\cos p_{n-1} - p_{n-1}}{-\sin p_{n-1} - 1}$$

An excellent approximation is obtained with n = 3.

Table 2.4

Newton's Method

n	p_n
0	0.7853981635
1	0.7395361337
2	0.7390851781
3	0.7390851332
4	0.7390851332

Table 2.3

n	p_n
0	0.7853981635
1	0.7071067810
2	0.7602445972
3	0.7246674808
4	0.7487198858
5	0.7325608446
6	0.7434642113
7	0.7361282565

Table 2.4

Newton's Method	
n	p_n
0	0.7853981635
1	0.7395361337
2	0.7390851781
3	0.7390851332
4	0.7390851332

- Newton's method can provide extremely accurate approximations with very few iterations.

Convergence using Newton's Method

Theorem 2.6 Let $f \in C^2[a,b]$ .If $p \in (a,b)$ is such that $f(p) = 0$ and $f'(p) \neq 0$,then there exists a $\delta > 0$ such that Newton's method generates a sequence $\{p_n\}_{n=1}^{\infty}$ converging to p for any initial approximation $p_0 \in [p-\delta, p+\delta]$.

- The convergence theorem for Newton's method illustrates the theoretical importance of the choice of p_0.

Proof

Order of Convergence

Definition 2.7 Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n. If positive constants λ and α exist with

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^{\alpha}} = \lambda$$

then $\{p_n\}_{n=0}^{\infty}$ **converges to p of order α**, with **asymptotic error constant** λ.

- An iterative technique of the form $p_n = g(p_{n-1})$ is said to be of order α if the sequence $\{p_n\}_{n=0}^{\infty}$ converges to the solution $p = g(p)$ of order α.

- In general, a sequence with a high order of convergence converges more rapidly than a sequence with a lower order.
- The asymptotic constant affects the speed of convergence but not to the extent of the order.

Two cases of order are given special attention.

(i) If $\alpha = 1$ (and $\lambda < 1$), the sequence is **linearly convergent**.

(ii) If $\alpha = 2$, the sequence is **quadratically convergent**.

The illustration compares a linearly convergent sequence to one that is quadratically convergent.

Illustration Suppose that $\{p_n\}_{n=0}^{\infty}$ is linearly convergent to 0 with

$$\lim_{n\to\infty} \frac{|p_{n+1}|}{|p_n|} = 0.5$$

and that $\{\tilde{p}_n\}_{n=0}^{\infty}$ is quadratically convergent to 0

$$\lim_{n\to\infty} \frac{|\tilde{p}_{n+1}|}{|\tilde{p}_n|^2} = 0.5$$

assume that for each n we have

$$\frac{|p_{n+1}|}{|p_n|} \approx 0.5 \quad \text{and} \quad \frac{|\tilde{p}_{n+1}|}{|\tilde{p}_n|} \approx 0.5$$

Continued For the linearly convergent scheme, this means that

$$|p_n - 0| = |p_n| \approx 0.5|p_{n-1}| \approx (0.5)^2|p_{n-2}| \approx \ldots \approx (0.5)^n|p_0|$$

whereas the quadratically convergent procedure has

$$|\tilde{p}_n - 0| = |\tilde{p}_n| \approx 0.5|\tilde{p}_{n-1}|^2 \approx (0.5)[0.5|\tilde{p}_{n-2}|^2]^2$$

$$= (0.5)^3|\tilde{p}_{n-2}|^4 \approx (0.5)^3[0.5|\tilde{p}_{n-3}|^2]^4$$

$$= (0.5)^7|\tilde{p}_{n-3}|^8 \approx \cdots \approx (0.5)^{2^n-1}|\tilde{p}_0|^{2^n}$$

Table 2.7 illustrates the relative speed of convergence of the sequences to 0 if $|p_0| = |\tilde{p}_0| = 1$

n	Linear Convergence Sequence $\{p_n\}_{n=0}^{\infty}$ $(0.5)^n$	Quadratic Convergence Sequence $\{\tilde{p}_n\}_{n=0}^{\infty}$ $(0.5)^{2^n-1}$
1	5.0000×10^{-1}	5.0000×10^{-1}
2	2.5000×10^{-1}	1.2500×10^{-1}
3	1.2500×10^{-1}	7.8125×10^{-3}
4	6.2500×10^{-2}	3.0518×10^{-5}
5	3.1250×10^{-2}	4.6566×10^{-10}
6	1.5625×10^{-2}	1.0842×10^{-19}
7	7.8125×10^{-3}	5.8775×10^{-39}

The quadratically convergent sequence is within 10^{-38} of 0 by the seventh term. At least 126 terms are needed to ensure this accuracy for the linearly convergent sequence.

Theorem 2.8 Let $g \in C[a,b]$ be such that $g(x) \in [a,b]$, for all $x \in [a,b]$. Suppose, in addition, that g' is continuous on (a,b) and a positive constant $k < 1$ exists with

$$|g'(x)| \leq k, \qquad for\ all\ x \in (a,b)$$

If $g'(p) \neq 0$, then for any number $p_0 \neq p$ in [a,b], the sequence

$$p_n = g(p_{n-1}), \qquad for\ n \geq 1$$

converges only linearly to the unique fixed point p in [a, b].

Proof

Note: Theorem 2.8 implies that higher-order convergence for fixed-point methods of the form $g(p) = p$ can occur only when $g'(p) = 0$.

Theorem 2.9 Let p be a solution of the equation $x = g(x)$. Suppose that $g'(p) = 0$ and g'' is continuous with $|g''(x)| < M$ on an open interval I containing p. Then there exists a $\delta > 0$ such that,for $p_0 \in [p - \delta, p + \delta]$, the sequence defined by $p_n = g(p_{n-1})$,when $n \geq 1$, converges at least quadratically to p.Moreover, for sufficiently large values of n,

$$|p_{n+1} - p| < \frac{M}{2}|p_n - p|^2$$

Proof

Summary Theorems 2.8 and 2.9 tell us that our search for quadratically convergent fixed-point methods should point in the direction of functions whose derivatives are zero at the fixed point.

- For a fixed point method to converge quadratically we need to have both $g(p) = p$, and $g'(p) = 0$.

Consider the sequence

$$p_n = g(p_{n-1}), \qquad for\ \ n \geq 1$$

for g in the form

$$g(x) = x - \Phi(x)f(x)$$

For the iterative procedure derived from g to be quadratically convergent, we need to have $g'(p) = 0$ when $f(p) = 0$. Because

$$g'(x) = 1 - \Phi'(x)f(x) - f'(x)\Phi(x)$$

also $f(p) = 0$, we have

$$g'(p) = 1 - \Phi'(p)f(p) - f'(p)\Phi(p) = 1 - \Phi'(p) \cdot 0 - f'(p)\Phi(p)$$
$$= 1 - f'(p)\Phi(p)$$

$g'(p) = 0$ if and only if $\Phi(p) = 1/f'(p)$

If we let $\Phi(x) = 1/f'(x)$, then we will ensure that $\Phi(p) = 1/f'(p)$ and produce the quadratically convergent procedure

$$p_n = g(p_{n-1}) = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$$ Newton's method

- If $f(p) = 0$ and $f'(p) \neq 0$, then for starting values sufficiently close to p, Newton's method will converge at least quadratically.

Example Assume that $a > 0$, for the equation $x^2 - a = 0$, prove that for any initial value $x_0 > 0$, Newton's method convergent to $\sqrt{a}$.

The Secant Method

Shortcomings of Newton's method--: the need to know the value of the derivative of f at each approximation.

By definition

$$f'(p_{n-1}) = \lim_{x \to p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}}$$

If p_{n-2} is close to p_{n-1}, then

$$f'(p_{n-1}) \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}} = \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}$$

Using this approximation for $f'(p_{n-1})$ in Newton's formula gives

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1}-p_{n-2})}{f(p_{n-1})-f(p_{n-2})}. \quad (2.7')$$

This technique is called the **Secant method**. In contrast, each step of **Newton's method** requires an evaluation of both the function and its derivative.

Figure 2.10

ALGORITHM 2.4

To find a solution to $f(x) = 0$ given an initial approximation p_0 and p_1:

INPUT *initial approximation* p_0, p_1*; tolerance TOL; maximum number of iterations* N_0.

OUTPUT *approximate solution p or message of failure.*

Step 1 *Set i = 2;*

$$q_0 = f(p_0);$$

$$q_1 = f(p_1);$$

Step 2 *While* $i \leq N_0$ *do Steps 3–6.*

Step 3 *Set* $p = p_1 - q_1(p_1 - p_0)/(q_1 - q_0)$. *(Compute pi.)*

Step 4 *If* $|p - p_0| < TOL$ *then*

OUTPUT (p); (The procedure was successful.)

STOP.

Step 5 *Set i = i + 1.*

Step 6 *Set* $p_0 = p$.(Update p_0, q_0, p_1, q_1)$q_0 = q_1; p_1 = p; q_1 = f(p)$.

Step 7 OUTPUT ('The method failed after N_0 iterations, N_0 =',N_0);

(The procedure was unsuccessful.)

STOP

Example 2

Use the Secant method to find a solution to $x = \cos x$, and compare the approximations with those given in Example 1 which applied Newton's method.

Solution

For the Secant method we need two initial approximations.

use $p_0 = 0.5, p_1 = \pi/4$. Succeeding approximations are generated by the formula

$$p_n = p_{n-1} - \frac{(p_{n-1} - p_{n-2})(\cos p_{n-1} - p_{n-1})}{(\cos p_{n-1} - p_{n-1}) - (\cos p_{n-2} - p_{n-2})}$$

Comparing the results in Table 2.5 from the Secant method and Newton's method we see that the Secant method approximation p_5 is accurate to the tenth decimal place, whereas Newton's method obtained this accuracy by p_3

Table 2.5

	Secant		Newton
n	p_n	n	p_n
0	0.5	0	0.7853981635
1	0.7853981635	1	0.7395361337
2	0.7363841388	2	0.7390851781
3	0.7390581392	3	0.7390851332
4	0.7390851493	4	0.7390851332
5	0.7390851332		

Newton's method or the Secant method is often used to refine an answer obtained by another technique.

Theorem 2.9' Let p be a solution of the equation $f(x) = 0$. Suppose that f'' is continuous on $[p-\delta, p+\delta]$ and $f'(x) \neq 0$, for initial values $p_0, p_1 \in [p-\delta, p+\delta]$, when δ is sufficiently small, (2.7') can convergent to p of order $\frac{1+\sqrt{5}}{2} \approx 1.618$.

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1}-p_{n-2})}{f(p_{n-1})-f(p_{n-2})}. \quad (2.7')$$

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 2-4

Iterative Methods for multiple roots

Multiple Roots

Definition 2.10 A solution p of $f(x) = 0$ is a **zero of multiplicity** m of f if for $x \neq p$, we can write $f(x) = (x - p)^m q(x)$, where $\lim_{x \to p} q(x) \neq 0$.

- $q(x)$ represents that portion of $f(x)$ that does not contribute to the zero of f.

Theorem 2.11 The function $f \in C^1[a, b]$ has a simple zero at p in (a, b) if and only if $f(p) = 0$, but $f'(p) \neq 0$.

- The Theorem gives a means to easily identify simple zeros of a function, those that have multiplicity one.

Theorem 2.12 The function $f \in C^m[a,b]$ has a zero of multiplicity m at p in (a,b) if and only if

$$0 = f(p) = f'(p) = f''(p) = \ldots = f^{(m-1)}(p), f^{(m)}(p) \neq 0$$

- an interval about p exists where Newton's method converges quadratically to p for any initial approximation $p_0 = p$, provided that p is a simple zero.

Example 1 Let $f(x) = e^x - x - 1$. **(a)** Show that f has a zero of multiplicity 2 at $x = 0$. **(b)** Show that Newton's method with $p_0 = 1$ converges to this zero but not quadratically.

Solution **(a)** We have

$$f(x) = e^x - x - 1 \qquad f'(x) = e^x - 1 \qquad f''(x) = e^x$$

so

$$f(0) = e^0 - 0 - 1 = 0 \qquad f'(0) = e^0 - 1 = 0 \qquad f''(0) = e^0 = 1$$

f has a zero of multiplicity 2 at $x = 0$

Solution **(b)** The first two terms generated by Newton's method applied to f with $p_0 = 1$ are

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)} = 1 - \frac{e-2}{e-1} \approx 0.58198$$

and

$$p_2 = p_1 - \frac{f(p_1)}{f'(p_1)} \approx 0.58198 - \frac{0.20760}{0.78957} \approx 0.31906$$

The first sixteen terms of the sequence generated by Newton's method are shown in Table 2.8. The sequence is clearly converging to 0, but not quadratically.

Table 2.8

n	p_n
0	1.0
1	0.58198
2	0.31906
3	0.16800
4	0.08635
5	0.04380
6	0.02206
7	0.01107
8	0.005545
9	2.7750×10^{-3}
10	1.3881×10^{-3}
11	6.9411×10^{-4}
12	3.4703×10^{-4}
13	1.7416×10^{-4}
14	8.8041×10^{-5}
15	4.2610×10^{-5}
16	1.9142×10^{-6}

The example shows that quadratic convergence might not occur if the zero is not simple.

- One method of handling the problem of multiple roots of a function f is to define

$$\mu(x) = \frac{f(x)}{f'(x)}$$

If p is a zero of f of multiplicity m with $f(x) = (x-p)^m q(x)$, then

also has a zero at p. However, $q(p) \neq 0$, so

$$\frac{q(p)}{mq(p) + (p-p)q'(p)} = \frac{1}{m} \neq 0$$

Continued and p is a simple zero of $\mu(x)$.Newton's method can then be applied to $\mu(x)$ to give

$$g(x) = x - \frac{\mu(x)}{\mu'(x)} = x - \frac{f(x)/f'(x)}{\{[f'(x)]^2 - [f(x)][f''(x)]\}/[f'(x)]^2}$$

which simplifies to

$$g(x) = x - \frac{f(x)f'(x)}{[f'(x)]^2 - [f(x)][f''(x)]} \qquad (2.13)$$

- If g has the required continuity conditions, functional iteration applied to g will be quadratically convergent regardless of the multiplicity of the zero of f.
- Theoretically, the only drawback to this method is the additional calculation of $f''(x)$ and the more laborious procedure of calculating the iterates.

Example 2 In Example 1 it was shown that $f(x) = e^x - x - 1$ has a zero of multiplicity 2 at $x = 0$ and that Newton's method with $p_0 = 1$ converges to this zero but not quadratically. Show that the modification of Newton's method as given in Eq. (2.13) improves the rate of convergence.

Solution Modified Newton's method gives

$$p_1 = p_0 - \frac{f(p_0)f'(p_0)}{f'(p_0)^2 - f(p_0)f''(p_0)} = 1 - \frac{(e-2)(e-1)}{(e-1)^2 - (e-2)e}$$
$$\approx -2.3421061 \times 10^{-1}$$

This is considerably closer to 0 than the first term using Newton's method, which was 0.58918.

The following illustrates that the modified Newton's method converges quadratically even when in the case of a simple zero.

Illustration In Section 2.2 we found that a zero of $f(x) = x^3 + 4x^2 - 10 = 0$ is $p = 1.36523001$.Here we will compare convergence for a simple zero using both Newton's method and the modified Newton's method listed in Eq. (2.13).

Solution Let**(i)**$p_n = p_{n-1} - \frac{p_{n-1}^3+4p_{n-1}^2-10}{3p_{n-1}^2+8p_{n-1}}$ from Newton's method

and, from the Modified Newton's method given by Eq. (2.13),

(ii)$$p_n = p_{n-1} - \frac{(p_{n-1}^3+4p_{n-1}^2-10)(3p_{n-1}^2+8p_{n-1})}{(3p_{n-1}^2+8p_{n-1})^2-(p_{n-1}^3+4p_{n-1}^2-10)(6p_{n-1}+8)}$$

With $p_0 = 1.5$, we have

Newton's method

$$p_1 = 1.37333333, p_2 = 1.36526201, p_3 = 1.36523001$$

Modified Newton's method

$$p_1 = 1.35689898, p_2 = 1.36519585, p_3 = 1.36523001$$

Both methods are rapidly convergent to the actual zero.

- in the case of a simple zero the original Newton's method requires substantially less computation.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 2-5

Accelerating Convergence

Aitken's Δ^2 Method

For the iteration $p_{n+1} = g(p_n)(n = 0,1,\cdots)$, we can compute $p_1 = g(p_0), p_2 = g(p_1)$, if p_0 and the solution p is close sufficiently, p_1 is a better approximation than p_0. We use secant method for $x - g(x) = 0$, we can get

$$p \approx \frac{p_{n+2}p_n - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n} = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n} = \hat{p}_1.$$

Aitken's Δ^2 method is based on the assumption that the sequence $\{\hat{p}_n\}_{n=0}^{\infty}$, defined by

$$\hat{p}_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n},$$

converges more rapidly to p than does the original sequence $\{p_n\}_{n=0}^{\infty}$.

The corresponding iteration function is $\psi(x) = x - \frac{(g(x)-x)^2}{g(g(x))-2g(x)+x}$.

Example 1 The sequence $\{p_n\}_{n=1}^{\infty}$, where $p_n = \cos(1/n)$, converges linearly to $p = 1$. Determine the first five terms of the sequence given by Aitken's Δ^2 method.

Solution:

$$\hat{p}_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$$

From Table 2.10. It certainly appears that $\{p_n\}_{n=1}^{\infty}$ converges more rapidly to $p = 1$ than does $\{p_n\}_{n=1}^{\infty}$.

Table 2.10

n	p_n	$\hat{p}_n$
1	0.54030	0.96178
2	0.87758	0.98213
3	0.94496	0.98979
4	0.96891	0.99342
5	0.98007	0.99541
6	0.98614	
7	0.98981	

The Δ notation of Aitken's Δ^2 method

Definition 2.13 For a given sequence $\{p_n\}_{n=1}^{\infty}$, the **forward difference** Δp_n

$$\Delta p_n = p_{n+1} - p_n, \text{ for } n \geq 0.$$

Higher powers of the operator Δ are defined recursively by

$$\Delta^k p_n = \Delta(\Delta^{k-1} p_n), \text{ for } k \geq 2.$$

The definition implies that

$$\Delta^2 p_n = \Delta(p_{n+1} - p_n) = \Delta p_{n+1} - \Delta p_n = (p_{n+2} - p_{n+1}) - (p_{n+1} - p_n).$$

So $\Delta^2 p_n = p_{n+2} - 2p_{n+1} + p_n$, and the formula for $\hat{p}_n$ can be written as

$$\hat{p}_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}, \text{ for } n \geq 0. \qquad (2.15)$$

So far, we know that the sequence $\{p_n\}_{n=0}^{\infty}$, converges to p more rapidly than does the original sequence $\{p_n\}_{n=0}^{\infty}$. The term "more rapid" convergence will be explained in Theorem 2.14.

Theorem 2.14 Suppose that $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges linearly to the limit p and that

$$\lim_{n\to\infty} \frac{p_{n+1} - p}{p_n - p} < 1 .$$

Then the Aitken's Δ^2 sequence $\{p_n\}_{n=0}^{\infty}$ converges to p faster than $\{p_n\}_{n=0}^{\infty}$ in the sense that

$$\lim_{n\to\infty} \frac{\hat{p}_n - p}{p_n - p} = 0$$

Steffensen's Method

Then we can choose the simple form $g(x) = x + f(x)$, the corresponding iteration function is

$$\Psi(x) = x - \frac{f(x)}{f[x+f(x)]-f(x)} f(x),$$

$$p_{n+1} = p_n - \frac{[f(p_n)]^2}{f[p_n + f(p_n)] - f(p_n)}, n = 0,1, \cdots$$

Steffensen's method constructs the same first four terms, p_0, p_1, p_2 and $\hat{p}_0$ as

$$p_0, p_1 = g(p_0), p_2 = g(p_1), \hat{p}_0 = \{\Delta^2\}(p_0),$$

we assume that $\hat{p}_0$ is a better approximation to p than is p_2 and apply fixed-point iteration to $\hat{p}_0$ instead of p_2. Using this notation, the sequence is

$$p_0^{(0)}, p_1^{(0)} = g(p_0^{(0)}), p_2^{(0)} = g(p_1^{(0)}), p_0^{(1)} = \{\Delta^2\}(p_0^{(0)}), p_1^{(1)} = g\left(p_0^{(1)}\right), \ldots$$

Every third term of the Steffensen sequence is generated by Eq. (2.15); the others use fixed-point iteration on the previous term.

$$\hat{p}_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}, (2.15)$$

Algorithm 2.6 Steffensen's

To find a solution to $p = g(p)$ given an initial approximation p_0:

INPUT initial approximation p_0; tolerance TOL; maximum number of iterations N_0.

OUTPUT approximate solution p or message of failure.

Step 1 Set $i = 1$.

Step 2 While $i \leq N_0$ do Steps 3–6.

Step 3 Set $p_1 = g(p_0)$; (*Compute* $p_1^{(i-1)}$.)

$p_2 = g(p_1)$; (*Compute* $p_2^{(i-1)}$.)

$p = p_0 - (p_1 - p_0)^2/(p_2 - 2p_1 + p_0)$. (*Compute* $p_0^{(i)}$.)

Step 4 If $|p - p_0| < TOL$ then

OUTPUT (p); (*Procedure completed successfully.*)
STOP.

Step 5 Set $i = i + 1$.

Step 6 Set $p_0 = p$. (*Update* p_0.)

Step 7 OUTPUT ('*Method failed after* N_0 *iterations,* N_0='$, N_0$);

(*Procedure completed unsuccessfully.*)
STOP.

Note: $\Delta^2 p_n$ might be 0. If this occurs, we terminate the sequence and select $p_2^{(n-1)}$ as the best approximation.

Illustration To solve $x^3 + 4x^2 - 10 = 0$ using Steffensen's method, the fixed-point method

$$g(x) = \left(\frac{10}{x+4}\right)^{1/2}$$

Applying Steffensen's procedure with $p_0 = 1.5$ gives the values in Table 2.11.

Table 2.11

k	$p_0^{(k)}$	$p_1^{(k)}$	$p_2^{(k)}$
0	1.5	1.348399725	1.367376372
1	1.365265224	1.365225534	1.365230583
2	1.365230013		

From the Illustration, it appears that Steffensen's method gives quadratic convergence without evaluating a derivative.

Theorem 2.15

Suppose that $x = g(x)$ has the solution p with $g'(p) \neq 1$. If there exists a $\delta > 0$ such that $g \in C^3[p-\delta, p+\delta]$, then Steffensen's method gives quadratic convergence for any $p_0 \in [p-\delta, p+\delta]$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Chapter 3

Interpolation and Polynomial Approximation

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 3-1

Interpolation and the Lagrange Polynomial

插值和拉格朗日多项式

Example

$$f(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

x	0.520	0.521	0.522	0.524	$\cdots$
$f(x)$	0.53790	0.53876	0.53962	0.54048	$\cdots$

$f(0.52136)$?

Description of the question

For given data

$$\{x_i, y_i\} \quad (i = 0,1,\cdots,n)$$

with $x_0 < x_1 < \cdots < x_n$, we seek a function $\varphi(x)$ such that

$$\varphi(x_i) = y_i \quad (i = 0,1,\cdots,n)$$

We call φ an interpolating function, or simply interpolant, for the given data.

Some families of functions commonly used for interpolant include

- Polynomials
- Trigonometric functions
- Rational functions
- Piecewise polynomials

The question that we need to consider:

- existence and uniqueness
- if it exists, how to construct it?
- How to evaluate the error if we use $\varphi(x)$ to approximate $f(x)$

One of the most useful and well-known classes of functions mapping the set of real numbers into itself is the algebraic polynomials, the set of functions of the form

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

where n is a nonnegative integer and $a_0, \ldots, a_n$ are real constants.

One reason for their importance is that they uniformly approximate continuous functions. Weierstrass Approximation Theorem.

Theorem 3.1 (Weierstrass Approximation Theorem)

Suppose that f is defined and continuous on $[a, b]$. For each $\varepsilon > 0$, there exists a polynomial $P(x)$, with the property that

$$|f(x) - P(x)| < \varepsilon, \text{ for all } x \text{ in } [a, b].$$

Another important reason for considering the class of polynomials in the approximation of functions is that the derivative and indefinite integral of a polynomial are easy to determine and are also polynomials.

Figure 3.1

The Taylor polynomials.

Drawbacks: The Taylor polynomials agree as closely as possible with a given function at a specific point, but they concentrate their accuracy near that point. A good interpolation polynomial needs to provide a relatively accurate approximation over an entire interval, and Taylor polynomials do not generally do this.

Example

calculate the first six Taylor polynomials about $x_0 = 0$ for $f(x) = e^x$.

$$P_0(x) = 1, P_1(x) = 1 + x, P_2(x) = 1 + x + \frac{x^2}{2}, P_3(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6},$$

$$P_4(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}, \qquad P_5(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$$

Notice that even for the higher-degree polynomials, the error becomes progressively worse as we move away from zero.

Figure 3.2

Although better approximations are obtained for $f\ (x) = e^x$ if higher-degree Taylor polynomials are used, this is not true for all functions.

Example

using Taylor polynomials of various degrees for $f\ (x) = \frac{1}{x}$ expanded about $x_0 = 1$ to approximate $f\ (3) = \frac{1}{3}$.

the Taylor polynomials are

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(1)}{k!}(x-1)^k = \sum_{k=0}^{n} (-1)^k (x-1)^k.$$

To approximate $f(3) = \frac{1}{3}$ by $P_n(3)$ for increasing values of n, see the values in Table 3.1—rather a dramatic failure! When we approximate $f(3) = \frac{1}{3}$ by $P_n(3)$ for larger values of n, the approximations become increasingly inaccurate.

n	0	1	2	3	4	5	6	7
$P_n(3)$	1	−1	3	−5	11	−21	43	−85

Table 3.1

Summary: The primary use of Taylor polynomials in numerical analysis is not for approximation purposes, but for the derivation of numerical techniques and error estimation.

The existence of interpolant

If the polynomial is

$$p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$$

Since $p_n(x_i) = f(x_i) = y_i$, we can get

$$\begin{cases} a_0 + a_1 x_0 + \cdots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + \cdots + a_n x_1^n = y_1 \\ \cdots \\ a_0 + a_1 x_n + \cdots + a_n x_n^n = y_n \end{cases}$$

The existence of interpolant

The determinant of the coefficients is

$$D = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & & x_1^n \\ & \vdots & & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} = \prod_{0 \le j < i \le n} (x_i - x_j)$$

$D \neq 0$, $a_i (\mathrm{i} = 0,1,\cdots,n)$ are exist and unique.

Lagrange Interpolating Polynomials

The problem of determining a polynomial of degree one that passes through the distinct points (x_0, y_0) and (x_1, y_1) is the same as approximating a function f for which $f\ (x_0)\ =\ y_0$ and $f\ (x_1)\ =\ y_1$ by means of a first-degree polynomial interpolating, or agreeing with, the values of f at the given points. Using this polynomial for approximation within the interval given by the endpoints is called polynomial interpolation

The linear polynomial through (x_0, y_0) and (x_1, y_1) is

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

$$\boxed{P(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1).}$$

Define the functions

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

Note that

	x_0	x_1
$L_0(x)$	1	0
$L_1(x)$	0	1

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

which implies that

$$P(x_0) = 1 \cdot f(x_0) + 0 \cdot f(x_1) = f(x_0) = y_0$$

and

$$P(x_1) = 0 \cdot f(x_0) + 1 \cdot f(x_1) = y_1 .$$

So P is the unique polynomial of degree at most one that passes through (x_0, y_0) and (x_1, y_1).

Example 1. Determine the linear Lagrange interpolating polynomial that passes through the points(2, 4) and (5, 1).

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

Example Compute $\sqrt{115}$ $(\sqrt{100} = 10, \sqrt{121} = 11)$.

To generalize the concept of linear interpolation, consider the construction of a polynomial of degree at most n that passes through the $n+1$ points

$$(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n)).$$

(See Figure 3.4.)

Figure 3.4

In this case we first construct, for each $k = 0, 1, \dots, n$, a function $L_{n,k}(x)$ with the property that $L_{n,k}(x_i) = \begin{cases} 1, i = k \\ 0, i \neq k \end{cases}$.

$$L_{n,k}(x) = A(x - x_0)(x - x_1) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n).$$

$L_{n,k}(x_k) = 1$, thus

$$L_{n,k}(x) = \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}.$$

$$= \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}.$$

Example

$n = 2$

$$L_{2,0}(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}.$$

$$L_{2,1}(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}.$$

$$L_{2,2}(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}.$$

A sketch of the graph of a typical $L_{n,k}$ (when n is even) is shown in Figure 3.5.

Figure 3.5

The interpolating polynomial is easily described once the form of $L_{n,k}$ is known. This polynomial, called the nth **Lagrange interpolating polynomial**, is defined in the following theorem.

Theorem 3.2

If $x_0, x_1, \ldots, x_n$ are $n+1$ distinct numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$f(x_k) = P(x_k), \qquad \text{for each } k = 0, 1, \ldots, n.$$

This polynomial is given by

$$P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x), \quad (3.1)$$

where, for each $k = 0,1,\dots,n$,

$$L_{n,k}(x) = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)} = \prod_{\substack{i=0\\ i\neq k}}^{n} \frac{x-x_i}{x_k-x_i} \quad (3.2)$$

$$= \frac{w_{n+1}(x)}{(x-x_k)w'_{n+1}(x_k)}$$

where $w_{n+1}(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$

We will write $L_{n,k}(x)$ simply as $L_k(x)$ when there is no confusion as to its degree.

Example 2

(a) Use the numbers (called nodes) $x_0 = 2, x_1 = 2.75$, and $x_2 = 4$ to find the second Lagrange interpolating polynomial for $f(x) = 1/x$.
(b) Use this polynomial to approximate $f(3) = 1/3$

Solution **(a)** We first determine the coefficient polynomials $L_0(x), L_1(x)$, and $L_2(x)$. In nested form they are

$$L_0(x) = \frac{(x-2.75)(x-4)}{(2-2.75)(2-4)} = \frac{2}{3}(x-2.75)(x-4),$$

$$L_1(x) = \frac{(x-2)(x-4)}{(2.75-2)(2.75-4)} = -\frac{16}{15}(x-2)(x-4),$$

and

$$L_2(x) = \frac{(x-2)(x-2.75)}{(4-2)(4-2.75)} = \frac{2}{5}(x-2)(x-2.75).$$

Also, $f(x_0) = f(2) = 1/2, f(x_1) = f(2.75) = 4/11$, and $f(x_2) = f(4) = 1/4$, so

$$P(x) = \sum_{k=0}^{2} f(x_k)\, L_k(x)$$

$$= \frac{1}{3}(x-2.75)(x-4) - \frac{64}{165}(x-2)(x-4) + \frac{1}{10}(x-2)(x-2.75)$$

$$= \frac{1}{22}x^2 - \frac{35}{88}x + \frac{49}{44}.$$

(b) An approximation to $f(3) = 1/3$ (see Figure 3.6) is

$$f(3) \approx P(3) = \frac{9}{22} - \frac{105}{88} + \frac{49}{44} = \frac{29}{88} \approx 0.32955.$$

n	0	1	2	3	4	5	6	7
$P_n(3)$	1	−1	3	−5	11	−21	43	−85

From the above Table, no Taylor polynomial expanded about $x_0 = 1$ could be used to reasonably approximate $f(x) = 1/x$ at $x = 3$.

Figure 3.6

$$P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x), \quad (3.1).$$

Theorem 3.3

Suppose $x_0, x_1, \ldots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each x in $[a, b]$, a number $\xi(x)$ (generally unknown) between $x_0, x_1, \ldots, x_n$, and hence in (a, b), exists with

$$f(x) = P(x) + \frac{f^{n+1}(\xi(x))}{(n+1)!} w_{n+1}(x) \qquad (3.3)$$

where $P(x)$ is the interpolating polynomial given in Eq. (3.1).

Proof

Note that the error form for the Lagrange polynomial is quite similar to that for the Taylor polynomial. The nth Taylor polynomial about x_0 concentrates all the known information at x_0 and has an error term of the form

$$\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)^{n+1}.$$

The Lagrange polynomial of degree n uses information at the distinct numbers $x_0, x_1, \ldots, x_n$ and, in place of $(x-x_0)^n$, its error formula uses a product of the $n+1$ terms $(x-x_0)$, $(x-x_1)$, … ,$(x-x_n)$:

$$\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\ldots(x-x_n).$$

Note about the error: $\left(R_n(x)\frac{f^{n+1}(\xi(x))}{(n+1)!}w_{n+1}(x)\right)$

- $\xi(x)$ is generally unknown, so we can not know the accurate error, we only evaluate its bound. If $\max_{a\le x\le b}|f^{(n+1)}(\xi)|\le M_{n+1}$, then

$$R_n(x)\le\frac{M_{n+1}}{(n+1)!}|w_{n+1}(x)|$$

- If we know M_{n+1}, we can compute $w_{n+1}(x)$, then sometimes, when n is large, the error is small, but it does not hold for all functions.
- The error is relate with $w_{n+1}(x)$, when $w_{n+1}(x)$ is small, the error is small, so if x belongs to $[x_0,x_n]$, w_{n+1} is small.
- If $f(x)$ is a polynomial with degree at most n, then $P(x)=f(x)$.

Example 3

In Example 2 we found the second Lagrange polynomial for $f(x) = 1/x$ on $[2, 4]$ using the nodes $x_0 = 2, x_1 = 2.75,$ and $x_2 = 4$. Determine the error form for this polynomial, and the maximum error when the polynomial is used to approximate $f(x)$ for $x \in [2, 4]$.

Solution Because $f(x) = x^{-1}$, we have

$$f'(x) = -x^{-2}, \qquad f''(x) = 2x^{-3}, \qquad \text{and} \quad f'''(x) = -6x^{-4}.$$

As a consequence, the second Lagrange polynomial has the error form

$$\frac{f'''(\xi(x))}{3!}(x - x_0)(x - x_1)(x - x_2)$$

$$= -(\xi(x))^{-4}(x - 2)(x - 2.75)(x - 4), \text{for } \xi(x) \text{ in } (2, 4).$$

$$R_2(x) = -\big(\xi(x)\big)^{-4}(x-2)(x-2.75)(x-4), \text{for } \xi(x) \text{ in } (2,4).$$

We now need to determine the maximum value on this interval of the absolute value of the polynomial

$$g(x) = (x-2)(x-2.75)(x-4) = x^3 - \frac{35}{4}x^2 + \frac{49}{2}x - 22.$$

Because

$$\left(x^3 - \frac{35}{4}x^2 + \frac{49}{2}x - 22\right)' = 3x^2 - \frac{35}{2}x + \frac{49}{2} = \frac{1}{2}(3x-7)(2x-7),$$

the critical points occur at

$$x = \frac{7}{3}, \text{with } g\left(\frac{7}{3}\right) = \frac{25}{108}, \quad \text{and} \quad x = \frac{7}{2}, \text{with } g\left(\frac{7}{2}\right) = -\frac{9}{16}.$$

$$\left|\frac{f'''(\xi(x))}{3!}\right| |(x-x_0)(x-x_1)(x-x_2)| \le \frac{1}{16}\left|-\frac{9}{16}\right| = \frac{9}{256} \approx 0.03515625$$

Example 4

Suppose a table is to be prepared for the function $f(x) = e^x$, for x in $[0, 1]$. Assume the number of decimal places to be given per entry is $d \geq 8$ and that the difference between adjacent x-values, the step size, is h. What step size h will ensure that linear interpolation gives an absolute error of at most 10^{-6} for all x in $[0, 1]$?

Solution Let $x_0, x_1, \ldots$ be the numbers at which f is evaluated, x be in $[0,1]$, and suppose j satisfies $x_j \leq x \leq x_{j+1}$. Eq. (3.3) implies that the error in linear interpolation is

$$|f(x) - P(x)| = \left|\frac{f^{(2)}(\xi)}{2!}(x - x_j)(x - x_{j+1})\right| = \frac{|f^{(2)}(\xi)|}{2}|(x - x_j)||(x - x_{j+1})|.$$

The step size is h, so $x_j = jh, x_{j+1} = (j+1)h$, and

$$|f(x) - P(x)| \le \frac{|f^{(2)}(\xi)|}{2!}|(x - jh)(x - (j+1)h)|.$$

Hence

$$|f(x) - P(x)| \le \frac{\max_{\xi\in[0,1]} e^{\xi}}{2} \max_{x_j \le x \le x_{j+1}} |(x - jh)(x - (j+1)h)|$$

$$\le \frac{e}{2} \max_{x_j \le x \le x_{j+1}} |(x - jh)(x - (j+1)h|.$$

Consider the function $g(x) = (x - jh)(x - (j+1)h)$, for $jh \le x \le (j+1)h$. Because

$$g'(x) = (x - (j+1)h) + (x - jh) = 2\left(x - jh - \frac{h}{2}\right),$$

the only critical point for g is at $x = jh + \frac{h}{2}$, with $g(jh + h/2) = -\left(\frac{h}{2}\right)^2 = -\frac{h^2}{4}$.

Since $g(jh) = 0$ and $g\big((j+1)h\big) = 0$, the maximum value of $|g'(x)|$ in $[jh, (j+1)h]$ must occur at the critical point which implies that

$$|f(x) - P(x)| \le \frac{e}{2} \max_{x_j \le x \le x_{j+1}} |g(x)| \le \frac{e}{2} \cdot \frac{h^2}{4} = \frac{eh^2}{8}$$

it is sufficient for h to be chosen so that

$$\frac{eh^2}{8} \le 10^{-6}. \text{ This implies that } h < 1.72 \times 10^{-3}.$$

Because $n = (1-0)/h$ must be an integer, a reasonable choice for the step size is $h = 0.001$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 3-2

Data Approximation and Neville's Method

Illustration

Table 3.2 lists values of a function f at various points. The approximations to $f(1.5)$ obtained by various Lagrange polynomials that use this data will be compared to try and determine the accuracy of the approximation.

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Table 3.2

The most appropriate linear polynomial uses $x_0 = 1.3$ and $x_1 = 1.6$ because 1.5 is between 1.3 and 1.6. The value of the interpolating polynomial at 1.5 is

$$P_1(1.5) = \frac{1.5-1.6}{1.3-1.6}f(1.3) + \frac{1.5-1.3}{1.6-1.3}f(1.6)$$
$$= \frac{1.5-1.6}{1.3-1.6}(0.6200860) + \frac{1.5-1.3}{1.6-1.3}(0.4554022) = 0.5102968$$

Two polynomials of degree 2 can reasonably be used, one with , $x_0 = 1.3$, $x_1 = 1.6$ and $x_2 = 1.9$, which gives

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

$$P_{2(1.5)} = \frac{(1.5-1.6)(1.5-1.9)}{(1.3-1.6)(1.3-1.9)}(0.6200860) + \frac{(1.5-1.3)(1.5-1.9)}{(1.6-1.3)(1.6-1.9)}(0.4554022) + \frac{(1.5-1.3)(1.5-1.6)}{(1.9-1.3)(1.9-1.6)}(0.2818186) = 0.5112857,$$

and one with $x_0 = 1.0, x_1 = 1.3,$ and $x_2 = 1.6$, which gives $\hat{P}_2(1.5) = 0.5124715$.

In the third-degree case,

(i) $x_0 = 1.3, x_1 = 1.6, x_2 = 1.9$ and $x_3 = 2.2$, $P_{3(1.5)} = 0.5118302$.

(ii) $x_0 = 1.0, x_1 = 1.3, x_2 = 1.6$ and $x_3 = 1.9$, $\hat{P}_3(1.5) = 0.5118127$.

The fourth-degree Lagrange polynomial

$x_0 = 1.0, x_1 = 1.3, x_2 = 1.6$ and $x_3 = 1.9$, and $x_4 = 2.2$,

$P_4(1.5) = 0.5118200$.

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Because $P_3(1.5)$, $\hat{P}_3(1.5)$, and $P_4(1.5)$ all agree to within 2×10^{-5} units, we expect this degree of accuracy for these approximations. We also expect $P_4(1.5)$ to be the most accurate approximation, since it uses more of the given data.

The function we are approximating is actually the Bessel function of the first kind of order zero, whose value at 1.5 is known to be 0.5118277. Therefore, the true accuracies of the approximations are as follows:

$$
\begin{aligned}
|P_1(1.5) - f(1.5)| &\approx 1.53 \times 10^{-3}, \\
|P_2(1.5) - f(1.5)| &\approx 5.42 \times 10^{-4}, \\
\left|\hat{P}_2(1.5) - f(1.5)\right| &\approx 6.44 \times 10^{-4}, \\
|P_3(1.5) - f(1.5)| &\approx 2.5 \times 10^{-6}, \\
\left|\hat{P}_3(1.5) - f(1.5)\right| &\approx 1.50 \times 10^{-5}, \\
|P_4(1.5) - f(1.5)| &\approx 7.7 \times 10^{-6}.
\end{aligned}
$$

Although $P_3(1.5)$ is the most accurate approximation, if we had no knowledge of the actual value of $f(1.5)$, we would accept $P_4(1.5)$ as the best approximation since it includes the most data about the function. The Lagrange error term derived in Theorem 3.3 cannot be applied here because we have no knowledge of the fourth derivative of f. Unfortunately, this is generally the case.

$$f(x) = \frac{1}{1 + 25x^2}, x \in [-1,1]$$

Neville's Method

Drawbacks of Lagrange interpolation:

- the error term is difficult to apply
- the work done in calculating the approximation by the second polynomial does not lessen the work needed to calculate the third approximation;

We will now derive these approximating polynomials in a manner that uses the previous calculations to greater advantage.

Definition 3.4

Let f be a function defined at $x_0, x_1, x_2, \dots, x_n$, and suppose that $m_1, m_2, \dots, m_k$ are k distinct integers, with $0 \le m_i \le n$ for each i. The Lagrange polynomial that agrees with $f(x)$ at the k points $x_{m_1}, x_{m_2}, \dots, x_{m_k}$ is denoted $P_{m_1,m_2,\dots,m_k}(x)$.

Theorem 3.5

Let f be defined at $x_0, x_1, \ldots, x_k$, , and let x_j and x_i be two distinct numbers in this set. Then

$$P(x) = \frac{(x - x_j)P_{0,1,\ldots,j-1,j+1,\ldots,k}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,\ldots,k}(x)}{(x_i - x_j)}$$

is the kth Lagrange polynomial that interpolates f at the $k + 1$ points $x_0, x_1, \ldots, x_k$.

Proof For ease of notation, let $Q \equiv P_{0,1,\ldots,i-1,i+1,\ldots,k}$ and $\hat{Q} \equiv P_{0,1,\ldots,j-1,j+1,\ldots,k}$. Since $Q(x)$ and $\hat{Q}(x)$ are polynomials of degree $k-1$ or less, $P(x)$ is of degree at most k.

First note that $\hat{Q}(x_i) = f(x_i)$, implies that

$$P(x_i) = \frac{(x_i - x_j)\hat{Q}(x_i) - (x_i - x_i)Q(x_i)}{x_i - x_j} = \frac{(x_i - x_j)}{(x_i - x_j)} f(x_i) = f(x_i)$$

Similarly, since $Q(x_j) = f(x_j)$, we have $P(x_j) = f(x_j)$.

In addition, if $0 \leq r \leq k$ and r is neither i nor j, then $Q(x_r) = \hat{Q}(x_r) = f(x_r)$ .So

$$P(x_r) = \frac{(x_r - x_j)\hat{Q}(x_r) - (x_r - x_i)Q(x_r)}{x_i - x_j} = \frac{(x_i - x_j)}{(x_i - x_j)} f(x_r) = f(x_r)$$

But, by definition, $P_{0,1,\ldots,k}(x)$ is the unique polynomial of degree at most k that agrees with f at $x_0, x_1, \ldots, x_k$. Thus, $P = P_{0,1,\ldots,k}$.

Theorem 3.5 implies that the interpolating polynomials can be generated recursively. For example, we have

$$P_{0,1} = \frac{1}{x_1 - x_0}[(x - x_0)P_1 - (x - x_1)P_0],$$

$$P_{1,2} = \frac{1}{x_2 - x_1}[(x - x_1)P_2 - (x - x_2)P_1],$$

$$P_{0,1,2} = \frac{1}{x_2 - x_0}\left[(x - x_0)P_{1,2} - (x - x_2)P_{0,1}\right],$$

and so on. They are generated in the manner shown in Table 3.3, where each row is completed before the succeeding rows are begun.

x_0	P_0				
x_1	P_1	$P_{0,1}$			
x_2	P_2	$P_{1,2}$	$P_{0,1,2}$		
x_3	P_3	$P_{2,3}$	$P_{1,2,3}$	$P_{0,1,2,3}$	
x_4	P_4	$P_{3,4}$	$P_{2,3,4}$	$P_{1,2,3,4}$	$P_{0,1,2,3,4}$

Table 3.3

The procedure that uses the result of Theorem 3.5 to recursively generate interpolating polynomial approximations is called **Neville's method**.

To avoid the multiple subscripts, we let $Q_{ij}(x)$, for $0 \le j \le i$, denote the interpolating polynomial of degree j on the $(j+1)$ numbers $x_{i-j}, x_{i-j+1}, \ldots, x_{i-1}, x_i$; that is,

$$Q_{i,j} = P_{i-j,i-j+1,\ldots,i-1,i}.$$

Using this notation provides the Q notation array in Table 3.4.

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

Table 3.4

Example2

Values of various interpolating polynomials at x = 1.5 were obtained in the Illustration at the beginning of the Section using the data shown in Table 3.5. Apply Neville′ s method to the data by constructing a recursive table of the form shown in Table 3.4.

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Table 3.5

Solution Let $x_0 = 1.0, x_1 = 1.3, x_2 = 1.6, x_3 = 1.9$, and $x_4 = 2.2$, then $Q_{0,0} = f(1.0)$,$Q_{1,0} = f(1.3)$, $Q_{2,0} = f(1.6)$, $Q_{3,0} = f(1.9)$, and $Q_{4,0} = f(2.2)$. Calculating the first-degree approximation $Q_{1,1}(1.5)$ gives

$$Q_{1,1}(1.5) = \frac{(x - x_0)Q_{1,0} - (x - x_1)Q_{0,0}}{x_1 - x_0}$$

$$= \frac{(1.5 - 1.0)Q_{1,0} - (1.5 - 1.3)Q_{0,0}}{1.3 - 1.0}$$

$$= \frac{0.5(0.6200860) - 0.2(0.7651977)}{0.3}$$

$$= 0.5233449.$$

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

$$Q_{2,1}(1.5) = \frac{(1.5-1.3)(0.4554022)-(1.5-1.6)(0.6200860)}{1.6-1.3} = 0.5102968,$$

$$Q_{3,1}(1.5) = 0.5132634,$$

$$Q_{4,1}(1.5) = 0.5104270$$

The best linear approximation is expected to be $Q_{2,1}$ because 1.5 is between $x_1 = 1.3$ and $x_2 = 1.6$

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

In a similar manner, approximations using higher-degree polynomials are given by

$$Q_{2,2}(1.5) = \frac{(1.5-1.0)(0.5102968)-(1.5-1.6)(0.5233449)}{1.6-1.0} = 0.5124715,$$

$$Q_{3,2}(1.5) = 0.5112857$$

$$Q_{4,2}(1.5) = 0.5137361.$$

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

1.0	0.7651977				
1.3	0.6200860	0.5233449			
1.6	0.4554022	0.5102968	0.5124715		
1.9	0.2818186	0.5132634	0.5112857	0.5118127	
2.2	0.1103623	0.5104270	0.5137361	0.5118302	0.5118200

If the latest approximation, $Q_{4,4}$, was not sufficiently accurate, another node, x_5, could be selected, and another row added to the table:

$$x_5 \quad Q_{5,0} \quad Q_{5,1} \quad Q_{5,2} \quad Q_{5,3} \quad Q_{5,4} \quad Q_{5,5}.$$

Then $Q_{4,4}, Q_{5,4}$, and $Q_{5,5}$ could be compared to determine further accuracy.

The function in Example 2 is the Bessel function of the first kind of order zero, whose value at 2.5 is -0.0483838, and the next row of approximations to $f(1.5)$ is

$$2.5 \quad -0.0483838 \quad 0.4807699 \quad 0.5301984 \quad 0.5119070 \quad 0.5118430 \quad 0.5118277.$$

The final new entry, 0.5118277, is correct to all seven decimal places.

Example 3

Table 3.7 lists the values of $f(x) = \ln x$ accurate to the places given. Use Neville's method and four-digit rounding arithmetic to approximate $f(2.1) = \ln 2.1$ by completing the Neville table.

i	x_i	$\ln x_i$
0	2.0	0.6931
1	2.2	0.7885
2	2.3	0.8329

Table 3.7

Solution $x - x_0 = 0.1, x - x_1 = -0.1, x - x_2 = -0.2$, and we are given $Q_{0,0} = 0.6931, Q_{1,0} = 0.7885$, and $Q_{2,0} = 0.8329$, we have

$$Q_{1,1} = \frac{1}{0.2}[(0.1)0.7885 - (-0.1)0.6931] = \frac{0.1482}{0.2} = 0.7410$$

and

$$Q_{2,1} = \frac{1}{0.1}[(-0.1)0.8329 - (-0.2)0.7885] = \frac{0.07441}{0.1} = 0.7441.$$

The final approximation we can obtain from this data is

$$Q_{2,2} = \frac{1}{0.3}[(0.1)0.7441 - (-0.2)0.7410] = \frac{0.2276}{0.3} = 0.7420.$$

These values are shown in Table 3.8.

i	x_i	$\ln x_i$
0	2.0	0.6931
1	2.2	0.7885
2	2.3	0.8329

i	x_i	$x - x_i$	Q_{i0}	Q_{i1}	Q_{i2}
0	2.0	0.1	0.6931		
1	2.2	−0.1	0.7885	0.7410	
2	2.3	−0.2	0.8329	0.7441	0.7420

Table 3.8

In the preceding example we have $f(2.1) = \ln 2.1 = 0.7419$ to four decimal places, so the absolute error is

$$|f(2.1) - P_2(2.1)| = |0.7419 - 0.7420| = 10^{-4}.$$

$$f(x) = P(x) + \frac{f^{n+1}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_n) \qquad (3.3)$$

However, $f'(x) = 1/x, f''(x) = -1/x^2$, and $f'''(x) = 2/x^3$, so the error bound

$$|f(2.1) - P_2(2.1)| = \left|\frac{f'''(\xi(2.1))}{3!}(x - x_0)(x - x_1)(x - x_2)\right|$$

$$= \left|\frac{1}{3(\xi(2.1))^3}(0.1)(-0.1)(-0.2)\right| \le \frac{0.002}{3(2)^3} = 8.\bar{3} \times 10^{-5}$$

Algorithm 3.1—Neville's Iterated Interpolation

To evaluate the interpolating polynomial P on the $n+1$ distinct numbers $x_0, \dots, x_n$ at the number x for the function f :

INPUT number $x, x_0, x_1, \dots, x_n$; values $f(x_0)$, $f(x_1)$,…, $f(x_n)$ as the first column $Q_{0,0}, Q_{1,0}, \dots, Q_{n,0}$ of Q.

OUTPUT the table Q with $P(x)$=$Q_{n,n}$.

Step 1 For i=1,2,…,n

for j=1,2,…,i

set $Q_{i,j} = \frac{(x-x_{i-j})Q_{i,j-1}-(x-x_i)Q_{i-1,j-1}}{x_i-x_{i-j}}$.

Step 2 OUTPUT(Q);

STOP.

The algorithm can be modified to allow for the addition of new interpolating nodes. For example, the inequality

$$\left|Q_{i,i} - Q_{i-1,i-1}\right| < \varepsilon$$

can be used as a stopping criterion, where ε is a prescribed error tolerance. If the inequality is true, $Q_{i,j}$ is a reasonable approximation to $f(x)$. If the inequality is false, a new interpolation point, x_{i+1}, is added.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 3-3

Divided Differences

Suppose that $P_n(x)$ is the nth Lagrange polynomial that agrees with the function f at the distinct numbers $x_0, x_1 \dots, x_n$. Although this polynomial is unique, there are alternate algebraic representations that are useful in certain situations. The divided differences of f with respect to $x_0, x_1 \dots, x_n$ are used to express $P_n(x)$ in the form

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}) \quad (3.5)$$

for appropriate constants $a_0, a_1, \dots, a_n$. Now we determine the constants.

$$a_0 = P_n(x_0) = f(x_0). \qquad f(x_0) + a_1(x_1 - x_0) = P_n(x_1) = f(x_1);$$

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (3.6)$$

We now introduce the **divided-difference(差商)** notation. The *zeroth divided difference* of the function f with respect to x_i, denoted $f[x_i]$, is simply the value of f at x_i:

$$f[x_i] = f(x_i). \qquad (3.7)$$

the *first divided difference* of f with respect to x_i and x_{i+1} is denoted $f[x_i, x_{i+1}]$ and defined as

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}. \qquad (3.8)$$

The *second divided difference*, $f[x_i, x_{i+1}, x_{i+2}]$, is defined as

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

The kth divided difference relative to $x_i, x_{i+1}, x_{i+2}, \dots, x_{i+k}$ is

$$f[x_i, x_{i+1}, \dots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \dots, x_{i+k}] - f[x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i}. \quad (3.9)$$

The process ends with the single *nth divided difference*,

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}) \quad (3.5)$$

Hence the interpolating polynomial in Eq. (3.5) is

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}),$$

$P_n(x)$ can be rewritten in a form called **Newton's Divided Difference**:

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, x_2, \ldots, x_k](x - x_0)\cdots(x - x_{k-1}) \quad (3.10)$$

The value of $f[x_0, x_1, \ldots, x_k]$ is independent of the order of the numbers x_0, $x_1, \ldots, x_k$.

$$R_n(x) = f[x, x_0, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_n)$$

The generation of the divided differences is outlined in Table 3.9. Two fourth and one fifth difference can also be determined from these data.

x	$f(x)$	First divided differences	Second divided differences	Third divided differences
x_0	$f[x_0]$			
		$f[x_0,x_1]=\dfrac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2]=\dfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$	
		$f[x_1,x_2]=\dfrac{f[x_2]-f[x_1]}{x_2-x_1}$		$f[x_0,x_1,x_2,x_3]=\dfrac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$
x_2	$f[x_2]$		$f[x_1,x_2,x_3]=\dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	
		$f[x_2,x_3]=\dfrac{f[x_3]-f[x_2]}{x_3-x_2}$		$f[x_1,x_2,x_3,x_4]=\dfrac{f[x_2,x_3,x_4]-f[x_1,x_2,x_3]}{x_4-x_1}$
x_3	$f[x_3]$		$f[x_2,x_3,x_4]=\dfrac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$	
		$f[x_3,x_4]=\dfrac{f[x_4]-f[x_3]}{x_4-x_3}$		$f[x_2,x_3,x_4,x_5]=\dfrac{f[x_3,x_4,x_5]-f[x_2,x_3,x_4]}{x_5-x_2}$
x_4	$f[x_4]$		$f[x_3,x_4,x_5]=\dfrac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}$	
		$f[x_4,x_5]=\dfrac{f[x_5]-f[x_4]}{x_5-x_4}$		
x_5	$f[x_5]$			

Table 3.9

ALGORITHM 3.2

Newton's Divided-Difference Formula

To obtain the divided-difference coefficients of the interpolatory polynomial P on the $n+1$ distinct numbers $x_0, x_1, \ldots, x_n$ for the function f :

INPUT number $x, x_0, x_1, \ldots, x_n$; values $f(x_0)$, $f(x_1)$,…, $f(x_n)$ as $F_{0,0}, F_{1,0}, \ldots, F_{n,0}$.

OUTPUT the number $F_{0,0}, F_{1,1}, \ldots, F_{n,n}$ where

$$P_n(x) = F_{0,0} + \sum_{i=1}^{n} F_{i,i} \prod_{j=0}^{i-1} (x - x_j). \quad (F_{i,i} \text{ is } f[x_0, x_1, \ldots, x_i].)$$

Step 1 For i=1,2,…,n

For j=1,2,…,

set $F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$. $(F_{i,j} = f[x_{i-j}, \dots, x_i].)$

Step 2 OUTPUT($F_{0,0}, F_{1,1}, \dots, F_{n,n}$);

STOP.

The form of the output in Algorithm 3.2 can be modified to produce all the divided differences, as shown in Example 1.

Example 1

Complete the divided difference table for the data used in Example 1 of Section 3.2, and reproduced in Table 3.10, and construct the interpolating polynomial that uses all this data.

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Table 3.10

Solution The first divided difference involving x_0 and x_1 is

$$f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} = \frac{0.6200860-0.7651977}{1.3-1.0} = -0.4837057$$

$$f[x_0, x_1, x_2,] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} = -0.1087339.$$

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0} = 0.0658784$$

$$f[x_0, x_1, x_2, x_3, x_4] = \frac{f[x_1,x_2,x_3,x_4]-f[x_0,x_1,x_2,x_3]}{x_4-x_0} = 0.0018251$$

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Table 3.11

i	x_i	$f[x_i]$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$	$f[x_{i-3}, \ldots, x_i]$	$f[x_{i-4}, \ldots, x_i]$
0	1.0	0.7651977				
			−0.4837057			
1	1.3	0.6200860		−0.1087339		
			−0.5489460		0.0658784	
2	1.6	0.4554022		−0.0494433		0.0018251
			−0.5786120		0.0680685	
3	1.9	0.2818186		0.0118183		
			−0.5715210			
4	2.2	0.1103623				

$$P_4(x) = 0.7651977 - 0.4837057(x - 1.0) - 0.1087339(x - 1.0)(x - 1.3)$$
$$+0.0658784(x - 1.0)(x - 1.3)(x - 1.6)$$
$$+0.0018251(x - 1.0)(x - 1.3)(x - 1.6)(x - 1.9).$$

Notice that the value $P_4(1.5) = 0.5118200$ agrees with the result in Table 3.6 for Example 2 of Section 3.2, as it must because the polynomials are the same.

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}. \quad (3.8)$$

The Mean Value Theorem 1.8 applied to Eq. (3.8) when $i = 0$,

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

implies that when f' exists, $f[x_0, x_1] = f'(\xi)$ for some number ξ between x_0 and x_1. The following theorem generalizes this result.

Theorem 3.6

Suppose that $f \in C^n[a,b]$ and $x_0, x_1, \dots, x_n$ are distinct numbers in $[a,b]$. Then a number ξ exists in (a,b) with

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

Proof Let

$$g(x) = f(x) - P_n(x).$$

Since $f(x_i) = P_n(x_i)$ for each $i = 0,1,\dots,n$, the function g has $n+1$ distinct zeros in $[a,b]$. Generalized Rolle's Theorem 1.10 implies that a number ξ in (a,b) exists with $g^{(n)}(\xi) = 0$, so

$$0 = f^{(n)}(\xi) - P_n^{(n)}(\xi)$$

Since $P_n(x)$ is a polynomial of degree n whose leading coefficient is $f[x_0, x_1, \dots, x_\mathrm{n}]$

$$P_n^{(n)}(x) = n!\, f[x_0, x_1, \dots, x_n],$$

for all values of x. As a consequence

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

Example

Given $f(1) = 1, f(2) = 4, f(3) = 7, f(4) = 8, f(5) = 6$, please write down the Newton interpolation polynomial of degree at most 3 and 4.

Example

Given $f(-1) = -2, f(0) = -1, f(1) = 0, f'(0) = 0$, please write down the polynomial $P_3(x)$ of degree 3 which satisfies

$$P_3(-1) = f(-1), P_3(0) = f(0), P_3(1) = f(1), P_3'(0) = f'(0)$$

Example

$f(x) = x^7 + x^4 + 3x + 1$, compute

$$f[2^0, 2^1, \cdots, 2^7] \text{ and } f[2^0, 2^1, \cdots, 2^7, 2^8]$$

Given $(x_{\mathrm{i}}, y_i)(\mathrm{i} = 0,1,\cdots,n)$,

$$P(x) = \sum_{i=0}^{n} y_i L_i(x)$$

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

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} w_{n+1}(x)$$

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, x_2, \dots, x_k](x - x_0)\cdots(x - x_{k-1}) \qquad (3.10)$$

Newton's divided-difference formula can be expressed in a simplified form when the nodes are arranged consecutively with equal spacing. We introduce $h = x_{i+1} - x_i$, $i = 0,1,\dots,n-1$ and let $x = x_0 + sh$. Then the difference $x - x_i$ is $x - x_i = (s - i)h$. So Eq. (3.10) becomes

$$\begin{aligned} P_n(x) &= P_n(x_0 + sh) \\ &= f[x_0] + shf[x_0, x_1] + s(s-1)h^2 f[x_0, x_1, x_2] + \cdots \\ &+ s(s-1)\cdots(s-n+1)h^n f[x_0, x_1, \dots, x_n] \\ &= f[x_0] + \sum_{k=1}^{n} s(s-1)\cdots(s-k+1)h^k f[x_0, x_1, \dots, x_k]. \end{aligned}$$

Using binomial-coefficient notation,

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!},$$

we can express $P_n(x)$ compactly as

$$P_n(x) = P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} k!\, h^k f[x_0, x_i, \ldots, x_k]. \quad (3.11)$$

Forward Differences

$$\Delta f(x_0) = f(x_1) - f(x_0) = f_1 - f_0$$

$$\Delta^2 f(x_0) = \Delta(\Delta f(x_0)) = \Delta(f_1 - f_0)$$

$$\cdots\cdots\cdots$$

Example Compute the forward difference table for the following data.

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0	0				
1	1				
2	8				
3	27				
4	64				

$Ef_k = f_{k+1}$(shift operator) $\quad If_k = f_k$(idendity operator)

$E^{-1}f_k = f_{k-1}$

Property

(1) $f_{k+n} = E^n f_k = (\Delta + I)^n f_k = \sum_{i=0}^{n} C_n^i \Delta^i f_k$

(2) $\Delta^n f_k = (E - I)^n f_k = \sum_{i=0}^{n} (-1)^i C_n^i f_{k+n-i}$

Forward Differences

The **Newton forward-difference formula**, is constructed by making use of the forward difference notation Δ introduced in Aitken's Δ^2 method. With this notation,

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}\big(f(x_1) - f(x_0)\big) = \frac{1}{h}\Delta f(x_0)$$

$$f[x_0, x_1, x_2] = \frac{1}{2h}\left[\frac{\Delta f(x_1) - \Delta f(x_0)}{h}\right] = \frac{1}{2h^2}\Delta^2 f(x_0),$$

$$P_n(x) = P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} k!\, h^k f[x_0, x_i, \dots, x_k]. \qquad (3.11)$$

and, in general,

$$f[x_0, x_1, \dots, x_k] = \frac{1}{k!\, h^k} \Delta^k f(x_0).$$

Since $f[x_0] = f(x_0)$,Eq. (3.11) has the following form.

Newton Forward-Difference Formula

$$P_n(x) = f(x_0) + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f(x_0) \qquad (3.12)$$

$$R_n(x) = \frac{s(s-1)\cdots(s-n)}{(n+1)!} h^{n+1} f^{(n+1)}(\xi)$$

Backward Differences

If the interpolating nodes are reordered from last to first as $x_n, x_{n-1}, \dots, x_0$ we can write the interpolatory formula as

$$P_n(x) = f[x_n] + f[x_n, x_{n-1}](x - x_n) + f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1}) + \cdots + f[x_n, \dots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_1).$$

If, in addition, the nodes are equally spaced with $x = x_n + sh$ and $x - x_i = (s + n - i)h$,then

$$P_n(x) = P_n(x_0 + sh)$$
$$= f[x_n] + shf[x_n, x_{n-1}] + s(s+1)h^2 f[x_n, x_{n-1}, x_{n-2}] + \cdots$$
$$+ s(s+1)\cdots(s+n-1)h^n f[x_n, \ldots, x_0].$$

This is used to derive a commonly applied formula known as the **Newton backward difference formula**. To discuss this formula, we need the following definition.

Definition 3.7

Given the sequence $\{p_n\}_{n=0}^{\infty}$,define the backward ∇p_n(read nabla p_n) by

$$\nabla p_n = p_n - p_{n-1}, \quad \text{for } n \geq 1.$$

Higher powers are defined recursively by

$$\nabla^k p_n = \nabla\left(\nabla^{k-1} p_n\right), \quad \text{for } n \geq 2.$$

Property

(1) $\nabla = I - E^{-1}$

(2) $\nabla^n f_k = (I - E^{-1})^n f_k = \sum_{i=0}^{n} (-1)^{n-i} C_n^i f_{k+i-n}$

Definition 3.7 implies that

$$f[x_n, x_{n-1}] = \frac{1}{h}\nabla f(x_n), f[x_n, x_{n-1}, x_{n-2}] = \frac{1}{2h^2}\nabla^2 f(x_n),$$

And, in general,

$$f[x_n, x_{n-1}, \dots, x_{n-k}] = \frac{1}{k!\,h^k}\nabla^k f(x_n).$$

Consequently

$$P_n(x) = f[x_n] + s\nabla f(x_n) + \frac{s(s+1)}{2}\nabla^2 f(x_n) + \cdots + \frac{s(s+1)\cdots(s+n-1)}{n!}\nabla^n f(x_n).$$

If we extend the binomial coefficient notation to include all real values of s by letting

$$\binom{-s}{k} = \frac{-s(-s-1)\cdots(-s-k+1)}{k!} = (-1)^k \frac{s(s+1)\cdots(s+k-1)}{k!},$$

Then

$$P_n(x) = f[x_n] + (-1)^1\binom{-s}{1}\nabla f(x_n) + (-1)^2\binom{-s}{2}\nabla^2 f(x_n) + \cdots + (-1)^n\binom{-s}{n}\nabla^n f(x_n).$$

This gives the following result.

Newton Backward–Difference Formula

$$P_n(x) = f[x_n] + \sum_{k=1}^{n}(-1)^k\binom{-s}{k}\nabla^k f(x_n)$$

Example

Given the following data for $f(x) = \sqrt{x}$, compute $f(1.01), f(1.28)$ by the Newton forward-difference and backward-difference formula with degree 3.

x_i	y_i
1.00	1.00000
1.05	1.02470
1.10	1.04881
1.15	1.07238
1.20	1.09544
1.25	1.11803
1.30	1.14017

Example

Given the following data for $f(x) = \sqrt{x}$, compute $f(1.01), f(1.28)$ by the Newton forward-difference and backward-difference formula with degree 3.

x_i	y_i
1.00	1.00000
1.05	1.02470
1.10	1.04881
1.15	1.07238
1.20	1.09544
1.25	1.11803
1.30	1.14017

Example

Given the following data for $f(x) = \sin x$, compute $\sin(0.57891)$ by the Newton forward-difference and backward-difference formula with degree 2.

x	$\sin x$	Δ	Δ^2	Δ^3
0.4	0.38942			
0.5	0.47943	0.09001		
0.6	0.56464	0.08521	0.00480	
0.7	0.64422	0.07958	−0.00563	−0.00083

Example

Given the following data for $f(x) = \sin x$, compute $\sin(0.57891)$ by the Newton forward-difference and backward-difference formula with degree 2.

x	$\sin x$	Δ	Δ^2	Δ^3
0.4	0.38942			
0.5	0.47943	0.09001		
0.6	0.56464	0.08521	0.00480	
0.7	0.64422	0.07958	−0.00563	−0.00083

Centered Differences

The Newton forward-difference and backward-difference formulas are not appropriate for approximating $f(x)$ when x lies near the center of the table because neither will permit the highest-order difference to have x_0 close to x. A number of divided-difference formulas are available for this case, each of which has situations when it can be used to maximum advantage. These methods are known as **centered-difference formulas**. We will consider only one centered-difference formula, Stirling's method.

For the centered-difference formulas, we choose x_0 near the point being approximated and label the nodes directly below x_0 as x_1, x_2, ... and those directly above as x_{-1}, x_{-2}, ... With this convention, **Stirling's formula** is given by

$$P_n(x) = P_{2m+1}(x) = f[x_0] + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + s^2h^2 f[x_{-1}, x_0, x_1]$$
$$+ \frac{s(s^2-1)h^3}{2} f[x_{-2}, x_{-1}, x_0, x_1] + f[x_{-1}, x_0, x_1, x_2]) + \cdots$$
$$+ s^2(s^2-1)(s^2-4)\cdots(s^2(m-1)^2)h^{2m} f[x_{-m}, \ldots, x_m]$$
$$+ \frac{s(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2}(f[x_{-m-1}, \ldots, x_m] + f[x_{-m}, \ldots, x_{m+1}]),$$

x	$f(x)$	First divided differences	Second divided differences	Third divided differences	Fourth divided differences
x_{-2}	$f[x_{-2}]$				
		$f[x_{-2}, x_{-1}]$			
x_{-1}	$f[x_{-1}]$		$f[x_{-2}, x_{-1}, x_0]$		
		$\underline{f[x_{-1}, x_0]}$		$\underline{f[x_{-2}, x_{-1}, x_0, x_1]}$	
x_0	$\underline{f[x_0]}$		$\underline{f[x_{-1}, x_0, x_1]}$		$\underline{f[x_{-2}, x_{-1}, x_0, x_1, x_2]}$
		$\underline{f[x_0, x_1]}$		$\underline{f[x_{-1}, x_0, x_1, x_2]}$	
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$			
x_2	$f[x_2]$				

if $n = 2m + 1$ is odd. If $n = 2m$ is even, we use the same formula but delete the last line. The entries used for this formula are underlined in Table 3.13.

Example 2 Consider the table of data given in the previous examples. Use Stirling's formula to approximate $f(1.5)$ with $x_0 = 1.6$.

Solution To apply Stirling's formula we use the underlined entries in the difference Table 3.14.

x	$f(x)$	First divided differences	Second divided differences	Third divided differences	Fourth divided differences
1.0	0.7651977				
		−0.4837057			
1.3	0.6200860		−0.1087339		
		−0.5489460		0.0658784	
1.6	0.4554022		−0.0494433		0.0018251
		−0.5786120		0.0680685	
1.9	0.2818186		0.0118183		
		−0.5715210			
2.2	0.1103623				

Table 3.14

The formula, with $h = 0.3, x_0 = 1.6$, and $s = -\frac{1}{3}$, becomes

$$f(1.5) \approx P_4\left(1.6 + \left(-\frac{1}{3}\right)(0.3)\right)$$

$$= 0.4554022 + \left(-\frac{1}{3}\right)\left(\frac{0.3}{2}\right)\big((-0.5489460) + (-0.5786120)\big)$$

$$+ \left(-\frac{1}{3}\right)^2 (0.3)^2(-0.0494433) + \frac{1}{2}\left(-\frac{1}{3}\right)\left(\left(-\frac{1}{3}\right)^2 - 1\right)(0.3)^3(0.0658784 + 0.0680685)$$

$$+ \left(-\frac{1}{3}\right)^2\left(\left(-\frac{1}{3}\right)^2 - 1\right)(0.3)^4(0.0018251) = 0.5118200.$$

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 3-4

Hermite Interpolation

埃尔米特插值

Definition 3.8

Let x_0 ,$x_1, \dots, x_n$ be $n+1$ distinct numbers in $[a,b]$ and for $i = 0, \dots, n$ Suppose that $f \in C^1\,[a,b]$, the **Hermite polynomial** approximating f is the polynomial $H(x)$ of least degree $2n+1$ such that

$$H(x_i) = f(x_i), H'(x_i) = f'(x_i), \qquad i = 0,1, \dots, n$$

For a given function f , these polynomials agree with f at $x_0, x_1, \dots, x_n$. In addition, since their first derivatives agree with those of f , they have the same "shape" as the function at$(x_i, f(x_i))$ in the sense that the tangent lines to the polynomial and the function agree.

Question

Suppose that $f \in C^1\,[a, b]$, it satisfies the following table,

x	x_0	x_1
$f(x)$	y_0	y_1
$f'(x)$	m_0	m_1

We can construct a polynomial with degree 3.

Existence and uniqueness

We consider the polynomial

$$H_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

$$H_3'(x) = a_1 + 2a_2 x + 3a_3 x^2$$

x	x_0	x_1
$f(x)$	y_0	y_1
$f'(x)$	m_0	m_1

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + a_3 x_0^3 = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 = y_1 \\ a_1 + 2a_2 x_0 + 3a_3 x_0^2 = m_0 \\ a_1 + 2a_2 x_1 + 3a_3 x_0^3 = m_1 \end{cases}$$

$$\begin{vmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 0 & 1 & 2x_0 & 3x_0^2 \\ 0 & 1 & 2x_1 & 3x_1^2 \end{vmatrix} = -(x_0 - x_1)^4 \neq 0$$

Existence and uniqueness

Existence and uniqueness

Theorem

If $f \in C^1[a,b]$ and $x_0, x_1 \in [a,b]$ are distinct, the unique polynomial of least degree agreeing with f and f at x_0, x_1 is the Hermite polynomial of degree at most 3 given by

$$H_3(x) = \sum_{j=0}^{1} f(x_j) H_{1,j}(x) + \sum_{j=0}^{1} f'(x_j) \widehat{H}_{1,j}(x)$$

where, for $L_{1,j}(x)$ denoting the jth Lagrange coefficient polynomial of degree 1, we have

$$H_{1,j}(x) = \left[1 - 2(x - x_j) L'_{1,j}(x_j)\right] L_{1,j}^2(x) \text{ and } \widehat{H}_{1,j}(x) = (x - x_j) L_{1,j}^2(x).$$

Moreover, if $f \in C^4[a,b]$, then

$$f(x) = H_3(x) + \frac{(x-x_0)^2(x-x_1)^2}{4!} f^{(4)}(\xi(x))$$

for some (generally unknown) $\xi(x)$ in the interval (a,b).

Theorem 3.9

If $f \in C^1[a,b]$ and $x_0, x_1, \dots, x_n \in [a,b]$ are distinct, the unique polynomial of least degree agreeing with f and f at $x_0, x_1, \dots, x_n$ is the Hermite polynomial of degree at most $2n+1$ given by

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \widehat{H}_{n,j}(x)$$

where, for $L_{n,j}(x)$ denoting the jth Lagrange coefficient polynomial of degree n, we have

$$H_{n,j}(x) = \left[1 - 2(x - x_j) L'_{n,j}(x_j)\right] L^2_{n,j}(x) \text{ and } \widehat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x).$$

Moreover, if $f \in C^{2n+2}[a,b]$, then

$$f(x) = H_{2n+1}(x) + \frac{(x - x_0)^2 \dots (x - x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi(x))$$

for some (generally unknown) $\xi(x)$ in the interval (a,b).

Proof First recall that

$$L_{n,j}(x_i) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$$

Hence when i ≠ j

$$H_{n,j}(x_i) = 0 \qquad and \qquad \widehat{H}_{n,j}(x_i) = 0$$

whereas, for each i,

$$H_{n,i}(x_i) = \left[1 - 2(x_i - x_i)L'_{n,i}(x_i)\right] \cdot 1 = 1 \; and \; \widehat{H}_{n,i}(x_i) = (x_i - x_i) \cdot 1^2 = 0$$

As a consequence

$$H_{2n+1}(x_i) = \sum_{\substack{j=0 \\ j\neq i}}^{n} f\left(x_j\right)\cdot 0 + f(x_i)\cdot 1 + \sum_{j=0}^{n} f\,'\left(x_j\right)\cdot 0 = f(x_i)$$

so H_{2n+1} agrees with f at $x_0, x_1, \dots, x_n$.

To show the agreement of H_{2n+1} with f' at the nodes, first note that $L_{n,j}(x)$ is a factor of $H'_{n,j}(x)$, so $H'_{n,j}(x_i) = 0$ when $i \neq j$. In addition, when $i = j$ we have $L_{n,j}(x_i) = 1$, so

$$H'_{n,i}(x_i) = -2L'_{n,i}(x_i)\cdot L^2_{n,i}(x_i) + \left[1 - 2(x_i - x_i)L'_{n,i}(x_i)\right]2L_{n,i}(x_i)L'_{n,i}(x_i)$$
$$= -2L'_{n,i}(x_i) + 2L'_{n,i}(x_i) = 0$$

Hence, $H'_{n,j}(x_i) = 0$ for all i and j.

Finally,

$$\widehat{H}'_{n,j}(x_i) = L^2_{n,j}(x_i) + (x_i - x_j)2L_{n,j}(x_i)L'_{n,j}(x_i)$$
$$= L_{n,j}(x_i)\left[L_{n,j}(x_i) + 2(x_i - x_j)L'_{n,j}(x_i)\right]$$

so $\widehat{H}'_{n,j}(x_i) = 0$ if $i \neq j$ and $\widehat{H}'_{n,i}(x_i) = 1$.Combining these facts, we have

$$H'_{2n+1}(x_i) = \sum_{j=0}^{n} f(x_j) \cdot 0 + \sum_{\substack{j=0 \\ j \neq i}}^{n} f'(x_j) \cdot 0 + f'(x_i) \cdot 1 = f'(x_i)$$

Therefore, H_{2n+1} agrees with f and H'_{2n+1} with f' at $x_0, x_1, \dots, x_n$.

Example 1

Use the Hermite polynomial that agrees with the data listed in Table 3.15 to find an approximation of $f(1.5)$.

k	x_k	$f(x_k)$	$f'(x_k)$
0	1.3	0.6200860	−0.5220232
1	1.6	0.4554022	−0.5698959
2	1.9	0.2818186	−0.5811571

Table 3.15

Solution We first compute the Lagrange polynomials and their derivatives. This gives

$$L_{2,0}(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9}, \qquad L'_{2,0}(x) = \frac{100}{9}x - \frac{175}{9};$$

$$L_{2,1}(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{-100}{9}x^2 + \frac{320}{9}x - \frac{247}{9}, \quad L'_{2,1}(x) = \frac{-200}{9}x + \frac{320}{9};$$

and

$$L_{2,2} = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{50}{9}x^2 - \frac{145}{9}x + \frac{104}{9}, \qquad L'_{2,2}(x) = \frac{100}{9}x - \frac{145}{9}.$$

k	x_k	$f(x_k)$	$f'(x_k)$
0	1.3	0.6200860	−0.5220232
1	1.6	0.4554022	−0.5698959
2	1.9	0.2818186	−0.5811571

The polynomials $H_{2,j}(x)$ and $\widehat{H}_{2,j}(x)$ are then

$$H_{2,0}(x) = [1 - 2(x - 1.3)(-5)]\left(\frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9}\right)^2 = (10x - 12)\left(\frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9}\right)^2,$$

$$H_{2,1}(x) = 1 \cdot \left(\frac{-100}{9}x^2 + \frac{320}{9}x - \frac{247}{9}\right)^2,$$

$$H_{2,2}(x) = 10(2 - x)\left(\frac{50}{9}x^2 - \frac{145}{9}x + \frac{104}{9}\right)^2,$$

$$\widehat{H}_{2,0}(x) = (x - 1.3)\left(\frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9}\right)^2,$$

$$\widehat{H}_{2,1}(x) = (x - 1.6)\left(\frac{-100}{9}x^2 + \frac{320}{9}x - \frac{247}{9}\right)^2,$$

$$\widehat{H}_{2,2}(x) = (x - 1.9)\left(\frac{50}{9}x^2 - \frac{145}{9}x + \frac{104}{9}\right)^2.$$

Finally

$$H_5(x) = 0.6200860H_{2,0}(x) + 0.4554022H_{2,1}(x) + 0.2818186H_{2,2}(x) - 0.5220232\widehat{H}_{2,0}(x) - 0.5698959\widehat{H}_{2,1}(x) - 0.5811571\widehat{H}_{2,2}(x)$$

and

$$H_5(1.5) = 0.6200860\left(\frac{4}{27}\right) + 0.4554022\left(\frac{64}{81}\right) + 0.2818186\left(\frac{5}{81}\right) - 0.5220232\left(\frac{4}{405}\right)$$
$$-0.5698959\left(\frac{-32}{405}\right) - 0.5811571\left(\frac{-2}{405}\right) = 0.5118277,$$

a result that is accurate to the places listed.

Although Theorem 3.9 provides a complete description of the Hermite polynomials, it is clear from Example 1 that the need to determine and evaluate the Lagrange polynomials and their derivatives makes the procedure tedious even for small values of n.

Hermite Polynomials Using Divided Differences

There is an alternative method for generating Hermite approximations that has as its basis the Newton interpolatory divided-difference formula (3.10) at $x_0, x_1, \dots, x_n$, that is

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \dots, x_k](x - x_0) \cdots (x - x_{k-1})$$

The alternative method uses the connection between the nth divided difference and the nth derivative of f, as outlined in Theorem 3.6 in Section 3.3.

Theorem 3.6 Suppose that $f \in C^n[a, b]$ and $x_0, x_1, \dots, x_n$ are distinct numbers in $[a, b]$. Then a number ξ exists in (a, b) with

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

Suppose that the distinct numbers $x_0, x_1, \dots, x_n$ are given together with the values of f and f' at these numbers. Define a new sequence $z_0, z_1, \dots, z_{2n+1}$ by

$$z_{2i} = z_{2i+1} = x_i, \qquad \text{for each} \quad i = 0,1, \dots, n,$$

and construct the divided difference table in the form of Table 3.9 that uses $z_0, z_1, \dots, z_{2n+1}$.

Since $z_{2i} = z_{2i+1} = x_i$ for each i, we cannot define $f[z_{2i}, z_{2i+1}]$ by the divided difference formula.

However, if we assume, based on Theorem 3.6, that the reasonable substitution in this situation is $f[z_{2i}, z_{2i+1}] = f'(z_{2i}) = f'(x_i)$,we can use the entries

$$f'(x_0), f'(x_1), \dots, f'(x_n)$$

in place of the undefined first divided differences

$$f[z_0, z_1], f[z_2, z_3], \dots, f[z_{2n}, z_{2n+1}].$$

The remaining divided differences are produced as usual, and the appropriate divided differences are employed in Newton's interpolatory divided-difference formula.

Table 3.16 shows the entries that are used for the first three divided-difference columns when determining the Hermite polynomial $H_5(x)$ for x_0, x_1, and x_2. The Hermite polynomial is then given by

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, z_k](x - z_0)(x - z_1)\cdots(x - z_{k-1})$$

z	$f(z)$	First divided differences	Second divided differences
$z_0 = x_0$	$f[z_0] = f(x_0)$		
		$f[z_0, z_1] = f'(x_0)$	
$z_1 = x_0$	$f[z_1] = f(x_0)$		$f[z_0, z_1, z_2] = \dfrac{f[z_1, z_2] - f[z_0, z_1]}{z_2 - z_0}$
		$f[z_1, z_2] = \dfrac{f[z_2] - f[z_1]}{z_2 - z_1}$	
$z_2 = x_1$	$f[z_2] = f(x_1)$		$f[z_1, z_2, z_3] = \dfrac{f[z_2, z_3] - f[z_1, z_2]}{z_3 - z_1}$
		$f[z_2, z_3] = f'(x_1)$	
$z_3 = x_1$	$f[z_3] = f(x_1)$		$f[z_2, z_3, z_4] = \dfrac{f[z_3, z_4] - f[z_2, z_3]}{z_4 - z_2}$
		$f[z_3, z_4] = \dfrac{f[z_4] - f[z_3]}{z_4 - z_3}$	
$z_4 = x_2$	$f[z_4] = f(x_2)$		$f[z_3, z_4, z_5] = \dfrac{f[z_4, z_5] - f[z_3, z_4]}{z_5 - z_3}$
		$f[z_4, z_5] = f'(x_2)$	
$z_5 = x_2$	$f[z_5] = f(x_2)$		

Table 3.16

Example 2

Use the data given in Example 1 and the divided difference method to determine the Hermite polynomial approximation at $x = 1.5$.

Solution the second entry in the third column we use the second 1.3 entry in the second column and the first 1.6 entry in that column to obtain

1.3	0.6200860					
		−0.5220232				
1.3	0.6200860		−0.0897427			
		−0.5489460		0.0663657		
1.6	0.4554022		−0.0698330		0.0026663	
		−0.5698959		0.0679655		−0.0027738
1.6	0.4554022		−0.0290537		0.0010020	
		−0.5786120		0.0685667		
1.9	0.2818186		−0.0084837			
		−0.5811571				
1.9	0.2818186					

$$\frac{0.4554022 - 0.6200860}{1.6 - 1.3} = -0.5489460$$

For the first entry in the fourth column we use the first 1.3 entry in the third column and the first 1.6 entry in that column to obtain

$$\frac{-0.5489460-(-0.5220232)}{1.6-1.3}=-0.0897427$$

The value of the Hermite polynomial at 1.5 is

$$\begin{aligned}
&H_5(1.5)\\
&=f[1.3]+f'(1.3)(1.5-1.3)+f[1.3,1.3,1.6](1.5-1.3)^2\\
&+f[1.3,1.3,1.6,1.6](1.5-1.3)^2(1.5-1.6)\\
&+f[1.3,1.3,1.6,1.6,1.9](1.5-1.3)^2(1.5-1.6)^2\\
&+f[1.3,1.3,1.6,1.6,1.9,1.9](1.5-1.3)^2(1.5-1.6)^2(1.5-1.9)\\
&=0.6200860+(-0.5220232)(0.2)+(-0.0897427)(0.2)^2\\
&+0.0663657(0.2)^2(-0.1)+0.0026663(0.2)^2(-0.1)^2\\
&+(-0.0027738)(0.2)^2(-0.1)^2(-0.4)=0.5118277.
\end{aligned}$$

ALGORITHM 3.3

Hermite Interpoaltion

To obtain the coefficients of the Hermite interpolating polynomial $H(x)$ on the $(n+1)$ distinct numbers $x_0, \dots, x_n$ for the function f :

INPUT numbers $x_0, x_1, \dots, x_n$; values $f(x_0), \dots, f(x_n)$ and $f'(x_0), \dots, f'(x_n)$.

OUTPUT the numbers $Q_{0,0}, Q_{1,1}, \dots, Q_{2n+1,2n+1}$ where

$$H(x) = Q_{0,0} + Q_{1,1}(x - x_0) + Q_{2,2}(x - x_0)^2 + Q_{3,3}(x - x_0)^2(x - x_1)$$

$$+Q_{4,4}(x - x_0)^2(x - x_1)^2 + \cdots$$

$$+Q_{2n+1,2n+1}(x - x_0)^2(x - x_1)^2 \cdots (x - x_{n-1})^2(x - x_n).$$

Step 1 For $i = 0,1,\dots,n$ do Steps 2 and 3.

Step 2 Set $z_{2i} = x_i$

$z_{2i+1} = x_i;$

$Q_{2i,0} = f(x_i);$

$Q_{2i+1,0} = f(x_i);$

$Q_{2i+1,1} = f'(x_i);$

Step 3 If $i \neq 0$ then set

$$Q_{2i,1} = \frac{Q_{2i,0} - Q_{2i-1,0}}{z_{2i} - z_{2i-1}}.$$

Step 4 For $i = 2,3,\dots,2n+1$

for $j = 2,3,\dots,i$ set $Q_{i,j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}.$

Step 5 **OUTPUT** $(Q_{0,0}, Q_{1,1}, \dots, Q_{2n+1,2n+1});$

STOP

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 3-5

Cubic Spline Interpolation

三次样条插值

In the previous sections a single polynomial approximate the arbitrary functions on closed intervals. However, high-degree polynomials can oscillate erratically, that is, a minor fluctuation over a small portion of the interval can induce large fluctuations over the entire range.

$$f(x) = \frac{1}{1 + 25x^2}$$

An alternative approach is to divide the approximation interval into a collection of subintervals and construct a (generally) different approximating polynomial on each subinterval. This is called **piecewise-polynomial approximation**.

Piecewise linear interpolation $f(x) \in C[a,b], a = x_0 < x_1 < \cdots < x_n = b$

x	x_0	x_1	$\cdots$	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	$\cdots$	$f(x_n)$

Definition (piecewise linear interpolation) If $I(x)$ satisfies

(1) $I(x) \in C[a,b]$

(2) $I(x_k) = y_k$

(3) In every subinterval $[x_k, x_{k+1}], I(x)$ is a linear polynomial.

We call it a piecewise linear interpolating polynomial.

$$I(x) = \sum_{j=0}^{n} y_j l_j(x)$$

$$l_j(x) = \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}}, x_{j-1} \le x \le x_j & (j \ne 0) \\ \frac{x-x_{j+1}}{x_j-x_{j+1}}, x_j \le x \le x_{j+1} & (j \ne n) \\ 0, & \text{others} \end{cases}$$

Theorem If $f(x) \in C^2[a,b], M_2 = \max_{a \le x \le b} |f'(x)|$, then for any $x \in [a,b]$, the error of piecewise linear polynomial approximation is

$$|f(x) - I(x)| \le \frac{M_2}{8} h^2.$$

Piecewise-Polynomial Approximation

The simplest piecewise-polynomial approximation is **piecewise-linear** interpolation, which consists of joining a set of data points

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))\}$$

by a series of straight lines, as shown in Figure 3.7.

Figure 3.7

A disadvantage of linear function approximation is that there is likely no differentiability at the endpoints of the subintervals, which, in a geometrical context, means that the interpolating function is not "smooth." Often it is clear from physical conditions that smoothness is required, so the approximating function must be continuously differentiable.

An alternative procedure is to use a piecewise polynomial of Hermite type. For example, if the values of f and of f' are known at each of the points $x_0 < x_1 < \cdots < x_n$, a cubic Hermite polynomial can be used on each of the subintervals $[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]$ to obtain a function that has a continuous derivative on the interval $[x_0, x_n]$.

Piecewise Hermite cubic interpolation

$f(x) \in C[a,b], a = x_0 < x_1 < \cdots < x_n = b$

x	x_0	x_1	$\cdots$	x_n
$f(x)$	y_0	y_1	$\cdots$	y_n
$f'(x)$	m_0	m_1	$\cdots$	m_n

Definition (piecewise Hermite cubic interpolation) If $I(x)$ satisfies

(1) $I(x) \in C^1[a,b]$

(2) $I(x_k) = y_k, I'(x_k) = m_k$

(3) In every subinterval $[x_k, x_{k+1}], I(x)$ is a cubic polynomial.

We call it a piecewise Hermite cubic interpolating polynomial.

$$I(x) = \sum_{j=0}^{n}[y_j\alpha_j(x) + m_j\beta_j(x)]$$

$$\alpha_j(x) = \begin{cases} \left(\frac{x-x_{j-1}}{x_j-x_{j-1}}\right)^2 (1+2\frac{x-x_j}{x_{j-1}-x_j}), x_{j-1} \le x \le x_j & (j \ne 0) \\ \left(\frac{x-x_{j+1}}{x_j-x_{j+1}}\right)^2 (1+2\frac{x-x_j}{x_{j+1}-x_j}), x_j \le x \le x_{j+1} & (j \ne n) \\ 0, & \text{others} \end{cases}$$

$$\beta_j(x) = \begin{cases} \left(\frac{x-x_{j-1}}{x_j-x_{j-1}}\right)^2 (x-x_j), x_{j-1} \le x \le x_j & (j \ne 0) \\ \left(\frac{x-x_{j+1}}{x_j-x_{j+1}}\right)^2 (x-x_j), x_j \le x \le x_{j+1} & (j \ne n) \\ 0, & \text{others} \end{cases}$$

Theorem If $f(x) \in C^4[a,b]$, and the functional and derivative table is $(x_i, y_i, m_i)(0 = 0,1,\cdots,n)$, then for any $x \in [a,b]$, the error of piecewise Hermite cubic polynomial approximation is

$$|f(x) - I(x)| \leq \frac{h^4}{384} \max_{a \leq x \leq b} f^{(4)}(x), x \in [a,b].$$

To determine the appropriate Hermite cubic polynomial on a given interval is simply a matter of computing $H_3(x)$ for that interval. The Lagrange interpolating polynomials needed to determine H_3 are of first degree, so this can be accomplished without great difficulty. However, to use Hermite piecewise polynomials for general interpolation, we need to know the derivative of the function being approximated, and this is frequently unavailable.

The remainder of this section considers approximation using piecewise polynomials that require no specific derivative information, except perhaps at the endpoints of the interval on which the function is being approximated.

1.5
1
0.5
-1
-0.5
0
0.5
1
x

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
-1
-0.5
0
0.5
1
x

Cubic Splines

The most common piecewise-polynomial approximation uses cubic polynomials between each successive pair of nodes and is called **cubic spline interpolation**. A general cubic polynomial involves four constants, so there is sufficient flexibility in the cubic spline procedure to ensure that the interpolant is not only continuously differentiable on the interval, but also has a continuous second derivative. The construction of the cubic spline does not, however, assume that the derivatives of the interpolant agree with those of the function it is approximating, even at the nodes. (See Figure 3.8.)

$S(x)$
S_0
S_1
S_j
S_{j+1}
S_{n-2}
S_{n-1}
$S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_{j+1})$
$S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$
$S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$
x_0 x_1 x_2 . . . x_j x_{j+1} x_{j+2} . . . x_{n-2} x_{n-1} x_n x

Figure 3.8

Definition 3.10

Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a **cubic spline interpolant** S for f is a function that satisfies the following conditions:

(a) $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0,1, \dots, n-1$;

(b) $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0,1, \dots, n-1$;

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0,1, \dots, n-2$; (*Implied by* (**b**).)

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0,1, \dots, n-2$

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0,1, \dots, n-2$;

(f) One of the following sets of boundary conditions is satisfied:

(i) $S''(x_0) = S''(x_n) = 0$ (**natural** (*or free*) **boundary**);

(ii) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (**clamped boundary**).

Although cubic splines are defined with other boundary conditions, the conditions given in (**f**) are sufficient for our purposes. When the free boundary conditions occur, the spline is called a **natural spline**, and its graph approximates the shape that a long flexible rod would assume if forced to go through the data points $\{(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))\}$.

In general, clamped boundary conditions lead to more accurate approximations because they include more information about the function. However, for this type of boundary condition to hold, it is necessary to have either the values of the derivative at the endpoints or an accurate approximation to those values.

Example 1

Construct a natural cubic spline that passes through the points (1,2),(2,3),and (3,5).

Solution This spline consists of two cubics. The first for the interval [1,2], denoted

$$S_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3,$$

and the other for [2,3], denoted

$$S_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3.$$

There are 8 constants to be determined, which requires 8 conditions. Four conditions come from the fact that the splines must agree with the data at the nodes. Hence

$$2 = f(1) = a_0, 3 = f(2) = a_0 + b_0 + c_0 + d_0, 3 = f(2) = a_1, \text{and}$$
$$5 = f(3) = a_1 + b_1 + c_1 + d_1$$

Two more come from the fact that $S_0'(2) = S_1'(2)$ and $S_0''(2) = S_1''(2)$. These are

$S_0'(2) = S_1'(2)$: $\quad b_0 + 2c_0 + 3d_0 = b_1$ and $S_0''(2) = S_1''(2)$: $2c_0 + 6d_0 = 2c_1$

The final two come from the natural boundary conditions:

$$S_0''(1) = 0: \qquad 2c_0 = 0 \qquad \text{and} \qquad S_1''(3) = 0: \qquad 2c_1 + 6d_1 = 0.$$

Solving this system of equations gives the spline

$$S(x) = \begin{cases} 2 + \frac{3}{4}(x-1) + \frac{1}{4}(x-1)^3, \text{for } x \in [1,2] \\ 3 + \frac{3}{2}(x-2) + \frac{3}{4}(x-2)^2 - \frac{1}{4}(x-2)^3, \text{for } x \in [2,3] \end{cases}$$

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0,1,\dots,n-2$;

Construction of a Cubic Spline

A spline defined on an interval that is divided into n subintervals will require determining $4n$ constants. To construct the cubic spline interpolant for a given function f, the conditions in the definition are applied to the cubic polynomials

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

for each $j = 0,1,\dots,n-1$. Since $S_j(x_j) = a_j = f(x_j)$, condition (c) can be applied to obtain

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3,$$

for each $j = 0,1,\dots,n-2$.

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}),\ j = 0,1,\dots,n-2$

Let $h_j = x_{j+1} - x_j$, for each $j = 0,1,\dots,n-1$. If we also define $a_n = f(x_n)$, then the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3, \quad j = 0,1,\cdots,n-1 \quad (3.15)$$

In a similar manner, define $b_n = S'(x_n)$ and observe that

$$S'_j(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

implies $S_j(x_j) = b_j$, for each $j = 0,1,\dots,n-1$. Applying condition (**d**) gives

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2 \quad (3.16)$$

for each $j = 0,1,\dots,n-1$.

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1}),\ j = 0,1,\dots,n-2$

Another relationship between the coefficients of S_j is obtained by defining $c_n = S''(x_n)/2$ and applying condition (**e**). Then, for each $j = 0,1,\dots,n-1$,

$$c_{j+1} = c_j + 3d_j h_j. \tag{3.17}$$

Solving for d_j in Eq. (3.17) and substituting this value into Eq. (3.15) and (3.16) gives, for each $j = 0,1,\dots,n-1$, the new equations

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}\left(2c_j + c_{j+1}\right) \tag{3.18}$$

and

$$b_{j+1} = b_j + h_j\left(c_j + c_{j+1}\right). \tag{3.19}$$

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of equation (3.18), first for b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \qquad (3.20)$$

Substituting these values into the equation derived from Eq. (3.19), with the index reduced by one, gives the linear system of equations

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

for each $j = 1,2,\dots,n-1$.

(3.21)

$$c_{j+1} = c_j + 3d_j h_j. \qquad (3.17)$$

This system involves only the $\{c_j\}_{j=0}^{n}$ as unknowns. The values of $\{h_j\}_{j=0}^{n-1}$ and $\{a_j\}_{j=0}^{n}$ are given. So once the values of $\{c_j\}_{j=0}^{n}$ are determined, it is a simple matter to find the remainder of the constants $\{b_j\}_{j=0}^{n-1}$ from Eq. (3.20) and $\{d_j\}_{j=0}^{n-1}$ from Eq. (3.17). Then we can construct the cubic polynomials$\{S_j(x)\}_{j=0}^{n-1}$.

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \qquad (3.21)$$

The major question that arises in connection with this construction is whether the values of $\{c_j\}_{j=0}^{n}$ can be found using the system of equations given in (3.21) and, if so, whether these values are unique. The following theorems indicate that this is the case when either of the boundary conditions given in part (**f**) of the definition are imposed.

Natural Splines

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \qquad (3.21)$$

Theorem 3.11

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$, then f has a unique natural spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is, a spline interpolant that satisfies the natural boundary conditions $S''(a) = 0$ and $S''(b) = 0$.

Proof The boundary conditions in this case imply that $c_n = S''(x_n)/2 = 0$ and that

$$0 = S''(x_0) = 2c_0 + 6d_0(x_0 - x_0)$$

so $c_0 = 0$. The two equations $c_0 = 0$ and $c_n = 0$ together with the equations in (3.21) produce a linear system described by the vector equation $\boldsymbol{Ax} = \boldsymbol{b}$, where A is the $(n+1) \times (n+1)$ matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix},$$

and **b** and **x** are the vectors

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

The matrix A is **strictly diagonally dominant**, that is, in each row the magnitude of the diagonal entry exceeds the sum of the magnitudes of all the other entries in the row. A linear system with a matrix of this form will be shown by Theorem 6.21 in Section 6.6 to have a unique solution for $c_0, c_1, \dots, c_n$.

The solution to the cubic spline problem with the boundary conditions $S''(x_0) = S''(x_n) = 0$ can be obtained by applying Algorithm 3.4.

ALGORITHM

Natural Cubic Spline

To construct the cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

INPUT $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0,1, \ldots, n-1$.

(*Note:* $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ *for* $x_j \leq x \leq x_{j+1}$.)

Step 1 For $i = 0,1,\dots,n-1$ set $h_i = x_{i+1} - x_i$

Step 2 For $i = 1,2,\dots,n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

Step 3 Set $l_0 = 1$; *(Steps 3, 4, 5, and part of Step 6 solve a tridiagonal linear system using a method described in Algorithm 6.7.)*

$\mu_0 = 0;$

$z_0 = 0.$

Step 4 For $i = 1,2,\dots,n-1$

set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1};$

$\mu_i = h_i / l_i;$

$z_i = (\alpha_i - h_{i-1}z_{i-1})/l_i.$

Step 5 Set $l_n = 1$;

$z_n = 0$;

$c_n = 0$.

Step 6 For $j = n-1, n-2, \ldots, 0$

set $c_j = z_j - \mu_j c_{j+1}$;

$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;

$d_j = (c_{j+1} - c_j)/(3h_j)$.

Step 7 **OUTPUT** $(a_j, b_j, c_j, d_j$ for $j = 0,1 \ldots, n-1)$;

STOP.

Example 2

At the beginning of Chapter 3 we gave some Taylor polynomials to approximate the exponential $f(x) = e^x$. Use the data points $(0,1), (1, e)$, $(2, e^2)$, and $(3, e^3)$ to form a natural spline $S(x)$ that approximates $f(x) = e^x$.

Solution We have $n = 3, h_0 = h_1 = h_2 = 1, a_0 = 1, a_1 = e, a_2 = e^2$, and $a_3 = e^3$. So the matrix A and the vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ given in Theorem 3.11 have the forms

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 0 \\ 3(e^2 - 2e + 1) \\ 3(e^3 - 2e^2 + e) \\ 0 \end{bmatrix}, \text{and } \boldsymbol{x} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

The vector-matrix equation $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to the system of equations

$$c_0 = 0,$$
$$c_0 + 4c_1 + c_2 = 3(e^2 - 2e + 1),$$
$$c_1 + 4c_2 + c_3 = 3(e^3 - 2e^2 + e),$$
$$c_3 = 0.$$

This system has the solution $c_0 = c_3 = 0$, and to 5 decimal places,

$$c_1 = \frac{1}{5}(-e^3 + 6e^2 - 9e + 4) \approx 0.75685,$$

and $c_2 = \frac{1}{5}(4e^3 - 9e^2 + 6e - 1) \approx 5.83007$

Solving for the remaining constants gives

$$b_0 = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(c_1 + 2c_0)$$

$$= (e - 1) - \frac{1}{15}(-e^3 + 6e^2 - 9e + 4) \approx 1.46600$$

$$b_1 = \frac{1}{h_1}(a_2 - a_1) - \frac{h_1}{3}(c_2 + 2c_1)$$

$$= (e^2 - e) - \frac{1}{15}(2e^3 + 3e^2 - 12e + 7) \approx 2.22285$$

$$b_2 = \frac{1}{h_2}(a_3 - a_2) - \frac{h_2}{3}(c_3 + 2c_2)$$

$$= (e^3 - e^2) - \frac{1}{15}(8e^3 - 18e^2 + 12e - 2) \approx 8.80977$$

$$d_0 = \frac{1}{3h_0}(c_1 - c_0) = \frac{1}{15}(-e^3 + 6e^2 - 9e + 4) \approx 0.25228$$

$$d_1 = \frac{1}{3h_1}(c_2 - c_1) = \frac{1}{3}(e^3 - 3e^2 + 3e - 1) \approx 1.69107$$

and

$$d_2 = \frac{1}{3h_2}(c_3 - c_1) = \frac{1}{15}(-4e^3 + 9e^2 - 6e + 1) \approx -1.94336$$

The natural cubic spine is described piecewise by

$$S(x) = \begin{cases} 1 + 1.46600x + 0.25228x^3, & \text{for} x \in [0,1], \\ 2.71828 + 2.22285(x-1) + 0.75685(x-1)^2 + 1.69107(x-1)^3, & \text{for} x \in [1,2], \\ 7.38906 + 8.80977(x-2) + 5.83007(x-2)^2 - 1.94336(x-2)^3, & \text{for} x \in [2,3]. \end{cases}$$

The spline and its agreement with $f(x) = e^x$ are shown in Figure 3.10.

Figure 3.10

Illustration

To approximate the integral of $f(x) = e^x$ on [0,3], which has the value

$$\int_0^3 e^x\,dx = e^3 - 1 \approx 20.08553692 - 1 = 19.08553692,$$

we can piecewise integrate the spline that approximates f on this integral. This gives

$$\begin{aligned}\int_0^3 S(x) = &\int_0^1 1 + 1.46600x + 0.25228x^3 dx \\ &+ \int_1^2 2.71828 + 2.22285(x-1) + 0.75685(x-1)^2 + 1.69107(x-1)^3 dx \\ &+ \int_2^3 7.38906 + 8.80977(x-2) + 5.83007(x-2)^2 - 1.94336(x-2)^3 dx\end{aligned}$$

Integrating and collecting values from like powers gives

$$\int_0^3 S(x) = \left[x + 1.46600\frac{x^2}{2} + 0.25228\frac{x^4}{4}\right]_0^1$$

$$+\left[2.71828(x-1) + 2.22285\frac{(x-1)^2}{2} + 0.75685\frac{(x-1)^3}{3} + 1.69107\frac{(x-1)^4}{4}\right]_1^2$$

$$+\left[7.38906(x-2) + 8.80977\frac{(x-2)^2}{2} + 5.83007\frac{(x-2)^3}{3} - 1.94336\frac{(x-2)^4}{4}\right]_2^3$$

$$= (1 + 2.71828 + 7.38906) + \frac{1}{2}(1.46600 + 2.22285 + 8.80977)$$

$$+\frac{1}{3}(0.75685 + 5.83007) + \frac{1}{4}(0.25228 + 1.69107 - 1.94336) = 19.55229.$$

Because the nodes are equally spaced in this example the integral approximation is simply

$$\int_0^3 S(x)\,dx = (a_0 + a_1 + a_2) + \frac{1}{2}(b_0 + b_1 + b_2) + \frac{1}{3}(c_0 + c_1 + c_2) + \frac{1}{4}(d_0 + d_1 + d_2). \quad (3.22)$$

If we create the natural spline using Maple as described after Example 2, we can then use Maple's integration command to find the value in the Illustration. Simply enter

$int(sn(t), t = 0..3)$

19.55228648

Clamped Splines

Example 3

In Example 1 we found a natural spline S that passes through the points $(1,2), (2,3)$, and $(3,5)$. Construct a clamped spline s through these points that has $s'(1) = 2$ and $s'(3) = 1$

Solution Let

$$s_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3,$$

be the cubic on $[1,2]$ and the cubic on $[2,3]$ be

$$s_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3.$$

Then most of the conditions to determine the 8 constants are the same as those in Example 1.

That is,

$$2 = f(1) = a_0, 3 = f(2) = a_0 + b_0 + c_0 + d_0, 3 = f(2) = a_1,$$
$$\text{and } 5 = f(3) = a_1 + b_1 + c_1 + d_1.$$

$s_0'(2) = s_1'(2)$: $b_0 + 2c_0 + 3d_0 = b_1$ and $s_0''(2) = s_1''(2)$: $2c_0 + 6d_0 = 2c_1$

However, the boundary conditions are now

$$s_0'(1) = 2: \quad b_0 = 2 \quad \text{and} \quad s_1'(3) = 1: \quad b_1 + 2c_1 + 3d_1 = 1.$$

Solving this system of equations gives the spline as

$$s(x) = \begin{cases} 2 + 2(x-1) - \frac{5}{2}(x-1)^2 + \frac{3}{2}(x-1)^3, \text{for } x \in [1,2] \\ 3 + \frac{3}{2}(x-2) + 2(x-2)^2 - \frac{3}{2}(x-2)^3, \text{for } x \in [2,3] \end{cases}$$

In the case of general clamped boundary conditions we have a result that is similar to the theorem for natural boundary conditions described in Theorem 3.11.

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \qquad (3.20)$$

Theorem 3.12

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$ and differentiable at a and b, then f has a unique clamped spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is, a spline interpolant that satisfies the clamped boundary conditions $S'(a) = f'(a)$ and $S'(b) = f'(b)$.

Proof Since $f'(a) = S'(a) = S'(x_0) = b_0$, Eq. (3.20) with $j = 0$ implies

$$f'(a) = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1).$$

Consequently,

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a).$$

Similarly,

$$f'(b) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n),$$

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \quad (3.21)$$

so Eq. (3.20) with $j = n - 1$ implies that

$$f'(b) = \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) + h_{n-1}(c_{n-1} + c_n)$$
$$= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n),$$

And

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}).$$

Equations (3.21) together with the equations

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a)$$

And

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$$

determine the linear system **Ax = b,** where

$$A = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}, \text{ and } \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

This matrix A is also strictly diagonally dominant, therefore, the linear system has a unique solution for $c_0, c_1, \dots, c_n$.

The solution to the cubic spline problem with the boundary conditions $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ can be obtained by applying Algorithm 3.5.

ALGORITHM 3.5

Clamped Cubic Spline

To construct the cubic spline interpolant S for the function f defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$:

INPUT $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n); FPO = f'(x_0); FPN = f'(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0,1, \ldots, n-1$.

(*Note*: $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ *for* $x_j \le x \le x_{j+1}$)

Step 1 For $i = 0,1,\ldots,n-1$ set $h_i = x_{i+1} - x_i$.

Step 2 Set $\alpha_0 = 3(a_1 - a_0)/h_0 - 3FPO$;

$$\alpha_n = 3FPN - 3(a_n - a_{n-1})/h_{n-1}.$$

Step 3 For $i = 1,2,\ldots,n-1$

$$set\ \alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

Step 4 Set $l_0 = 2h_0$; (*Steps* 4,5,6, *and part of Step 7 solve a tridiagonal linear system using a method described in Algorithm 6.7.*)

$$\mu_0 = 0.5;$$

$$z_0 = \alpha_0/l_0.$$

Step 5 For $i = 1,2,\dots,n-1$

set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;

$$\mu_i = \frac{h_i}{l_i};$$

$$z_i = (\alpha_i - h_{i-1}z_{i-1})/l_i.$$

Step 6 Set $l_n = h_{n-1}(2 - \mu_{n-1})$;

$$z_n = (\alpha_n - h_{n-1}z_{n-1})/l_n;$$

$$c_n = z_n.$$

Step 7 For $j = n-1, n-2, \dots, 0$

set $c_j = z_j - \mu_j c_{j+1}$;

$$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3;$$

$$d_j = (c_{j+1} - c_j)/(3h_j).$$

Step 8 OUTPUT $(a_j, b_j, c_j, d_j \text{for} j = 0,1 \dots, n-1)$;

STOP.

Example 4

Example 2 used a natural spline and the data points $(0,1), (1, e), (2, e^2)$, and $(3, e^3)$ to form a new approximating function $S(x)$. Determine the clamped spline $s(x)$ that uses this data and the additional information that, since $f'(x) = e^x$, so $f'(0) = 1$ and $f'(3) = e^3$.

Solution As in Example 2, we have $n = 3, h_0 = h_1 = h_2 = 1, a_0 = 0, a_1 = e, a_2 = e^2$, and $a_3 = e^3$. This together with the information that $f'(0) = 1$ and $f'(3) = e^3$ gives the the matrix A and the vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ with the forms

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3(e-2) \\ 3(e^2-2e+1) \\ 3(e^3-2e^2+e) \\ 3e^2 \end{bmatrix}, \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

The vector-matrix equation $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to the system of equations

$$2c_0 + c_1 = 3(e-2),$$
$$c_0 + 4c_1 + c_2 = 3(e^2 - 2e + 1),$$
$$c_1 + 4c_2 + c_3 = 3(e^3 - 2e^2 + e),$$
$$c_2 + 2c_3 = 3e^2$$

Solving this system simultaneously for c_0, c_1, c_2 and c_3 gives, to 5 decimal places,

$$c_0 = \frac{1}{15}(2e^3 - 12e^2 + 42e - 59) = 0.44468,$$

$$c_1 = \frac{1}{15}(-4e^3 + 24e^2 - 39e + 28) = 1.26548,$$

$$c_2 = \frac{1}{15}(14e^3 - 39e^2 + 24e - 8) = 3.35087,$$

$$c_3 = \frac{1}{15}(-7e^3 + 42e^2 - 12e + 4) = 9.40815$$

Solving for the remaining constants in the same manner as Example 2 gives

$$b_0 = 1.00000, b_1 = 2.71016, b_2 = 7.32652,$$

and

$$d_0 = 0.27360, d_1 = 0.69513, d_2 = 2.01909.$$

This gives the clamped cubic spine

$$s(x) = \begin{cases} 1 + x + 0.44468x^2 + 0.27360x^3, & \text{if } 0 \le x < 1, \\ 2.71828 + 2.71016(x-1) + 1.26548(x-1)^2 + 0.69513(x-1)^3, & \text{if } 1 \le x < 2, \\ 7.38906 + 7.32652(x-2) + 3.35087(x-2)^2 + 2.01909(x-2)^3, & \text{if } 2 \le x \le 3. \end{cases}$$

The graph of the clamped spline and $f(x) = e^x$ are so similar that no difference can be seen.

We can also approximate the integral of f on [0,3], by integrating the clamped spline. The exact value of the integral is

$$\int_0^3 e^x\,dx = e^3 - 1 \approx 20.08554 - 1 = 19.08554.$$

Because the data is equally spaced, piecewise integrating the clamped spline results in the same formula as in (3.22), that is,

$$\int_0^3 s(x)\,dx = (a_0 + a_1 + a_2) + \frac{1}{2}(b_0 + b_1 + b_2) + \frac{1}{3}(c_0 + c_1 + c_2) + \frac{1}{4}(d_0 + d_1 + d_2).$$

Hence the integral approximation is

$$\int_0^3 s(x)\,dx = (1 + 2.71828 + 7.38906) + \frac{1}{2}(1 + 2.71016 + 7.32652)$$
$$+ \frac{1}{3}(0.44468 + 1.26548 + 3.35087) + \frac{1}{4}(0.27360 + 0.69513 + 2.01909) = 19.05965.$$

The absolute error in the integral approximation using the clamped and natural splines are

$$\text{Natural: } |19.08554 - 19.55229| = 0.46675$$

and

$$\text{Clamped : } |19.08554 - 19.05965| = 0.02589$$

For integration purposes the clamped spline is vastly superior. This should be no surprise since the boundary conditions for the clamped spline are exact, whereas for the natural spline we are essentially assuming that, since $f''(x) = e^x$,

$$0 = S''(0) \approx f''(0) = e^1 = 1 \text{ and } 0 = S''(3) \approx f''(3) = e^3 \approx 20.$$

Illustration

Figure 3.11 shows a ruddy duck in flight. To approximate the top profile of the duck, we have chosen points along the curve through which we want the approximating curve to pass. Table 3.18 lists the coordinates of 21 data points relative to the superimposed coordinate system shown in Figure 3.12. Notice that more points are used when the curve is changing rapidly than when it is changing more slowly.

Figure 3.11

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

Table 3.18

Figure 3.12

Using Algorithm 3.4 to generate the natural cubic spline for this data produces the coefficients shown in Table 3.19. This spline curve is nearly identical to the profile, as shown in Figure 3.13.

j	x_j	a_j	b_j	c_j	d_j
0	0.9	1.3	5.40	0.00	−0.25
1	1.3	1.5	0.42	−0.30	0.95
2	1.9	1.85	1.09	1.41	−2.96
3	2.1	2.1	1.29	−0.37	−0.45
4	2.6	2.6	0.59	−1.04	0.45
5	3.0	2.7	−0.02	−0.50	0.17
6	3.9	2.4	−0.50	−0.03	0.08
7	4.4	2.15	−0.48	0.08	1.31
8	4.7	2.05	−0.07	1.27	−1.58
9	5.0	2.1	0.26	−0.16	0.04
10	6.0	2.25	0.08	−0.03	0.00
11	7.0	2.3	0.01	−0.04	−0.02
12	8.0	2.25	−0.14	−0.11	0.02
13	9.2	1.95	−0.34	−0.05	−0.01
14	10.5	1.4	−0.53	−0.10	−0.02
15	11.3	0.9	−0.73	−0.15	1.21
16	11.6	0.7	−0.49	0.94	−0.84
17	12.0	0.6	−0.14	−0.06	0.04
18	12.6	0.5	−0.18	0.00	−0.45
19	13.0	0.4	−0.39	−0.54	0.60
20	13.3	0.25			

Table 3.19

Figure 3.13

For comparison purposes, Figure 3.14 gives an illustration of the curve that is generated using a Lagrange interpolating polynomial to fit the data given in Table 3.18. The interpolating polynomial in this case is of degree 20 and oscillates wildly. It produces a very strange illustration of the back of a duck, in flight or otherwise.

Figure 3.14

To use a clamped spline to approximate this curve we would need derivative approximations for the endpoints. Even if these approximations were available, we could expect little improvement because of the close agreement of the natural cubic spline to the curve of the top profile.

Constructing a cubic spline to approximate the lower profile of the ruddy duck would be more difficult since the curve for this portion cannot be expressed as a function of x, and at certain points the curve does not appear to be smooth. These problems can be resolved by using separate splines to represent various portions of the curve, but a more effective approach to approximating curves of this type is considered in the next section.

The clamped boundary conditions are generally preferred when approximating functions by cubic splines, so the derivative of the function must be known or approximated at the endpoints of the interval. When the nodes are equally spaced near both end-points, approximations can be obtained by any of the appropriate formulas given in Sections 4.1 and 4.2. When the nodes are unequally spaced, the problem is considerably more difficult.

To conclude this section, we list an error-bound formula for the cubic spline with clamped boundary conditions. The proof of this result can be found in [Schul], pp. 57–58.

Theorem 3.13

If $f \in C^4[a,b]$ with $\max_{a \le x \le b} |f^{(4)}(x)| = M$. S is the unique clamped cubic spline interpolant to f with respect to the nodes $a = x_0 < x_1 < \cdots < x_n = b$, then for all x in $[a,b]$,

$$|f(x) - S(x)| \le \frac{5M}{384} \max_{0 \le j \le n-1} \left(x_{j+1} - x_j\right)^4.$$

The natural boundary conditions will generally given less accurate results than the clamped conditions near the ends of the interval $[x_0, x_n]$ unless the functions f happens to nearly satisfy $f''(x_0) = f''(x_n) = 0$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4

Numerical Differentiation and Integration

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4-1

Numerical Differentiation

To approximate $f'(x_0)$, suppose first that $x_0 \in (a,b)$, where $f \in C^2[a,b]$, and that $x_1 = x_0 + h$ for some $h \neq 0$ that is sufficiently small to ensure that $x_1 \in [a,b]$. We construct the first Lagrange polynomial $P_{0,1}(x)$ for f determined by x_0 and x_1, with its error term:

$$f(x) = P_{0,1}(x) + \frac{(x-x_0)(x-x_1)}{2!} f''(\xi(x))$$

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

$$+ \frac{(x-x_0)(x-x_0-h)}{2} f''(\xi(x)),$$

for some $\xi(x)$ between x_0 and x_1. Differentiating gives

$$f'(x) = \frac{f(x_0+h)-f(x_0)}{h} + D_x\left[\frac{(x-x_0)(x-x_0-h)}{2} f''(\xi(x))\right]$$

$$= \frac{f(x_0+h)-f(x_0)}{h} + \frac{2(x-x_0)-h}{2} f''(\xi(x))$$

$$+ \frac{(x-x_0)(x-x_0-h)}{2} D_x\left(f''(\xi(x))\right).$$

Deleting the terms involving $\xi(x)$ gives

$$f'(x) \approx \frac{f(x_0+h)-f(x_0)}{h}.$$

One difficulty with this formula is that we have no information about $D_x f''(\xi(x))$, so the truncation error cannot be estimated. When x is x_0, however, the coefficient of $D_x f''(\xi(x))$ is 0 , and the formula simplifies to

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''(\xi). \qquad (4.1)$$

For small values of h, the difference quotient $[f(x_0 + h) - f(x_0)]/h$ can be used to approximate $f'(x_0)$ with an error bounded by $M|h|/2$, where M is a bound on $|f''(x)|$ for x between x_0 and $x_0 + h$. This formula is known as the **forward-difference formula** if $h > 0$ (see Figure 4.1) and the **backward-difference formula** if $h < 0$.

Example 1

Use the forward-difference formula to approximate the derivative of $f(x) = \ln x$ at $x_0 = 1.8$ using $h = 0.1, h = 0.05$, and $h = 0.01$, and determine bounds for the approximation errors.

Figure 4.1

Solution The forward-difference formula

$$\frac{f(1.8+h)-f(1.8)}{h}$$

with $h = 0.1$ gives

$$\frac{\ln 1.9 - \ln 1.8}{0.1} = \frac{0.64185389 - 0.58778667}{0.1}$$

$$= 0.5406722$$

Because $f''(x) = -1/x^2$ and $1.8 < \xi < 1.9$, a bound for this approximation error is

$$\frac{|hf''(\xi)|}{2} = \frac{|h|}{2\xi^2} < \frac{0.1}{2(1.8)^2} = 0.0154321$$

The approximation and error bounds when $h = 0.05$ and $h = 0.01$ are found in a similar manner and the results are shown in Table 4.1.

h	$f(1.8+h)$	$\frac{f(1.8+h)-f(1.8)}{h}$	$\frac{\lvert h\rvert}{2(1.8)^2}$
0.1	0.64185389	0.5406722	0.0154321
0.05	0.61518564	0.5479795	0.0077160
0.01	0.59332685	0.5540180	0.0015432

Table 4.1

Since $f'(x) = 1/x$, the exact value of $f'(1.8)$ is $0.55\overline{5}$, and in this case the error bounds are quite close to the true approximation error.

To obtain general derivative approximation formulas, suppose that $\{x_0, x_1, \dots, x_n\}$ are $(n+1)$ distinct numbers in some interval I and that $f \in C^{n+1}(I)$.We have

$$f(x) = \sum_{k=0}^{n} f(x_k) L_k(x) + \frac{(x - x_0) \cdots (x - x_n)}{(n+1)!} f^{(n+1)}(\xi(x))$$

for some $\xi(x)$ in I, where $L_k(x)$ denotes the kth Lagrange coefficient polynomial for f at $x_0, x_1, \dots, x_n$.

Differentiating this expression gives

$$f'(x) = \sum_{k=0}^{n} f(x_k)L_k'(x) + D_x\left[\frac{(x-x_0)\cdots(x-x_n)}{(n+1!)}\right]f^{(n+1)}(\xi(x))$$
$$+\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}D_x\left[f^{(n+1)}(\xi(x))\right].$$

When $x = x_j$, $D_x\left[f^{(n+1)}(\xi(x))\right] = 0$, and the formula becomes

$$f'(x_j) = \sum_{k=0}^{n} f(x_k)L_k'(x) + \frac{f^{(n+1)}\left(\xi(x_j)\right)}{(n+1)!}\prod_{\substack{k=0\\k\neq j}}^{n}(x_j - x_k), \qquad (4.2)$$

which is called an **$(n+1)$-point formula** to approximate $f'(x_j)$.

The most common formulas are those involving three and five evaluation points.

Three-Point Formulas

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \qquad L_0'(x) = \frac{2x-x_1-x_2}{(x_0-x_1)(x_0-x_2)}.$$

Similarly,

$$L_1'(x) = \frac{2x-x_0-x_2}{(x_1-x_0)(x_1-x_2)} \qquad L_2'(x) = \frac{2x-x_0-x_1}{(x_2-x_0)(x_2-x_1)}.$$

$$f'(x_j) = f(x_0)\left[\frac{2x_j-x_1-x_2}{(x_0-x_1)(x_0-x_2)}\right] + f(x_1)\left[\frac{2x_j-x_0-x_2}{(x_1-x_0)(x_1-x_2)}\right]$$
$$+f(x_2)\left[\frac{2x_j-x_0-x_1}{(x_2-x_0)(x_2-x_1)}\right] + \frac{1}{6}f^{(3)}(\xi_j)\prod_{\substack{k=0\\k\neq j}}^{2}(x_j-x_k) \quad (4.3)$$

$j = 0,1,2$, where the notation ξ_j indicates that this point depends on x_j.

$$f'(x_j) = f(x_0)\left[\frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)}\right] + f(x_1)\left[\frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)}\right] + f(x_2)\left[\frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)}\right] + \frac{1}{6}f^{(3)}(\xi_j)\prod_{\substack{k=0\\k\neq j}}^{2}(x_j - x_k) \qquad (4.3)$$

Three-Point Formulas

We will assume equally-spaced nodes throughout the remainder of this section.

$$x_1 = x_0 + h \quad \text{and} \quad x_2 = x_0 + 2h, \qquad \text{for some } h \neq 0.$$

Using Eq. (4.3) with $x_j = x_0, x_1 = x_0 + h$, and $x_2 = x_0 + 2h$ gives

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_1) - \frac{1}{2}f(x_2)\right] + \frac{h^2}{3}f^{(3)}(\xi_0).$$

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0 + h) - \frac{1}{2}f(x_0 + 2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0),$$

$$f'(x_j) = f(x_0)\left[\frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)}\right] + f(x_1)\left[\frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)}\right] + f(x_2)\left[\frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)}\right] + \frac{1}{6}f^{(3)}(\xi_j)\prod_{\substack{k=0\\k\neq j}}^{2}(x_j - x_k)$$

$$f'(x_1) = \frac{1}{h}\left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_2)\right] - \frac{h^2}{6}f^{(3)}(\xi_1),$$

$$f'(x_0 + h) = \frac{1}{h}\left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_0 + 2h)\right] - \frac{h^2}{6}f^{(3)}(\xi_1),$$

$$f'(x_2) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_1) + \frac{3}{2}f(x_2)\right] + \frac{h^2}{3}f^{(3)}(\xi_2).$$

$$f'(x_0 + 2h) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_0 + h) + \frac{3}{2}f(x_0 + 2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_2).$$

As a matter of convenience, the variable substitution x_0 for $x_0 + h$(or $x_0 + 2h$) is used in the middle equation to change this formula to an approximation for $f'(x_0)$.

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0+h) - \frac{1}{2}f(x_0+2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0),$$

$$f'(x_0+h) = \frac{1}{h}\left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_0+2h)\right] - \frac{h^2}{6}f^{(3)}(\xi_1),$$

$$f'(x_0+2h) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_0+h) + \frac{3}{2}f(x_0+2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_2).$$

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0+h) - \frac{1}{2}f(x_0+2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0),$$

$$f'(x_0) = \frac{1}{2h}[-f(x_0-h) + f(x_0+h)] - \frac{h^2}{6}f^{(3)}(\xi_1)$$

$$f'(x_0) = \frac{1}{2h}[f(x_0-2h) - 4f(x_0-h) + 3f(x_0)] + \frac{h^2}{3}f^{(3)}(\xi_2).$$

Finally, note that the last of these equations can be obtained from the first by simply replacing h with $-h$, so there are actually only two formulas:

Three-Point Endpoint Formula

- $$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3} f^{(3)}(\xi_0) \qquad (4.4)$$

where ξ_0 lies between x_0 and $x_0 + 2h$.

Three-Point Midpoint Formula

- $$f'(x_0) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6} f^{(3)}(\xi_1) \qquad (4.5)$$

where ξ_1 lies between $x_0 - h$ and $x_0 + h$.

$$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3}f^{(3)}(\xi_0) \quad (4.4)$$

$$f'(x_0) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \quad (4.5)$$

Although the errors in both Eq. (4.4) and Eq. (4.5) are $O(h^2)$, the error in Eq. (4.5) is approximately half the error in Eq. (4.4). This is because Eq. (4.5) uses data on both sides of x_0 and Eq. (4.4) uses data on only one side. Note also that f needs to be evaluated at only two points in Eq. (4.5), whereas in Eq. (4.4) three evaluations are needed. The approximation in Eq. (4.4) is useful near the ends of an interval, because information about f outside the interval may not be available.

Figure 4.2

Five-Point Formulas

Five-Point Midpoint Formula

$$f'(x_0) = \frac{1}{12h}[f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)] + \frac{h^4}{30} f^{(5)}(\xi) \quad (4.6)$$

where ξ lies between $x_0 - 2h$ and $x_0 + 2h$.

The derivation of this formula is considered in Section 4.2.

Five-Point Endpoint Formula

$$f'(x_0) = \frac{1}{12h}[-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) + 16f(x_0 + 3h) - 3f(x_0 + 4h)] + \frac{h^4}{5} f^{(5)}(\xi) \quad (4.7)$$

where ξ lies between x_0 and $x_0 + 4h$.

Left-endpoint approximations are found using this formula with $h > 0$ and right-endpoint approximations with $h < 0$. The five-point endpoint formula is particularly useful for the clamped cubic spline interpolation of Section 3.5.

$$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3}f^{(3)}(\xi_0) \quad (4.4)$$

$$f'(x_0) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \quad (4.5)$$

Example 2

Values for $f(x) = xe^x$ are given in Table 4.2. Use all the applicable three-point and five-point formulas to approximate $f'(2.0)$.

Solution The data in the table permit us to find four different three-point approximations. We can use the endpoint formula (4.4) with $h = 0.1$ or with $h = -0.1$, and we can use the midpoint formula (4.5) with $h = 0.1$ or with $h = 0.2$.

Using the endpoint formula (4.4) with $h = 0.1$ gives

$$\frac{1}{0.2}[-3f(2.0) + 4f(2.1) - f(2.2]$$

$$= 5[-3(14.778112) + 4(17.148957) - 19.855030)] = 22.032310,$$

and with $h = -0.1$ gives 22.054525.

x	$f(x)$
1.8	10.889365
1.9	12.703199
2.0	14.778112
2.1	17.148957
2.2	19.855030

Table 4.2

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \quad (4.5)$$

$$f'(x_0) = \frac{1}{12h}[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)] + \frac{h^4}{30}f^{(5)}(\xi) \quad (4.6)$$

Using the midpoint formula (4.5) with $h = 0.1$ gives

$$\frac{1}{0.2}[f(2.1) - f(1.9)] = 5(17.148957 - 12.7703199) = 22.228790$$

and with $h = 0.2$ gives 22.414163

The only five-point formula for which the table gives sufficient data is the midpoint formula (4.6) with $h = 0.1$. This gives

$$\frac{1}{1.2}[f(1.8) - 8f(1.9) + 8f(2.1) - f(2.2)]$$

$$= \frac{1}{1.2}[10.889365 - 8(12.703199) + 8(17.148957) - 19.855030] = 22.166999$$

Table 4.2

x	$f(x)$
1.8	10.889365
1.9	12.703199
2.0	14.778112
2.1	17.148957
2.2	19.855030

If we had no other information we would accept the five-point midpoint approximation using $h = 0.1$ as the most accurate, and expect the true value to be between that approximation and the three-point mid-point approximation that is in the interval [22.166,22.229].

Three-point endpoint with $h = 0.1$: 1.35×10^{-1};

Three-point endpoint with $h = -0.1$: 1.13×10^{-1};

Three-point midpoint with $h = 0.1$: -6.16×10^{-2};

Three-point midpoint with $h = 0.2$: -2.47×10^{-1};

Five-point midpoint with $h = 0.1$: 1.69×10^{-4}.

Methods can also be derived to find approximations to higher derivatives of a function using only tabulated values of the function at various points. The derivation is algebraically tedious, however, so only a representative procedure will be presented.

Expand a function f in a third Taylor polynomial about a point x_0 and evaluate at $x_0 + h$ and $x_0 - h$. Then

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_1)h^4$$

and

$$f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_{-1})h^4,$$

where $x_0 - h < \xi_{-1} < x_0 < \xi_1 < x_0 + h$.

If we add these equations, the terms involving $f'(x_0)$ and $-f'(x_0)$ cancel, so

$$\begin{aligned} &f(x_0 + h) + f(x_0 - h) \\ &= 2f(x_0) + f''(x_0)h^2 + \frac{1}{24}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right]h^4. \end{aligned}$$

Solving this equation for $f''(x_0)$ gives

$$f''(x_0) = \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{24}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right]$$

Suppose $f^{(4)}$ is continuous on $[x_0 - h, x_0 + h]$. Since $\frac{1}{2}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right]$ is between $f^{(4)}(\xi_1)$ and $f^{(4)}(\xi_{-1})$, the Intermediate Value Theorem implies that a number ξ exists between ξ_1 and ξ_{-1}, and hence in $(x_0 - h, x_0 + h)$, with

$$f^{(4)}(\xi) = \frac{1}{2}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right].$$

This permits us to rewrite Eq. (4.8) in its final form.

Second Derivative Midpoint Formula

$$f''(x_0) = \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{12} f^{(4)}(\xi), \quad (4.9)$$

for some ξ, where $x_0 - h < \xi < x_0 + h$.

If $f^{(4)}$ is continuous on $[x_0 - h, x_0 + h]$ it is also bounded, and the approximation is $O(h^2)$.

Example 3

In Example 2 we used the data shown in Table 4.3 to approximate the first derivative of $f(x) = xe^x$ at $x = 2.0$. Use the second derivative formula (4.9) to approximate $f''(2.0)$.

x	$f(x)$
1.8	10.889365
1.9	12.703199
2.0	14.778112
2.1	17.148957
2.2	19.855030

Table 4.3

$$f''(x_0) = \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{12}f^{(4)}(\xi), \quad (4.9)$$

Solution The data permits us to determine two approximations for $f''(2.0)$. Using (4.9) with $h = 0.1$ gives

$$\frac{1}{0.01}[f(1.9) - 2f(2.0) + f(2.1)] = 100[12.703199 - 2(14.778112) + 17.148957] = 29.593200$$

and using (4.9) with $h = 0.2$ gives

$$\frac{1}{0.04}[f(1.8) - 2f(2.0) + f(2.2)] = 25[10.889365 - 2(14.778112) + 19.855030] = 29.704275$$

Because $f''(x) = (x + 2)e^x$, the exact value is $f''(2.0) = 29.556224$. Hence the actual errors are -3.70×10^{-2} and -1.48×10^{-1}, respectively.

x	$f(x)$
1.8	10.889365
1.9	12.703199
2.0	14.778112
2.1	17.148957
2.2	19.855030

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \qquad (4.5)$$

Round-Off Error Instability

It is particularly important to pay attention to round-off error when approximating derivatives. To illustrate the situation, let us examine the three-point midpoint formula Eq. (4.5),

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f^{(3)}(\xi_1)$$

more closely. Suppose that in evaluating $f(x_0+h)$ and $f(x_0-h)$ we encounter round-off errors $e(x_0+h)$ and $e(x_0-h)$. Then our computations actually use the values $\tilde{f}(x_0+h)$ and $\tilde{f}(x_0-h)$, which are related to the true values $f(x_0+h)$ and $f(x_0-h)$ by

$$f(x_0+h) = \tilde{f}(x_0+h) + e(x_0+h) \text{ and } f(x_0-h) = \tilde{f}(x_0-h) + e(x_0-h)$$

The total error in the approximation,

$$f'(x_0) - \frac{\tilde{f}(x_0+h) - \tilde{f}(x_0-h)}{2h} = \underbrace{\boxed{\frac{e(x_0+h) - e(x_0-h)}{2h}}}_{\text{round-off error}} \underbrace{\boxed{- \frac{h^2}{6} f^{(3)}(\xi_1)}}_{\text{truncation error}},$$

is due both to round-off error, the first part, and to truncation error. If we assume that the round-off errors $e(x_0 \pm h)$ are bounded by some number $\varepsilon > 0$ and that the third derivative of f is bounded by a number $M > 0$, then

$$\left| f'(x_0) - \frac{\tilde{f}(x_0+h) - \tilde{f}(x_0-h)}{2h} \right| \leq \frac{\varepsilon}{h} + \frac{h^2}{6} M.$$

To reduce the truncation error, $h^2 M/6$, we need to reduce h. But as h is reduced, the roundoff error ε/h grows. In practice, then, it is seldom advantageous to let h be too small, because in that case the round-off error will dominate the calculations.

Illustration

Consider using the values in Table 4.4 to approximate $f'(0.900)$, where $f(x) = \sin x$. The true value is $\cos 0.900 = 0.62161$. The formula

$$f'(0.900) \approx \frac{f(0.900+h) - f(0.900-h)}{2h},$$

with different values of h, gives the approximations in Table 4.5.

x	$\sin x$	x	$\sin x$
0.800	0.71736	0.901	0.78395
0.850	0.75128	0.902	0.78457
0.880	0.77074	0.905	0.78643
0.890	0.77707	0.910	0.78950
0.895	0.78021	0.920	0.79560
0.898	0.78208	0.950	0.81342
0.899	0.78270	1.000	0.84147

Table 4.4

h	Approximation to $f'(0.900)$	Error
0.001	0.62500	0.00339
0.002	0.62250	0.00089
0.005	0.62200	0.00039
0.010	0.62150	−0.00011
0.020	0.62150	−0.00011
0.050	0.62140	−0.00021
0.100	0.62055	−0.00106

Table 4.5

The optimal choice for h appears to lie between 0.005 and 0.05. We can verify that a minimum for

$$e(h) = \frac{\varepsilon}{h} + \frac{h^2}{6}M,$$

occurs at $h = \sqrt[3]{3\varepsilon/M}$, where

$$M = \max_{x\in[0.800,1.00]} |f'''(x)| = \max_{x\in[0.800,1.00]} |\cos x| = \cos 0.8 \approx 0.69671.$$

Because values of f are given to five decimal places, we will assume that the round-off error is bounded by $\varepsilon = 5 \times 10^{-6}$. Therefore, the optimal choice of h is approximately

$$h = \sqrt[3]{\frac{3(0.000005)}{0.69671}} \approx 0.028,$$

which is consistent with the results in Table 4.5.

h	Approximation to $f'(0.900)$	Error
0.001	0.62500	0.00339
0.002	0.62250	0.00089
0.005	0.62200	0.00039
0.010	0.62150	−0.00011
0.020	0.62150	−0.00011
0.050	0.62140	−0.00021
0.100	0.62055	−0.00106

Table 4.5

In practice, we cannot compute an optimal h to use in approximating the derivative, since we have no knowledge of the third derivative of the function. But we must remain aware that reducing the step size will not always improve the approximation.

We have considered only the round-off error problems that are presented by the three-point formula Eq. (4.5), but similar difficulties occur with all the differentiation formulas. The reason can be traced to the need to divide by a power of h. As we found in Section 1.2 (see, in particular, Example 3), division by small numbers tends to exaggerate round-off error, and this operation should be avoided if possible.

In the case of numerical differentiation, we cannot avoid the problem entirely, although the higher-order methods reduce the difficulty.

As approximation methods, numerical differentiation is unstable, since the small values of h needed to reduce truncation error also cause the round-off error to grow. This is the first class of unstable methods we have encountered, and these techniques would be avoided if it were possible. However, in addition to being used for computational purposes, the formulas are needed for approximating the solutions of ordinary and partial-differential equations.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4-2

Richardsons Extrapolation

理查森外推

Richardson's extrapolation is used to generate high-accuracy results while using low-order formulas.

Extrapolation can be applied whenever it is known that an approximation technique has an error term with a predictable form, one that depends on a parameter, usually the step size h. Suppose that for each number $h \neq 0$ we have a formula $N_1(h)$ that approximates an unknown constant M, and that the truncation error involved with the approximation has the form

$$M - N_1(h) = K_1 h + K_2 h^2 + K_3 h^3 + \cdots$$

for some collection of (unknown) constants $K_1, K_2, K_3, \ldots$

The truncation error is $O(h)$, so unless there was a large variation in magnitude among the constants $K_1, K_2, K_3, \ldots$,

$$M - N_1(0.1) \approx 0.1K_1, M - N_1(0.01) \approx 0.01K_1,$$

and, in general, $M - N_1(h) \approx K_1 h$

The object of extrapolation is to find an easy way to combine these rather inaccurate $O(h)$ approximations in an appropriate way to produce formulas with a higher-order truncation error.

Suppose, for example, we can combine the $N_1(h)$ formulas to produce an $O(h^2)$ approximation formula, $N_2(h)$, for M with

$$M - N_2(h) = \widehat{K}_2 h^2 + \widehat{K}_3 h^3 + \cdots,$$

for some, again unknown, collection of constants $\widehat{K}_2, \widehat{K}_3, \ldots$. Then we would have

$$M - N_2(0.1) \approx 0.01\widehat{K}_2, M - N_2(0.01) \approx 0.0001\widehat{K}_2,$$

and so on.

If the constants K_1 and $\widehat{K}_2$ are roughly of the same magnitude, then the $N_2(h)$ approximations would be much better than the corresponding $N_1(h)$ approximations. The extrapolation continues by combining the $N_2(h)$ approximations in a manner that produces formulas with $O(h^3)$ truncation error, and so on.

To see specifically how we can generate the extrapolation formulas, consider the $O(h)$ formula for approximating M

$$M = N_1(h) + K_1 h + K_2 h^2 + K_3 h^3 + \cdots. \quad (4.10)$$

The formula is assumed to hold for all positive h, so we replace the parameter h by half its value. Then we have a second $O(h)$ approximation formula

$$M = N_1\left(\frac{h}{2}\right) + K_1\frac{h}{2} + K_2\frac{h^2}{4} + K_3\frac{h^3}{8} + \cdots \quad (4.11)$$

$$M = N_1(h) + K_1 h + K_2 h^2 + K_3 h^3 + \cdots. \quad (4.10)$$

$$M = N_1\left(\frac{h}{2}\right) + K_1 \frac{h}{2} + K_2 \frac{h^2}{4} + K_3 \frac{h^3}{8} + \cdots \quad (4.11)$$

Subtracting Eq. (4.10) from twice Eq. (4.11) eliminates the term involving K_1 and gives

$$M = N_1\left(\frac{h}{2}\right) + \left[N_1\left(\frac{h}{2}\right) - N_1(h)\right] + K_2\left(\frac{h^2}{2} - h^2\right) + K_3\left(\frac{h^3}{4} - h^3\right) + \cdots. \quad (4.12)$$

Define

$$N_2(h) = N_1\left(\frac{h}{2}\right) + \left[N_1\left(\frac{h}{2}\right) - N_1(h)\right].$$

Then Eq. (4.12) is an $O(h^2)$ approximation formula for M :

$$M = N_2(h) - \frac{K_2}{2} h^2 - \frac{3K_3}{4} h^3 - \cdots. \quad (4.13)$$

Example 1

In Example 1 of Section 4.1 we use the forward-difference method with $h = 0.1$ and $h = 0.05$ to find approximations to $f'(1.8)$ for $f(x) = \ln(x)$. Assume that this formula has truncation error $O(h)$ and use extrapolation on these values to see if this results in a better approximation.

Solution In Example 1 of Section 4.1 we found that

with $h = 0.1$: $f'(1.8) \approx 0.5406722$, and with $h = 0.05$: $f'(1.8) \approx 0.5479795$

This implies that

$$N_1(0.1) = 0.5406722 \text{ and } N_1(0.05) = 0.5479795.$$

Extrapolating these results gives the new approximation

$$N_2(0.1) = N_1(0.05) + \left(N_1(0.05) - N_1(0.1)\right)$$
$$= 0.5479795 + (0.5479795 - 0.5406722) = 0.555287$$

The $h = 0.1$ and $h = 0.05$ results were found to be accurate to within 1.5×10^{-2} and 7.7×10^{-3}, respectively. Because $f'(1.8) = 1/1.8 = 0.\overline{5}$, the extrapolated value is accurate to within 2.7×10^{-4}.

Extrapolation can be applied whenever the truncation error for a formula has the form

$$\sum_{j=1}^{m-1} K_j h^{\alpha_j} + O(h^{\alpha_m}),$$

for a collection of constants K_j and when $\alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_m$. Many formulas used for extrapolation have truncation errors that contain only even powers of h, that is, have the form

$$M = N_1(h) + K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots. \quad (4.14)$$

$$M = N_1(h) + K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots. \quad (4.14)$$

The extrapolation is much more effective than when all powers of h are present because the averaging process produces results with errors $O(h^2), O(h^4), O(h^6), \ldots$, with essentially no increase in computation, over the results with errors, $O(h), O(h^2), O(h^3), \ldots$

Assume that approximation has the form of Eq. (4.14). Replacing h with $h/2$ gives the $O(h^2)$ approximation formula

$$M = N_1\left(\frac{h}{2}\right) + K_1\frac{h^2}{4} + K_2\frac{h^4}{16} + K_3\frac{h^6}{64} + \cdots.$$

Subtracting Eq. (4.14) from 4 times this equation eliminates the h^2 term,

$$3M = \left[4N_1\left(\frac{h}{2}\right) - N_1(h)\right] + K_2\left(\frac{h^4}{4} - h^4\right) + K_3\left(\frac{h^6}{16} - h^6\right) + \cdots.$$

Dividing this equation by 3 produces an $O(h^4)$ formula

$$M = \frac{1}{3}\left[4N_1\left(\frac{h}{2}\right) - N_1(h)\right] + \frac{K_2}{3}\left(\frac{h^4}{4} - h^4\right) + \frac{K_3}{3}\left(\frac{h^6}{16} - h^6\right) + \cdots.$$

Defining

$$N_2(h) = \frac{1}{3}\left[4N_1\left(\frac{h}{2}\right) - N_1(h)\right] = N_1\left(\frac{h}{2}\right) + \frac{1}{3}\left[N_1\left(\frac{h}{2}\right) - N_1(h)\right],$$

produces the approximation formula with truncation error $O(h^4)$:

$$M = N_2(h) - K_2\frac{h^4}{4} - K_3\frac{5h^6}{16} + \cdots. \quad (4.15)$$

Now replace h in Eq. (4.15) with $h/2$ to produce a second $O(h^4)$ formula

$$M = N_2\left(\frac{h}{2}\right) - K_2\frac{h^4}{64} - K_3\frac{5h^6}{1024} - \cdots.$$

Subtracting Eq. (4.15) from 16 times this equation eliminates the h^4 term and gives

$$15M = \left[16N_2\left(\frac{h}{2}\right) - N_2(h)\right] + K_3\frac{15h^6}{64} + \cdots.$$

Dividing this equation by 15 produces the new $O(h^6)$ formula

$$M = \frac{1}{15}\left[16N_2\left(\frac{h}{2}\right) - N_2(h)\right] + K_3\frac{h^6}{64} + \cdots.$$

We now have the $O(h^6)$ approximation formula

$$N_3(h) = \frac{1}{15}\left[16N_2\left(\frac{h}{2}\right) - N_2(h)\right] = N_2\left(\frac{h}{2}\right) + \frac{1}{15}\left[N_2\left(\frac{h}{2}\right) - N_2(h)\right].$$

Continuing this procedure gives, for each $j = 2,3,\ldots$, the $O\left(h^{2j}\right)$ approximation

$$\boldsymbol{N_j(h) = N_{j-1}\left(\frac{h}{2}\right) + \frac{N_{j-1}(h/2) - N_{j-1}(h)}{4^{j-1} - 1}}.$$

Table 4.6 shows the order in which the approximations are generated when

$$M = N_1(h) + K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots \qquad (4.16)$$

It is conservatively assumed that the true result is accurate at least to within the agreement of the bottom two results in the diagonal, in this case, to within $|N_3(h) - N_4(h)|$.

$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
1: $N_1(h)$			
2: $N_1(\frac{h}{2})$	**3:** $N_2(h)$		
4: $N_1(\frac{h}{4})$	**5:** $N_2(\frac{h}{2})$	**6:** $N_3(h)$	
7: $N_1(\frac{h}{8})$	**8:** $N_2(\frac{h}{4})$	**9:** $N_3(\frac{h}{2})$	**10:** $N_4(h)$

Table 4.6

Example 2

Taylor's theorem can be used to show that centered-difference formula in Eq. (4.5) to approximate $f'(x_0)$ can be expressed with an error formula:

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f'''(x_0) - \frac{h^4}{120}f^{(5)}(x_0) - \cdots.$$

Find approximations of order $O(h^2), O(h^4)$, and $O(h^6)$ for $f'(2.0)$ when $f(x) = xe^x$ and $h = 0.2$.

Solution The constants $K_1 = -f'''(x_0)/6, K_2 = -f^{(5)}(x_0)/120, \cdots$, are not likely to be known, but this is not important.

We have the $O(h^2)$ approximation

$$f'(x_0) = N_1(h) - \frac{h^2}{6} f'''(x_0) - \frac{h^4}{120} f^{(5)}(x_0) - \cdots, \quad (4.17)$$

where

$$N_1(h) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)].$$

x	$f(x)$
1.8	10.889365
1.9	12.703199
2.0	14.778112
2.1	17.148957
2.2	19.855030

This gives us the first $O(h^2)$ approximations

$$N_1(0.2) = \frac{1}{0.4}[f(2.2) - f(1.8)] = 2.5(19.855030 - 10.889365) = 22.414160,$$

And

$$N_1(0.1) = \frac{1}{0.2}[f(2.1) - f(1.9)] = 5(17.148957 - 12.703199) = 22.228786$$

Combining these to produce the first $O(h^4)$ approximation gives

$$N_2(0.2) = N_1(0.1) + \frac{1}{3}\big(N_1(0.1) - N_1(0.2)\big)$$

$$= 22.228786 + \frac{1}{3}(22.228786 - 22.414160) = 22.166995.$$

To determine an $O(h^6)$ formula we need another $O(h^4)$ result, which requires us to find the third $O(h^2)$ approximation

$$N_1(0.05) = \frac{1}{0.1}[f(2.05) - f(1.95)] = 10(15.924197 - 13.705941) = 22.182564.$$

We can now find the $O(h^4)$ approximation

$$N_2(0.1) = N_1(0.05) + \frac{1}{3}\big(N_1(0.05) - N_1(0.1)\big)$$

$$= 22.182564 + \frac{1}{3}(22.182564 - 22.228786) = 22.167157$$

and finally the $O(h^6)$ approximation

$$N_3(0.2) = N_2(0.1) + \frac{1}{15}\big(N_2(0.1) - N_1(0.2)\big)$$

$$= 22.167157 + \frac{1}{15}(22.167157 - 22.166995) = 22.167168$$

We would expect the final approximation to be accurate to at least the value 22.167 because the $N_2(0.2)$ and $N_3(0.2)$ give this same value. In fact, $N_3(0.2)$ is accurate to all the listed digits.

Each column beyond the first in the extrapolation table is obtained by a simple averaging process, so the technique can produce high-order approximations with minimal computational cost. However, as k increases, the round-off error in $N_1\left(h/2^k\right)$ will generally increase because the instability of numerical differentiation is related to the step size $h/2^k$. Also, the higher-order formulas depend increasingly on the entry to their immediate left in the table, which is the reason we recommend comparing the final diagonal entries to ensure accuracy.

In Section 4.1, the three-point methods were derived by differentiating a Lagrange interpolating polynomial for f. The five-point methods can be obtained in a similar manner, but the derivation is tedious. Extrapolation can be used to more easily derive these formulas, as illustrated below.

Illustration

Suppose we expand the function f in a fourth Taylor polynomial about x_0. Then

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \frac{1}{6}f'''(x_0)(x - x_0)^3$$
$$+ \frac{1}{24}f^{(4)}(x_0)(x - x_0)^4 + \frac{1}{120}f^{(5)}(\xi)(x - x_0)^5$$

for some number ξ between x and x_0. Evaluating f at $x_0 + h$ and $x_0 - h$ gives

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(x_0)h^4 + \frac{1}{120}f^{(5)}(\xi_1)h^5 \qquad (4.18)$$

and

$$f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f''(x_0)h^3 + \frac{1}{24}f^{(4)}(x_0)h^4 - \frac{1}{120}f^{(5)}(\xi_2)h^5, \qquad (4.19)$$

where $x_0 - h < \xi_2 < x_0 < \xi_1 < x_0 + h$.

Subtracting Eq. (4.19) from Eq. (4.18) gives a new approximation for $f'(x)$.

$$f(x_0+h)-f(x_0-h)=2hf'(x_0)+\frac{h^3}{3}f'''(x_0)+\frac{h^5}{120}\left[f^{(5)}(\xi_1)+f^{(5)}(\xi_2)\right],$$

which implies that (4.20)

$$f'(x_0)=\frac{1}{2h}[f(x_0+h)-f(x_0-h)]-\frac{h^2}{6}f'''(x_0)-\frac{h^4}{240}\left[f^{(5)}(\xi_1)+f^{(5)}(\xi_2)\right].$$

If $f^{(5)}$ is continuous on $[x_0-h, x_0+h]$, the Intermediate Value Theorem 1.11 implies that a number $\tilde{\xi}$ in (x_0-h, x_0+h) exists with

$$f^{(5)}(\tilde{\xi})=\frac{1}{2}\left[f^{(5)}(\xi_1)+f^{(5)}(\xi_2)\right]$$

As a consequence, we have the $O(h^2)$ approximation

$$f'(x_0)=\frac{1}{2h}[f(x_0+h)-f(x_0-h)]-\frac{h^2}{6}f'''(x_0)-\frac{h^4}{120}f^{(5)}(\tilde{\xi}) \quad (4.21)$$

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \quad (4.5)$$

Although the approximation in Eq. (4.21) is the same as that given in the three-point formula in Eq. (4.5), the unknown evaluation point occurs now in $f^{(5)}$, rather than in f'''. Extrapolation takes advantage of this by first replacing h in Eq. (4.21) with $2h$ to give the new formula

$$f'(x_0) = \frac{1}{4h}[f(x_0+2h) - f(x_0-2h)] - \frac{4h^2}{6}f'''(x_0) - \frac{16h^4}{120}f^{(5)}(\hat{\xi}) \quad (4.22)$$

where $\hat{\xi}$ is between $x_0 - 2h$ and $x_0 + 2h$.

Multiplying Eq. (4.21) by 4 and subtracting Eq. (4.22) produces

$$3f'(x_0) = \frac{2}{h}[f(x_0+h) - f(x_0-h)] - \frac{1}{4h}[f(x_0+2h) - f(x_0-2h)]$$
$$-\frac{h^4}{30}f^{(5)}(\tilde{\xi}) + \frac{2h^4}{15}f^{(5)}(\hat{\xi}).$$

$$3f'(x_0) = \frac{2}{h}[f(x_0+h) - f(x_0-h)] - \frac{1}{4h}[f(x_0+2h) - f(x_0-2h)] - \frac{h^4}{30}f^{(5)}(\tilde{\xi}) + \frac{2h^4}{15}f^{(5)}(\hat{\xi}).$$

Even if $f^{(5)}$ is continuous on $[x_0 - 2h, x_0 + 2h]$, the Intermediate Value Theorem 1.11 cannot be applied as we did to derive Eq. (4.21) because here we have the difference of terms involving $f^{(5)}$. However, an alternative method can be used to show that $f^{(5)}(\tilde{\xi})$ and $f^{(5)}(\hat{\xi})$ can still be replaced by a common value $f^{(5)}(\xi)$. Assuming this and dividing by 3 produces the five-point midpoint formula Eq. (4.6) that we saw in Section 4.1

$$f'(x_0) = \frac{1}{12h}[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)] + \frac{h^4}{30}f^{(5)}(\xi).$$

Five-Point Midpoint Formula

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4-4

Composite Numerical Integration

n	$C_k^{(n)}$								
1	$\frac{1}{2}$	$\frac{1}{2}$							
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$						
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$					
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$				
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$			
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$		
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$	
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{28350}$	$\frac{-4540}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

CSDN @大学一一二班

The Newton-Cotes formulas are generally unsuitable for use over large integration intervals. High-degree formulas would be required, and the values of the coefficients in these formulas are difficult to obtain. Also, the Newton-Cotes formulas are based on interpolatory polynomials that use equally-spaced nodes, a procedure that is inaccurate over large intervals because of the oscillatory nature of high-degree polynomials.

In this section, we discuss a piecewise approach to numerical integration that uses the low-order Newton-Cotes formulas. These are the techniques most often applied.

Example 1

Use Simpson's rule to approximate $\int_0^4 e^x\,dx$ and compare this to the results obtained by adding the Simpson's rule approximations for $\int_0^2 e^x\,dx$ and $\int_2^4 e^x\,dx$. Compare these approximations to the sum of Simpson's rule for $\int_0^1 e^x\,dx, \int_1^2 e^x\,dx, \int_2^3 e^x\,dx$, and $\int_3^4 e^x\,dx$.

Solution Simpson's rule on [0,4] uses $h = 2$ and gives

$$\int_0^4 e^x\,dx \approx \frac{2}{3}(e^0 + 4e^2 + e^4) = 56.76958$$

The exact answer in this case is $e^4 - e^0 = 53.59815$, and the error -3.17143 is far larger than we would normally accept.

Applying Simpson's rule on each of the intervals [0,2] and [2,4] uses $h = 1$ and gives

$$\int_0^4 e^x\,dx = \int_0^2 e^x\,dx + \int_2^4 e^x\,e^x dx$$
$$\approx \frac{1}{3}(e^0 + 4e + e^2) + \frac{1}{3}(e^2 + 4e^3 + e^4)$$
$$= \frac{1}{3}(e^0 + 4e + 2e^2 + 4e^3 + e^4)$$
$$= 53.86385$$

The error has been reduced to -0.26570.

For the integrals on $[0,1], [1,2], [3,4]$, and $[3,4]$ we use Simpson's rule four times with $h = \frac{1}{2}$ giving

$$\int_0^4 e^x\,dx = \int_0^1 e^x\,dx + \int_1^2 e^x\,dx + \int_2^3 e^x\,dx + \int_3^4 e^x\,dx$$
$$\approx \frac{1}{6}\left(e_0 + 4e^{1/2} + e\right) + \frac{1}{6}\left(e + 4e^{3/2} + e^2\right)$$
$$+ \frac{1}{6}\left(e^2 + 4e^{5/2} + e^3\right) + \frac{1}{6}\left(e^3 + 4e^{7/2} + e^4\right)$$
$$= \frac{1}{6}\left(e^0 + 4e^{1/2} + 2e + 4e^{3/2} + 2e^2 + 4e^{5/2} + 2e^3 + 4e^{7/2} + e^4\right)$$
$$= 53.61622$$

The error for this approximation has been reduced to -0.01807.

To generalize this procedure for an arbitrary integral $\int_a^b f(x)\,dx$, choose an even integer n. Subdivide the interval[a, b] into n subintervals, and apply Simpson's rule on each consecutive pair of subintervals.

Figure 4.7

With $h = (b-a)/n$ and $x_j = a + jh$, for each $j = 0,1,\dots,n$, we have

$$\int_a^b f(x)\,dx = \sum_{j=1}^{n/2} \int_{x_{2j-2}}^{x_{2j}} f(x)\,dx$$

$$= \sum_{j=1}^{n/2} \left\{\frac{h}{3}\left[f\left(x_{2j-2}\right) + 4f\left(x_{2j-1}\right) + f\left(x_{2j}\right)\right] - \frac{h^5}{90} f^{(4)}\left(\xi_j\right)\right\},$$

for some ξ_j with $x_{2j-2} < \xi_j < x_{2j}$, provided that $f \in C^4[a,b]$. we can reduce this sum to

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{(n/2)-1} f\left(x_{2j}\right) + 4\sum_{j=1}^{n/2} f\left(x_{2j-1}\right) + f(x_n)\right]$$
$$-\frac{h^5}{90}\sum_{j=1}^{n/2} f^{(4)}\left(\xi_j\right).$$

The error associated with this approximation is

$$E(f) = -\frac{h^5}{90}\sum_{j=1}^{n/2} f^{(4)}(\xi_j),$$

where $x_{2j-2} < \xi_j < x_{2j}$, for each $j = 1,2,\dots,n/2$.

If $f \in C^4[a,b]$, $f^{(4)}$ has maximum and minimum in $[a,b]$. Since

$$\min_{x\in[a,b]} f^{(4)}(x) \le f^{(4)}(\xi_j) \le \max_{x\in[a,b]} f^{(4)}(x),$$

we have

$$\frac{n}{2}\min_{x\in[a,b]} f^{(4)}(x) \le \sum_{j=1}^{n/2} f^{(4)}(\xi_j) \le \frac{n}{2}\max_{x\in[a,b]} f^{(4)}(x)$$

and

$$\min_{x\in[a,b]} f^{(4)}(x) \le \frac{2}{n}\sum_{j=1}^{n/2} f^{(4)}(\xi_j) \le \max_{x\in[a,b]} f^{(4)}(x).$$

there is a $\mu \in (a,b)$ such that

$$f^{(4)}(\mu) = \frac{2}{n}\sum_{j=1}^{n/2} f^{(4)}(\xi_j).$$

Thus

$$E(f) = -\frac{h^5}{90}\sum_{j=1}^{n/2} f^{(4)}(\xi_j) = -\frac{h^5}{180} n f^{(4)}(\mu)$$

or, since $h = (b-a)/n$,

$$E(f) = -\frac{(b-a)}{180} h^4 f^{(4)}(\mu)$$

Theorem 4.4

Let $f \in C^4[a,b]$, n be even, $h = (b-a)/n$, and $x_j = a + jh$, for each $j = 0,1,\dots,n$. There exists a $\mu \in (a,b)$ for which the **Composite Simpson's rule** for n subintervals can be written with its error term as

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{(n/2)-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)\right] - \frac{b-a}{180}h^4 f^{(4)}(\mu).$$

Notice that the error term for the Composite Simpson's rule is $O(h^4)$, whereas it was $O(h^5)$ for the standard Simpson's rule. However, these rates are not comparable because for standard Simpson's rule we have h fixed at $h = (b-a)/2$, but for Composite Simpson's rule we have $h = (b-a)/n$, for n an even integer. This permits us to considerably reduce the value of h when the Composite Simpson's rule is used.

ALGORITHM 4.1

Composite Simpson's Rule

To approximate the integral $I = \int_a^b f(x)dx$:

INPUT endpoints a, b; even positive integer n.

OUTPUT approximation XI to I.

Step 1 Set $h = (b - a)/n$.

Step 2 Set $XI0 = f(a) + f(b)$;

$XI1 = 0$; $(Summation\ of\ f(x_{2i-1}).)$

$XI2 = 0$. $(Summation\ of\ f(x_{2i}).)$

Step 3 For $i = 1, \ldots, n - 1$ do Steps 4 and 5.

Step 4 Set $X = a + ih$.

Step 5 If i is even then set $XI2 = XI2 + f(X)$
else set $XI1 = XI1 + f(X)$.

Step 6 Set $XI = h(XI0 + 2 \cdot XI2 + 4 \cdot XI1)/3$.

Step 7 **OUTPUT** (XI);
STOP.

The extensions of the Trapezoidal and Midpoint rules are given without proof. The Trapezoidal rule requires only one interval for each application, so the integer n can be either odd or even.

Figure 4.8

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi)$$

Theorem 4.5

Let $f \in [a, b], h = (b - a)/n$, and $x_j = a + jh$, for each $j = 0,1, \dots, n$. There exists a $\mu \in (a, b)$ for which the **Composite Trapezoidal rule** for n subintervals can be written with its error term as

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] - \frac{b-a}{12} h^2 f''(\mu).$$

Example 2

Determine values of h that will ensure an approximation error of less than 0.00002 when approximating $\int_0^{\pi} \sin x dx$ and employing

(a) Composite Trapezoidal rule and **(b)** Composite Simpson's rule.

Solution **(a)** The error form for the Composite Trapezoidal rule for $f(x) = \sin x$ on $[0, \pi]$ is

$$\left|\frac{\pi h^2}{12} f''(\mu)\right| = \left|\frac{\pi h^2}{12}(-\sin\mu)\right| = \frac{\pi h^2}{12}|\sin\mu|.$$

To ensure sufficient accuracy with this technique we need to have

$$\frac{\pi h^2}{12}|\sin\mu| \leq \frac{\pi h^2}{12} < 0.00002.$$

Since $h = \pi/n$ implies that $n = \pi/h$, we need

$$\frac{\pi^3}{12n^2} < 0.00002 \quad \text{which implies that} \quad n > \left(\frac{\pi^3}{12(0.00002)}\right)^{1/2} \approx 359.44$$

and the Composite Trapezoidal rule requires $n \geq 360$.

(b) The error form for the Composite Simpson's rule for $f(x) = \sin x$ on $[0, \pi]$ is

$$\left|\frac{\pi h^4}{180} f^{(4)}(\mu)\right| = \left|\frac{\pi h^4}{180} \sin\mu\right| = \frac{\pi h^4}{180} |\sin\mu|.$$

To ensure sufficient accuracy with this technique we need to have

$$\frac{\pi h^4}{180} |\sin\mu| \leq \frac{\pi h^4}{180} < 0.00002.$$

Using again the fact that $n = \pi/h$ gives

$$\frac{\pi^5}{180n^4} < 0.00002 \quad \text{which implies that} \quad n > \left(\frac{\pi^5}{180(0.00002)}\right)^{1/4} \approx 17.07.$$

So Composite Simpson's rule requires only $n \geq 18$.

Composite Simpson's rule with $n = 18$ gives

$$\int_0^{\pi} \sin x dx \approx \frac{\pi}{54}\left[2\sum_{j=1}^{8} \sin\left(\frac{j\pi}{9}\right) + 4\sum_{j=1}^{9} \sin\left(\frac{(2j-1)\pi}{18}\right)\right] = 2.0000104.$$

This is accurate to within about 10^{-5} because the true value is $-\cos(\pi) - \big(-\cos(0)\big) = 2$.

Composite Simpson's rule is the clear choice if you wish to minimize computation. For comparison purposes, consider the Composite Trapezoidal rule using $h = \pi/18$ for the integral in Example 2. This approximation uses the same function evaluations as Composite Simpson's rule but the approximation in this case

$$\int_0^{\pi} \sin x dx \approx \frac{\pi}{36}\left[2\sum_{j=1}^{17} \sin\left(\frac{j\pi}{18}\right) + \sin 0 + \sin\pi\right] = \frac{\pi}{36}\left[2\sum_{j=1}^{17} \sin\left(\frac{j\pi}{18}\right)\right]$$

$$= 1.9949205.$$

is accurate only to about 5×10^{-3}.

Round-Off Error Stability

In Example 2 we saw that ensuring an accuracy of 2×10^{-5} for approximating $\int_0^{\pi} \sin x dx$ required 360 subdivisions of $[0, \pi]$ for the Composite Trapezoidal rule and only 18 for Composite Simpson's rule. In addition to the fact that less computation is needed for the Simpson's technique, you might suspect that because of fewer computations this method would also involve less round-off error. However, an important property shared by all the composite integration techniques is a stability with respect to round-off error. That is, the round-off error does not depend on the number of calculations performed.

To demonstrate this rather amazing fact, suppose we apply the Composite Simpson's rule with n subintervals to a function f on $[a, b]$ and determine the maximum bound for the round-off error. Assume that $f(x_i)$ is approximated by $\tilde{f}(x_i)$ and that

$$f(x_i) = \tilde{f}(x_i) + e_i, \qquad \text{for each } i = 0,1,\dots,n,$$

where e_i denotes the round-off error associated with using $\tilde{f}(x_i)$ to approximate $f(x_i)$. Then the accumulated error, $e(h)$, in the Composite Simpson's rule is

$$e(h) = \left|\frac{h}{3}\left[e_0 + 2\sum_{j=1}^{(n/2)-1} e_{2j} + 4\sum_{j=1}^{n/2} e_{2j-1} + e_n\right]\right|$$

$$\leq \frac{h}{3}\left[|e_0| + 2\sum_{j=1}^{(n/2)-1} |e_{2j}| + 4\sum_{j=1}^{n/2} |e_{2j-1}| + |e_n|\right]$$

If the round-off errors are uniformly bounded by ε, then

$$e(h) \leq \frac{h}{3}\left[\varepsilon + 2\left(\frac{n}{2} - 1\right)\varepsilon + 4\left(\frac{n}{2}\right)\varepsilon + \varepsilon\right] = \frac{h}{3}3n\varepsilon = nh\varepsilon.$$

But $nh = b - a, so$

$$e(h) \leq (b - a)\varepsilon,$$

a bound independent of h (and n). This means that, even though we may need to divide an interval into more parts to ensure accuracy, the increased computation that is required does not increase the round-off error. This result implies that the procedure is stable as h approaches zero. Recall that this was not true of the numerical differentiation procedures considered at the beginning of this chapter.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4-5

Romberg Integration

In this section we will illustrate how Richardson extrapolation applied to results from the Composite Trapezoidal rule can be used to obtain high accuracy approximations with little computational cost.

In Section 4.4 we found that the Composite Trapezoidal rule has a truncation error of order $O(h^2)$. Specifically, we showed that for $h = (b-a)/n$ and $x_j = a + jh$ we have

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] - \frac{(b-a)f''(\mu)}{12}h^2$$

for some number μ in (a, b).

By an alternative method it can be shown, that if $f \in C^{\infty}[a,b]$, the Composite Trapezoidal rule can also be written with an error term in the form

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] + K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots, \quad (4.33)$$

where each K_i is a constant that depends only on $f^{(2i-1)}(a)$ and $f^{(2i-1)}(b)$.

Richardson extrapolation can be performed on any approximation procedure whose truncation error is of the form

$$\sum_{j=1}^{m-1} K_j h^{\alpha_j} + O(h^{\alpha_m}),$$

for a collection of constants K_j and when $\alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_m$. In that section we gave demonstrations to illustrate how effective this techniques is when the approximation procedure has a truncation error with only even powers of h, that is, when the truncation error has the form.

$$\sum_{j=1}^{m-1} K_j h^{2j} + O(h^{2m}).$$

Because the Composite Trapezoidal rule has this form, it is an obvious candidate for extrapolation. This results in a technique known as **Romberg integration**.

To approximate the integral $\int_a^b f(x)dx$ we use the results of the Composite Trapezoidal rule with $n = 1,2,4,8,16,\ldots$, and denote the resulting approximations, respectively, by $R_{1,1}, R_{2,1}, R_{3,1}$, etc. We then apply extrapolation, that is, we obtain $O(h^4)$ approximations $R_{2,2}, R_{3,2}, R_{4,2}$, etc., by

$$R_{k,2} = R_{k,1} + \frac{1}{3}\left(R_{k,1} - R_{k-1,1}\right), \qquad k = 2,3,\ldots$$

Then $O(h^6)$ approximations $R_{3,3}, R_{4,3}, R_{5,3}$, etc., by

$$R_{k,3} = R_{k,2} + \frac{1}{15}\left(R_{k,2} - R_{k-1,2}\right), \qquad k = 3,4,\ldots$$

In general, after the appropriate $R_{k,j-1}$ approximations have been obtained, we determine the $O\left(h^{2j}\right)$ approximations from

$$R_{k,j} = R_{k,j-1} + \frac{1}{4^{j-1} - 1}\left(R_{k,j-1} - R_{k-1,j-1}\right), \qquad k = j, j+1, \ldots$$

Example 1

Use the Composite Trapezoidal rule to find approximations to $\int_0^{\pi} \sin x dx$ with $n = 1,2,4,8,$ and 16 . Then perform Romberg extrapolation on the results.

The Composite Trapezoidal rule for the various values of n gives the following approximations to the true value 2 .

$$R_{1,1} = \frac{\pi}{2}[\sin 0 + \sin\pi] = 0;$$

$$R_{2,1} = \frac{\pi}{4}\left[\sin 0 + 2\sin\frac{\pi}{2} + \sin\pi\right] = 1.57079633;$$

$$R_{3,1} = \frac{\pi}{8}\left[\sin 0 + 2\left(\sin\frac{\pi}{4} + \sin\frac{\pi}{2} + \sin\frac{3\pi}{4}\right) + \sin\pi\right] = 1.89611890;$$

$$R_{4,1} = \frac{\pi}{16}\left[\sin 0 + 2\left(\sin\frac{\pi}{8} + \sin\frac{\pi}{4} + \cdots + \sin\frac{3\pi}{4} + \sin\frac{7\pi}{8}\right) + \sin\pi\right] = 1.97423160;$$

$$R_{5,1} = \frac{\pi}{32}\left[\sin 0 + 2\left(\sin\frac{\pi}{16} + \sin\frac{\pi}{8} + \cdots + \sin\frac{7\pi}{8} + \sin\frac{15\pi}{16}\right) + \sin\pi\right] = 1.99357034.$$

The $O(h^4)$ approximations are

$$R_{2,2} = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) = 2.09439511; \quad R_{3,2} = R_{3,1} + \frac{1}{3}(R_{3,1} - R_{2,1}) = 2.00455976;$$

$$R_{4,2} = R_{4,1} + \frac{1}{3}(R_{4,1} - R_{3,1}) = 2.00026917; \quad R_{5,2} = R_{5,1} + \frac{1}{3}(R_{5,1} - R_{4,1}) = 2.00001659;$$

The $O(h^6)$ approximations are

$$R_{3,3} = R_{3,2} + \frac{1}{15}(R_{3,2} - R_{2,2}) = 1.99857073; \quad R_{4,3} = R_{4,2} + \frac{1}{15}(R_{4,2} - R_{3,2}) = 1.99998313;$$

$$R_{5,3} = R_{5,2} + \frac{1}{15}(R_{5,2} - R_{4,2}) = 1.99999975.$$

The two $O(h^8)$ approximations are

$$R_{4,4} = R_{4,3} + \frac{1}{63}(R_{4,3} - R_{3,3}) = 2.00000555;\ R_{5,4} = R_{5,3} + \frac{1}{63}(R_{5,3} - R_{4,3}) = 2.00000001$$

and the final $O(h^{10})$ approximation is

$$R_{5,5} = R_{5,4} + \frac{1}{255}(R_{5,4} - R_{4,4}) = 1.99999999.$$

These results are shown in Table 4.9.

0				
1.57079633	2.09439511			
1.89611890	2.00455976	1.99857073		
1.97423160	2.00026917	1.99998313	2.00000555	
1.99357034	2.00001659	1.99999975	2.00000001	1.99999999

Table 4.9

Notice that when generating the approximations for the Composite Trapezoidal rule approximations in Example 1, each consecutive approximation included all the functions evaluations from the previous approximation. That is, $R_{1,1}$ used evaluations at 0 and $\pi, R_{2,1}$ used these evaluations and added an evaluation at the intermediate point $\pi/2$. Then $R_{3,1}$ used the evaluations of $R_{2,1}$ and added two additional intermediate ones at $\pi/4$ and $3\pi/4$. This pattern continues with $R_{4,1}$ using the same evaluations as $R_{3,1}$ but adding evaluations at the 4 intermediate points $\pi/8, 3\pi/8, 5\pi/8$, and $7\pi/8$, and so on.

This evaluation procedure for Composite Trapezoidal rule approximations holds for an integral on any interval $[\mathrm{a}, \mathrm{b}]$. In general, the Composite Trapezoidal rule denoted $R_{k+1,1}$ uses the same evaluations as $R_{k,1}$ but adds evaluations at the 2^{k-2} intermediate points. Efficient calculation of these approximations can therefore be done in a recursive manner.

To obtain the Composite Trapezoidal rule approximations for $\int_a^b f(x)dx$, let $h_k = (b-a)/m_k = (b-a)/2^{k-1}$. Then

$$R_{1,1} = \frac{h_1}{2}[f(a) + f(b)] = \frac{b-a}{2}[f(a) + f(b)];$$

and

$$R_{2,1} = \frac{h_2}{2}[f(a) + f(b) + 2f(a + h_2)].$$

By reexpressing this result for $R_{2,1}$ we can incorporate the previously determined approximation $R_{1,1}$

$$R_{2,1} = \frac{b-a}{4}\left[f(a) + f(b) + 2f\left(a + \frac{b-a}{2}\right)\right] = \frac{1}{2}[R_{1,1} + h_1 f(a + h_2)]$$

In a similar manner we can write

$$R_{3,1} = \frac{1}{2}\{R_{2,1} + h_2[f(a + h_3) + f(a + 3h_3)]\};$$

and, in general we have

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i - 1)h_k)\right], \qquad (4.34)$$

for each $k = 2,3, \dots, n$. (See Exercises 14 and 15.)

Figure 4.10

Extrapolation then is used to produce $O\left(h_k^{2j}\right)$ approximations by

$$R_{k,j} = R_{k,j-1} + \frac{1}{4^{j-1}-1}\left(R_{k,j-1} - R_{k-1,j-1}\right), \text{for } k = j, j+1, \ldots$$

as shown in Table 4.10.

k	$O\left(h_k^2\right)$	$O\left(h_k^4\right)$	$O\left(h_k^6\right)$	$O\left(h_k^8\right)$		$O\left(h_k^{2n}\right)$
1	$R_{1,1}$					
2	$R_{2,1}$	$R_{2,2}$				
3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
n	$R_{n,1}$	$R_{n,2}$	$R_{n,3}$	$R_{n,4}$	$\cdots$	$R_{n,n}$

Table 4.10

The effective method to construct the Romberg table makes use of the highest order of approximation at each step. That is, it calculates the entries row by row, in the order $R_{1,1}, R_{2,1}, R_{2,2}, R_{3,1}, R_{3,2}, R_{3,3}$, etc. This also permits an entire new row in the table to be calculated by doing only one additional application of the Composite Trapezoidal rule. It then uses a simple averaging on the previously calculated values to obtain the remaining entries in the row. Remember

- **Calculate the Romberg table one complete row at a time**.

Example 2

Add an additional extrapolation row to Table 4.9 to approximate $\int_0^{\pi} \sin x dx$.

0				
1.57079633	2.09439511			
1.89611890	2.00455976	1.99857073		
1.97423160	2.00026917	1.99998313	2.00000555	
1.99357034	2.00001659	1.99999975	2.00000001	1.99999999

Solution To obtain the additional row we need the trapezoidal approximation

$$R_{6,1} = \frac{1}{2}\left[R_{5,1} + \frac{\pi}{16}\sum_{k=1}^{2^4} \sin\frac{(2k-1)\pi}{32}\right] = 1.99839336.$$

$$R_{6,2} = R_{6,1} + \frac{1}{3}\left(R_{6,1} - R_{5,1}\right) = 1.99839336 + \frac{1}{3}(1.99839336 - 1.99357035)$$
$$= 2.00000103$$

$$R_{6,3} = R_{6,2} + \frac{1}{15}\left(R_{6,2} - R_{5,2}\right) = 2.00000103 + \frac{1}{15}(2.00000103 - 2.00001659)$$
$$= 2.00000000$$

$$R_{6,4} = R_{6,3} + \frac{1}{63}\left(R_{6,3} - R_{5,3}\right) = 2.00000000$$

$$R_{6,5} = R_{6,4} + \frac{1}{255}\left(R_{6,4} - R_{5,4}\right) = 2.00000000$$

and $R_{6,6} = R_{6,5} + \frac{1}{1023}\left(R_{6,5} - R_{5,5}\right) = 2.00000000.$

The new extrapolation table is shown in Table 4.11

$$R_{6,3} = R_{6,2} + \frac{1}{15}\left(R_{6,2} - R_{5,2}\right)$$

$$= 2.00000103 + \frac{1}{15}(2.00000103 - 2.00001659) = 2.00000000$$

$$R_{6,4} = R_{6,3} + \frac{1}{63}\left(R_{6,3} - R_{5,3}\right) = 2.00000000$$

$$R_{6,5} = R_{6,4} + \frac{1}{255}\left(R_{6,4} - R_{5,4}\right) = 2.00000000$$

and $R_{6,6} = R_{6,5} + \frac{1}{1023}\left(R_{6,5} - R_{5,5}\right) = 2.00000000.$

The new extrapolation table is shown in Table 4.11

0					
1.57079633	2.09439511				
1.89611890	2.00455976	1.99857073			
1.97423160	2.00026917	1.99998313	2.00000555		
1.99357034	2.00001659	1.99999975	2.00000001	1.99999999	
1.99839336	2.00000103	2.00000000	2.00000000	2.00000000	2.00000000

Table 4.11

Notice that all the extrapolated values except for the first (in the first row of the second column) are more accurate than the best composite trapezoidal approximation (in the last row of the first column).

Although there are 21 entries in Table 4.11, only the six in the left column require function evaluations since these are the only entries generated by the Composite Trapezoidal rule; the other entries are obtained by an averaging process. In fact, because of the recurrence relationship of the terms in the left column, the only function evaluations needed are those to compute the final Composite Trapezoidal rule approximation. In general, $R_{k,1}$ requires $1 + 2^{k-1}$ function evaluations, so in this case $1 + 2^5 = 33$ are needed.

Algorithm 4.2 uses the recursive procedure to find the initial Composite Trapezoidal Rule approximations and computes the results in the table row by row.

ALGORITHM 4.2

Romberg

To approximate the integral $I = \int_a^b f(x)dx$, select an integer $n > 0$.

INPUT endpoints a, b; integer n.

OUTPUT an array R. (*Compute R by rows; only the last 2 rows are saved in storage.*)

Step 1 Set $h = b - a$;

$$R_{1,1} = \frac{h}{2}\big(f(a) + f(b)\big).$$

Step 2 OUTPUT $(R_{1,1})$.

Step 3 For $i = 2, \ldots, n$ do Steps 4-8.

4UFQア; Set $R_{2,1} = \frac{1}{2}\left[R_{1,1} + h\sum_{k=1}^{2^{i-2}} f(a + (k - 0.5)h)\right]$.

(*Approximation from Trapezoidal method.*)

Step 5 For $j = 2, \ldots, i$

$$\text{set} R_{2,j} = R_{2,j-1} + \frac{R_{2,j-1} - R_{1,j-1}}{4^{j-1} - 1} (Extrapolation.)$$

Step 6 OUTPUT ($R_{2,J}$ for j=1,2,...,i)

Step 7 Set $h = h/2$.

Step 8 For $j = 1,2, \ldots, i$ set $R_{1,j} = R_{2,j}$. (*Update row 1 of R.*)

Step 9 STOP.

Algorithm 4.2 requires a preset integer n to determine the number of rows to be generated. We could also set an error tolerance for the approximation and generate n, within some upper bound, until consecutive diagonal entries $R_{n-1,n-1}$ and $R_{n,n}$ agree to within the tolerance. To guard against the possibility that two consecutive row elements agree with each other but not with the value of the integral being approximated, it is common to generate approximations until not only $\left|R_{n-1,n-1} - R_{n,n}\right|$ is within the tolerance, but also $\left|R_{n-2,n-2} - R_{n-1,n-1}\right|$. Although not a universal safeguard, this will ensure that two differently generated sets of approximations agree within the specified tolerance before $R_{n,n}$, is accepted as sufficiently accurate.

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] + K_1h^2 + K_2h^4 + K_3h^6 + \cdots, \quad (4.33)$$

Romberg integration applied to a function f on the interval $[a, b]$ relies on the assumption that the Composite Trapezoidal rule has an error term that can be expressed in the form of Eq. (4.33) ;that is, we must have $f \in C^{2k+2}[a, b]$ for the kth row to be generated. General-purpose algorithms using Romberg integration include a check at each stage to ensure that this assumption is fulfilled. These methods are known as cautious Romberg algorithms and are described in [Joh].

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 4-7

Gaussian Quadrature

The Newton-Cotes formulas in Section 4.3 were derived by integrating interpolating polynomials. The error term in the interpolating polynomial of degree n involves the $(n+1)$ st derivative of the function being approximated, so a Newton-Cotes formula is exact when approximating the integral of any polynomial of degree less than or equal to n.

All the Newton-Cotes formulas use values of the function at equally-spaced points. This restriction is convenient when the formulas are combined to form the composite rules we considered in Section 4.4, but it can significantly decrease the accuracy of the approximation. Consider, for example, the Trapezoidal rule applied to determine the integrals of the functions whose graphs are shown in Figure 4.15.

Figure 4.15

The Trapezoidal rule approximates the integral of the function by integrating the linear function that joins the endpoints of the graph of the function. But this is not likely the best line for approximating the integral. Lines such as those shown in Figure 4.16 would likely give much better approximations in most cases

Figure 4.16

Gaussian quadrature chooses the points for evaluation in an optimal, rather than equally-spaced, way. The nodes $x_1, x_2, \dots, x_n$ in the interval $[a, b]$ and coefficients $c_1, c_2, \dots, c_n$, are chosen to minimize the expected error obtained in the approximation

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i).$$

To measure this accuracy, we assume that the best choice of these values produces the exact result for the largest class of polynomials, that is, the choice that gives the greatest degree of precision.

The coefficients $c_1, c_2, \dots, c_n$ in the approximation formula are arbitrary, and the nodes $x_1, x_2, \dots, x_n$ are restricted only by the fact that they must lie in $[\mathrm{a}, \mathrm{b}]$, the interval of integration. This gives us $2n$ parameters to choose. If the coefficients of a polynomial are considered parameters, the class of polynomials of degree at most $2n-1$ also contains $2n$ parameters. This, then, is the largest class of polynomials for which it is reasonable to expect a formula to be exact. With the proper choice of the values and constants, exactness on this set can be obtained.

To illustrate the procedure for choosing the appropriate parameters, we will show how to select the coefficients and nodes when $n = 2$ and the interval of integration is $[-1,1]$. We will then discuss the more general situation for an arbitrary choice of nodes and coefficients and show how the technique is modified when integrating over an arbitrary interval.

Suppose we want to determine c_1, c_2, x_1, and x_2 so that the integration formula

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

gives the exact result whenever $f(x)$ is a polynomial of degree $2(2) - 1 = 3$ or less, that is, when

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$$

for some collection of constants, a_0, a_1, a_2, and a_3. Because

$$\int (a_0 + a_1 x + a_2 x^2 + a_3 x^3)\, dx = a_0 \int 1dx + a_1 \int xdx + a_2 \int x^2 dx + a_3 \int x^3 dx,$$

this is equivalent to showing that the formula gives exact results when $f(x)$ is $1, x, x^2$, and x^3. Hence, we need c_1, c_2, x_1, and x_2, so that

$$c_1 \cdot 1 + c_2 \cdot 1 = \int_{-1}^{1} 1dx = 2, \qquad c_1 \cdot x_1 + c_2 \cdot x_2 = \int_{-1}^{1} xdx = 0,$$

$$c_1 \cdot x_1^2 + c_2 \cdot x_2^2 = \int_{-1}^{1} x^2dx = \frac{2}{3}, \text{and} \quad c_1 \cdot x_1^3 + c_2 \cdot x_2^3 = \int_{-1}^{1} x^3dx = 0.$$

A little algebra shows that this system of equations has the unique solution

$$c_1 = 1, c_2 = 1, x_1 = -\frac{\sqrt{3}}{3}, \qquad \text{and} \quad x_2 = \frac{\sqrt{3}}{3},$$

which gives the approximation formula

$$\int_{-1}^{1} f(x)dx \approx f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right). \quad (4.40)$$

This formula has degree of precision 3 , that is, it produces the exact result for every polynomial of degree 3 or less.

Legendre Polynomials

The technique we have described could be used to determine the nodes and coefficients for formulas that give exact results for higher-degree polynomials, but an alternative method obtains them more easily. In Sections 8.2 and 8.3 we will consider various collections of orthogonal polynomials, functions that have the property that a particular definite integral of the product of any two of them is 0 . The set that is relevant to our problem is the Legendre polynomials, a collection $\{P_0(x), P_1(x), \dots P_n(x), \dots,\}$with properties:

Legendre polynomials: $\{P_0(x), P_1(x), \dots P_n(x), \dots,\}$

(1) For each $n, P_n(x)$ is a monic polynomial of degree n.

(2) $\int_{-1}^{1} P(x)P_n(x)dx = 0$ whenever $P(x)$ is a polynomial of degree less than n.

The first few Legendre polynomials are

$$P_0(x) = 1, \qquad P_1(x) = x, \qquad P_2(x) = x^2 - \frac{1}{3},$$

$$P_3(x) = x^3 - \frac{3}{5}x, \qquad \text{and} \quad P_4(x) = x^4 - \frac{6}{7}x^2 + \frac{3}{35}.$$

The roots of these polynomials are distinct, lie in the interval $(-1,1)$, have a symmetry with respect to the origin, and, most importantly, are the correct choice for determining the parameters that give us the nodes and coefficients for our quadrature method.

The nodes $x_1, x_2, \dots, x_n$ needed to produce an integral approximation formula that gives exact results for any polynomial of degree less than $2n$ are the roots of the n th-degree Legendre polynomial. This is established by the following result.

Theorem 4.7

Suppose that $x_1, x_2, \dots, x_n$ are the roots of the nth Legendre polynomial $P_n(x)$ and that for each $i = 1,2, \dots, n$, the numbers c_i are defined by

$$c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} dx.$$

If $P(x)$ is any polynomial of degree less than $2n$, then

$$\int_{-1}^{1} P(x) dx = \sum_{i=1}^{n} c_i P(x_i).$$

Proof Let us first consider the situation for a polynomial $P(x)$ of degree less than n. Rewrite $P(x)$ in terms of $(n-1)$st Lagrange coefficient polynomials with nodes at the roots of the nth Legendre polynomial $P_n(x)$. The error term for this representation involves the nth derivative of $P(x)$. Since $P(x)$ is of degree less than n, the nth derivative of $P(x)$ is 0 , and this representation of is exact. So

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

and

$$\int_{-1}^{1} P(x)dx = \int_{-1}^{1} \left[\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} P(x_i)\right] dx$$

$$= \sum_{i=1}^{n} \left[\int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} dx\right] P(x_i) = \sum_{i=1}^{n} c_i P(x_i).$$

Hence the result is true for polynomials of degree less than n.

Now consider a polynomial $P(x)$ of degree at least n but less than $2n$. Divide $P(x)$ by the nth Legendre polynomial $P_n(x)$. This gives two polynomials $Q(x)$ and $R(x)$, each of degree less than n, with

$$P(x) = Q(x)P_n(x) + R(x).$$

Note that x_i is a root of $P_n(x)$ for each $i = 1,2,\dots,n$, so we have

$$P(x_i) = Q(x_i)P_n(x_i) + R(x_i) = R(x_i).$$

We now invoke the unique power of the Legendre polynomials. First, the degree of the polynomial $Q(x)$ is less than n, so

$$\int_{-1}^{1} Q(x)P_n(x)dx = 0.$$

Then, since $R(x)$ is a polynomial of degree less than n, the opening argument implies that

$$\int_{-1}^{1} R(x)dx = \sum_{i=1}^{n} c_i R(x_i).$$

Putting these facts together verifies that the formula is exact for the polynomial $P(x)$:

$$\int_{-1}^{1} P(x)dx = \int_{-1}^{1} [Q(x)P_n(x) + R(x)]dx$$
$$= \int_{-1}^{1} R(x)dx = \sum_{i=1}^{n} c_i R(x_i) = \sum_{i=1}^{n} c_i P(x_i)$$

n	Roots $r_{n,i}$	Coefficients $c_{n,i}$
2	0.5773502692	1.0000000000
	−0.5773502692	1.0000000000
3	0.7745966692	0.5555555556
	0.0000000000	0.8888888889
	−0.7745966692	0.5555555556
4	0.8611363116	0.3478548451
	0.3399810436	0.6521451549
	−0.3399810436	0.6521451549
	−0.8611363116	0.3478548451
5	0.9061798459	0.2369268850
	0.5384693101	0.4786286705
	0.0000000000	0.5688888889
	−0.5384693101	0.4786286705
	−0.9061798459	0.2369268850

Table 4.12

Example 1

Approximate $\int_{-1}^{1} e^x \cos x dx$ using Gaussian quadrature with $n = 3$.

Solution The entries in Table 4.12 give us

$$\int_{-1}^{1} e^x \cos x dx \approx 0.\bar{5}e^{0.774596692}\cos 0.774596692$$
$$+0.\bar{8}\cos 0 + 0.\bar{5}e^{-0.774596692}\cos(-0.774596692)$$
$$= 1.9333904$$

Integration by parts can be used to show that the true value of the integral is 1.9334214, so the absolute error is less than 3.2×10^{-5}.

n	Roots $r_{n,i}$	Coefficients $c_{n,i}$
2	0.5773502692	1.0000000000
	−0.5773502692	1.0000000000
3	0.7745966692	0.5555555556
	0.0000000000	0.8888888889
	−0.7745966692	0.5555555556
4	0.8611363116	0.3478548451
	0.3399810436	0.6521451549
	−0.3399810436	0.6521451549
	−0.8611363116	0.3478548451
5	0.9061798459	0.2369268850
	0.5384693101	0.4786286705
	0.0000000000	0.5688888889
	−0.5384693101	0.4786286705
	−0.9061798459	0.2369268850

Gaussian Quadrature on Arbitrary Intervals

An integral $\int_a^b f(x)dx$ over an arbitrary [a, b] can be transformed into an integral over $[-1,1]$ by using the change of variables (see Figure 4.17):

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

Figure 4.17

This permits Gaussian quadrature to be applied to any interval $[a, b]$, because

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{(b-a)t+(b+a)}{2}\right)\frac{(b-a)}{2}dt. \quad (4.41)$$

Example 2 Consider the integral $\int_1^3 x^6 - x^2\sin(2x)dx = 317.3442466$.

(a) Compare the results for the closed Newton-Cotes formula with $n = 1$ and Gaussian Quadrature when $n = 2$.

Solution **(a)** Each of the formulas in this part requires 2 evaluations of the function $f(x) = x^6 - x^2\sin(2x)$. The Newton-Cotes approximations are

$$\textbf{Closed } n = 1: \quad \frac{2}{2}[f(1) + f(3)] = 731.6054420;$$

Gaussian quadrature:

$$\int_1^3 x^6 - x^2\sin(2x)dx = \int_{-1}^1 (t+2)^6 - (t+2)^2\sin\big(2(t+2)\big)dt.$$

$$\approx f(-0.5773502692 + 2) + f(0.5773502692 + 2) = 306.8199344$$

Example 2 Consider the integral $\int_1^3 x^6 - x^2\sin(2x)dx = 317.3442466$.

(b) Compare the results for the closed Newton-Cotes formula with $n = 2$, and Gaussian Quadrature when $n = 3$.

The Newton-Cotes approximations are

Closed $n = 1$: $\frac{1}{3}[f(1) + 4f(2) + f(3)] = 333.2380940$;

Gaussian quadrature

$$\int_1^3 x^6 - x^2\sin(2x)dx \approx 0.\bar{5}f(-0.7745966692 + 2) + 0.\bar{8}f(2) + 0.\bar{5}f(0.7745966692 + 2) = 317.2641516.$$

The Gaussian quadrature results are clearly superior in each instance.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Chapter 5

Initial-Value Problems for Ordinary Differential Equations

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-1

The Elementary Theory of Initial-Value Problems

Introduction

In common real-life situations, the differential equation that models the problem is too complicated to solve exactly, and one of two approaches is taken to approximate the solution.

- The first approach is to modify the problem by simplifying the differential equation to one that can be solved exactly and then use the solution of the simplified equation to approximate the solution to the original problem.

- The other approach, which we will examine in this chapter, uses methods for approximating the solution of the original problem.

Introduction

- The methods that we consider in this chapter do not produce a continuous approximation to the solution of the initial-value problem.

- Rather, approximations are found at certain specified, and often equally spaced, points.

- Some method of interpolation, commonly Hermite, is used if intermediate values are needed.

Definition 5.1 A function $f(t, y)$ is said to satisfy a **Lipschitz condition** in the variable y on a set $D \subset R^2$ if a constant $L > 0$ exists with

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

whenever(t, y_1) and (t, y_2) are in D. The constant L is called a **Lipschitz constant** for f .

Example 1 Show that $f(t,y) = t|y|$ satisfies a Lipschitz condition on the interval $D = \{(t,y)|1 \le t \le 2 \text{ and } -3 \le y \le 4\}$.

Solution: For each pair of points (t, y_1) and (t, y_2) in D we have

$$|f(t,y_1) - f(t,y_2)| = |t|y_1| - t|y_2|| = |t|||y_1| - |y_2|| \le 2|y_1 - y_2|$$

Thus f satisfies a Lipschitz condition on D in the variable y with Lipschitz constant 2. The smallest value possible for the Lipschitz constant for this problem is $L = 2$, because, for example,

$$|f(2,1) - f(2,0)| = |2 - 0| = 2|1 - 0|$$

Definition 5.2 A set $D \subset R^2$ is said to be **convex** if whenever (t_1, y_1) and (t_2, y_2)belong to D, then $((1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2)$ also belongs to D for every λ in [0,1].

In geometric terms, Definition 5.2 states that a set is convex provided that whenever two points belong to the set, the entire straight-line segment between the points also belongs to the set. (See Figure 5.1.)

Figure 5.1

The sets we consider in this chapter are generally of the form $D = \{(t, y) | a \leq t \leq b \text{ and } -\infty < y < \infty\}$ for some constants a and b. These sets are convex.

Theorem 5.3 Suppose $f(t, y)$ is defined on a **convex set** $D \subset R^2$. If a constant L > 0 exists with

$$\left|\frac{\partial f}{\partial y}(t, y)\right| \leq L, \qquad \text{for all } (t, y) \in D$$

then f satisfies a Lipschitz condition on D in the variable y with Lipschitz constant L.

Note: Theorem 5.3 gives only sufficient conditions for a Lipschitz condition to hold.

Theorem 5.4 Suppose that D = {(t, y) | a ≤ t ≤ b and −∞ < y < ∞} and that $f(t, y)$ is continuous on D. If f satisfies a Lipschitz condition on D in the variable y, then the initial-value problem

$$y'(t) = f(t, y), a \leq t \leq b, y(a) = \alpha$$

has a unique solution $y(t)$ for $a \leq t \leq b$.

- The theorem is a version of the fundamental existence and uniqueness theorem for first-order ordinary differential equations.

Example 2 Use Theorem 5.4 to show that there is a unique solution to the initial-value problem

$$y' = 1 + t\sin(ty), \qquad 0 \le t \le 2, \qquad y(0) = 0$$

Well-Posed Problems

- Now that we have, to some extent, taken care of the question of when initial-value problems have unique solutions.
- we can move to the second important consideration when approximating the solution to an initial-value problem.

Question: How do we determine whether a particular problem has the property that small changes, or perturbations, in the statement of the problem introduce correspondingly small changes in the solution?

Definition 5.5 The initial-value problem

$$\frac{dy}{dt} = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

is said to be a **well-posed problem** if:

- A unique solution, $y(t)$, to the problem exists, and
- There exist constants $\varepsilon_0 > 0$ and $k > 0$ such that for any ε, with $\varepsilon_0 > \varepsilon > 0$, whenever $\delta(t)$ is continuous with $|\delta(t)| < \varepsilon$ for all t in $[a, b]$, and when $|\delta_0| < \varepsilon$, the initial-value problem

$$\frac{dz}{dt} = f(t, z) + \delta(t), \qquad a \le t \le b, \qquad z(a) = \alpha + \delta_0$$

has a unique solution $z(t)$ that satisfies

$$|z(t) - y(t)| < k\varepsilon \quad \text{for all t in } [a, b].$$

Theorem 5.6 Suppose D = {(t, y) | a ≤ t ≤ b and −∞ < y < ∞}. If f is continuous and satisfies a Lipschitz condition in the variable y on the set D, then the initial-value problem

$$\frac{dy}{dt} = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

is well-posed.

- The theorem specifies conditions that ensure that an initial-value problem is well-posed.

Example 3 Show that the initial-value problem

$$\frac{dy}{dt} = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5 \qquad (5.4)$$

is well posed on $D = \{(t, y) \mid 0 \le t \le 2 \text{ and } -\infty < y < \infty\}$.

Solution: Because

$$\left|\frac{\partial(y - t^2 + 1)}{\partial y}\right| = |1| = 1$$

Theorem 5.3 implies that $f(t, y) = y - t^2 + 1$ satisfies a Lipschitz condition in y on D with Lipschitz constant 1. Since f is continuous on D, Theorem 5.6 implies that the problem is well-posed.

As an illustration, consider the solution to the perturbed problem

$$\frac{dz}{dt} = z - t^2 + 1 + \delta, \qquad 0 \le t \le 2, \qquad z(0) = 0.5 + \delta_0 \qquad (5.5)$$

where δ and δ_0 are constants. The solutions to Eqs. (5.4) and (5.5) are

$$y(t) = (t+1)^2 - 0.5e^t \text{ and } z(t) = (t+1)^2 + (\delta + \delta_0 - 0.5)e^t - \delta$$

respectively.

Suppose that ε is a positive number. If $|\delta| < \varepsilon$ and $|\delta_0| < \varepsilon$, then

$$|y(t) - z(t)| = |(\delta + \delta_0)e^t - \delta| \le |\delta + \delta_0|e^2 + |\delta| \le (2e^2 + 1)\varepsilon$$

for all t. This implies that problem (5.4) is well-posed with $k(\varepsilon) = 2e^2 + 1$ for all $\varepsilon > 0$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-2

Euler's Method

Introduction

- Euler's method is the most elementary approximation technique for solving initial-value problems.

- The object of Euler's method is to obtain approximations to the well-posed initial-value problem

$$\frac{dy}{dt} = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha \qquad (5.6)$$

Introduction

- A continuous approximation to the solution $y(t)$ will not be obtained; instead, approximations to y will be generated at various values, called **mesh points**, in the interval $[a, b]$.
- We first make the stipulation that the mesh points are equally distributed throughout the interval $[a, b]$.

$$t_i = a + ih, \qquad \text{for each } i = 0,1,2,\ldots,N$$

The common distance between the points $h = (b - a)/N = t_{i+1} - t_i$ is called the **step size**.

$$\frac{dy}{dt} = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha \qquad (5.6)$$

Taylor's Theorem to derive Euler's method

- Suppose that $y(t)$, the unique solution to (5.6), has two continuous derivatives on $[a, b]$,so that for each i = 0, 1, 2, ... ,N − 1

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i)$$

for some number ξ_i in (t_i, t_{i+1}). Because $h = t_{i+1} - t_i$, we have

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(\xi_i)$$

and, because $y(t)$ satisfies the differential equation (5.6),

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} y''(\xi_i). \qquad (5.7)$$

- Euler's method constructs $\omega_i \approx y(t_i)$, for each i = 1, 2, ... , N, by deleting the remainder term. Thus Euler's method is

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + hf(t_i, \omega_i) \end{cases} \quad i = 0,1,\ldots,N-1. \quad (5.8)$$

- Equation (5.8) is called the **difference equation** associated with Euler's method.

Illustration In Example 1, we will use Euler's method to approximate the solution to

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

at t = 2. Here we will simply illustrate the steps in the technique when we have h = 0.5.

ALGORITHM 5.1 Euler's

To approximate the solution of the initial-value problem

$$y' = f(t, y), a \leq t \leq b, \qquad y(a) = \alpha$$

at $(N + 1)$ equally spaced numbers in the interval $[a, b]$:

INPUT *endpoints a, b; integer N; initial condition α.*

OUTPUT *approximation ω to y at the (N + 1) values of t.*

Step 1 *Set h= (b-a)/N;*

$t = a.$

$\omega = \alpha$

OUTPUT (t,ω)

Step 2 For *i=1,2,...,N do Steps 3,4.*

Step 3 *Set* $\omega = \omega + hf(t,\omega)$; *(Compute* ω_i*.)*

t=a+ih. (Compute t_i*.)*

Step 4 OUTPUT (t,ω)

Step 5 *STOP.*

➢To interpret Euler's method geometrically, note that when ω_i is a close approximation to $y(t_i)$, the assumption that the problem is well-posed implies that

$$f(t_i, \omega_i) \approx y'(t_i) = f(t_i, y(t_i))$$

The graph of the function highlighting $y(t_i)$ is shown in Figure 5.2.

Figure 5.2

One step in Euler's method appears in Figure 5.3, and a series of steps appears in Figure 5.4.

Figure 5.3

Figure 5.4

Example 1 Euler's method was used in the first illustration with h = 0.5 to approximate the solution to the initial-value problem

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

Use Algorithm 5.1 with N = 10 to determine approximations, and compare these with the exact values given by $y(t) = (t+1)^2 - 0.5e^t$.

Solution With N = 10 we have h = 0.2, $t_i = 0.2i, \omega_0 = 0.5$, and

$$\omega_{i+1} = \omega_i + h\left(\omega_i - t_i^2 + 1\right) = \omega_i + 0.2[\omega_i - 0.04i^2 + 1]$$
$$= 1.2\omega_i - 0.008i^2 + 0.2, \qquad \text{for } i = 0,1,\ldots,9.$$

So

$$\omega_1 = 1.2(0.5) - 0.008(0)^2 = 0.8,$$
$$\omega_2 = 1.2(0.8) - 0.008(1)^2 + 0.2 = 1.152,$$

and so on.

Table 5.1 shows the comparison between the approximate values at ti and the actual values

Table 5.1

t_i	w_i	$y_i = y(t_i)$	$\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0.0000000
0.2	0.8000000	0.8292986	0.0292986
0.4	1.1520000	1.2140877	0.0620877
0.6	1.5504000	1.6489406	0.0985406
0.8	1.9884800	2.1272295	0.1387495
1.0	2.4581760	2.6408591	0.1826831
1.2	2.9498112	3.1799415	0.2301303
1.4	3.4517734	3.7324000	0.2806266
1.6	3.9501281	4.2834838	0.3333557
1.8	4.4281538	4.8151763	0.3870225
2.0	4.8657845	5.3054720	0.4396874

Note that the error grows slightly as the value of t increases.

Error Bounds for Euler's Method

To derive an error bound for Euler's method, we need two computational lemmas.

Lemma 5.7 For all $x \geq -1$ and any positive m, we have $0 \leq (1+x)^m \leq e^{mx}$.

Proof Applying Taylor's Theorem with $f(x) = e^x, x_0 = 0$, and n = 1 gives

$$e^x = 1 + x + \frac{1}{2}x^2 e^{\xi}$$

where ξ is between x and zero. Thus

$$0 \leq 1 + x \leq 1 + x + \frac{1}{2}x^2 e^{\xi} = e^x,$$

and, because $1 + x \geq 0$, we have

$$0 \leq (1+x)^m \leq (e^x)^m = e^{mx}$$

Lemma 5.8 If s and t are positive real numbers, $\{a_i\}_{i=0}^{k}$ is a sequence satisfying $a_0 \geq -t/s$, and

$$a_{i+1} \leq (1+s)a_i + t, \qquad \text{for each } i = 0,1,2,\ldots,k-1, \qquad (5.9)$$

then

$$a_{i+1} \leq e^{(i+1)s}\left(a_0 + \frac{t}{s}\right) - \frac{t}{s}$$

Proof For a fixed integer i, Inequality (5.9) implies that

$$a_{i+1} \leq (1+s)a_i + t$$
$$\leq (1+s)[(1+s)a_{i-1} + t] + t = (1+s)^2 a_{i-1} + [1 + (1+s)]t$$

$$\leq (1+s)^3 a_{i-2} + [1 + (1+s) + (1+s)^2]t$$

$$\vdots$$

$$\leq (1+s)^{i+1} a_0 + [1 + (1+s) + (1+s)^2 + \ldots + (1+s)^i]t$$

But

$$1 + (1+s) + (1+s)^2 + \ldots + (1+s)^i = \sum_{j=0}^{i} (1+s)^j$$

is a geometric series with ratio $(1+s)$that sums to

$$\frac{1-(1+s)^{i+1}}{1-(1+s)} = \frac{1}{s}[(1+s)^{i+1} - 1]$$

Thus

$$a_i \le (1+s)^{i+1}a_0 + \frac{(1+s)^{i+1}-1}{s}t = (1+s)^{i+1}(a_0 + \frac{t}{s}) - \frac{t}{s}$$

and using Lemma 5.7 with $x = 1 + s$ gives

$$a_i \le e^{(i+1)s}(a_0 + \frac{t}{s}) - \frac{t}{s}$$

Theorem 5.9 Suppose f is continuous and satisfies a Lipschitz condition with constant L on

$$D = \{(t, y) | a \le t \le b \text{ and } -\infty < y < \infty\}$$

and that a constant M exists with

$$|y''(t)| \le M, \qquad \text{for all } t \in [a, b]$$

where $y(t)$ denotes the unique solution to the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

Let $\omega_0, \omega_1, \ldots, \omega_N$ be the approximations generated by Euler's method for

some positive integer N. Then, for each i = 0, 1, 2, ... , N,

$$|y(t_i) - \omega_i| \leq \frac{hM}{2L}[e^{L(t_i - a)} - 1] \qquad (5.10)$$

Proof When i = 0 the result is clearly true, since $y(t_0) = \omega_0 = \alpha$. From Eq. (5.7), we have

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}y''(\xi_i)$$

for i = 0, 1, ... , N − 1, and from the equations in (5.8)

$$\omega_{i+1} = \omega_i + hf(t_i, \omega_i)$$

Using the notation $y_i = y(t_i)$ and$y_{i+1} = y(t_{i+1})$, we subtract these two equations to obtain

$$y_{i+1} - \omega_{i+1} = y_i - \omega_i + h[f(t_i, y_i) - f(t_i, \omega_i)] + \frac{h^2}{2}y''(\xi_i)$$

Hence

$$|y_{i+1} - \omega_{i+1}| \leq |y_i - \omega_i| + h|f(t_i, y_i) - f(t_i, \omega_i)| + \frac{h^2}{2}|y''(\xi_i)|$$

Now f satisfies a Lipschitz condition in the second variable with constant L, and $|y''(t)| \leq M, so$

$$|y_{i+1} - \omega_{i+1}| \leq (1 + hL)|y_i - \omega_i| + \frac{h^2 M}{2}$$

Referring to Lemma 5.8 and letting $s = hL, t = h^2 M/2$, and $a_j = |y_j - \omega_j|$, for each j = 0, 1, … , N, we see that

$$|y_{i+1} - \omega_{i+1}| \leq e^{(i+1)hL}(|y_0 - \omega_0| + \frac{h^2 M}{2hL}) - \frac{h^2 M}{2hL}$$

Because $|y_0 - \omega_0| = 0$ and $(i + 1)h = t_{i+1} - t_0 = t_{i+1} - a$, this implies that

$$|y_{i+1} - \omega_{i+1}| \leq \frac{hM}{2L}(e^{(t_{i+1}-a)L} - 1) \text{for each i} = 0, 1, \ldots, \text{N} - 1.$$

- The weakness of Theorem 5.9 lies in the requirement that a bound be known for the second derivative of the solution.
- it should be noted that if ∂f/∂t and ∂f/∂y both exist, the chain rule for partial differentiation implies that

$$y''(t) = \frac{dy'}{dt}(t) = \frac{df}{dt}(t, y(t)) = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t))$$

So it is at times possible to obtain an error bound for $y''(t)$ without explicitly knowing $y(t)$.

Example 2 The solution to the initial-value problem

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

was approximated by using Euler's method with h = 0.2. Use the inequality in Theorem 5.9 to find a bounds for the approximation errors and compare these to the actual errors.

Solution Because $f(t, y) = y - t^2 + 1$,we have $\partial f(t, y)/\partial y = 1$ for all y,so L=1.For this problem, the exact solution is $y(t) = (t + 1)^2 - 0.5e^t$,so $y''(t) = 2 - 0.5e^t$ and

$$|y''(t)| \le 0.5e^2 - 2, \qquad \text{for all } t \in [0,2]$$

Using the inequality in the error bound for Euler's method with h = 0.2, L = 1, and M = $0.5e^2 - 2$ gives

$$|y_i - \omega_i| \le 0.1(0.5e^2 - 2)(e^{t_i} - 1)$$

Hence

$$|y(0.2) - \omega_1| \le 0.1(0.5e^2 - 2)(e^{0.2} - 1) = 0.03752$$
$$|y(0.4) - \omega_2| \le 0.1(0.5e^2 - 2)(e^{0.4} - 1) = 0.08334$$

and so on. Table 5.2 lists the actual error found in Example 1, together with this error bound.

Table 5.2

t_i	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Actual Error	0.02930	0.06209	0.09854	0.13875	0.18268	0.23013	0.28063	0.33336	0.38702	0.43969
Error Bound	0.03752	0.08334	0.13931	0.20767	0.29117	0.39315	0.51771	0.66985	0.85568	1.08264

Table 5.2

t_i	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Actual Error	0.02930	0.06209	0.09854	0.13875	0.18268	0.23013	0.28063	0.33336	0.38702	0.43969
Error Bound	0.03752	0.08334	0.13931	0.20767	0.29117	0.39315	0.51771	0.66985	0.85568	1.08264

Note that even though the true bound for the second derivative of the solution was used, the error bound is considerably larger than the actual error, especially for increasing values of t.

The principal importance of the error-bound formula given in Theorem 5.9 is that the bound depends linearly on the step size h. Consequently, diminishing the step size should give correspondingly greater accuracy to the approximations.

- Neglected in the result of Theorem 5.9 is the effect that round-off error plays in the choice of step size.
- As h becomes smaller, more calculations are necessary and more roundoff error is expected.
- In actuality then, the difference-equation form

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + hf(t_i, \omega_i) \end{cases} \quad i = 0,1,\ldots,N-1.$$

We use instead an equation of the form

$$\omega_0 = \alpha + \delta_0$$

$$\omega_{i+1} = \omega_i + hf(t_i, u_i) + \delta_{i+1}, \quad \text{for each } i = 0,1,\ldots,N-1, \quad (5.11)$$

Using methods similar to those in the proof of Theorem 5.9, we can produce an error bound for the finite-digit approximations to y_i given by Euler's method.

Theorem 5.10 Let $y(t)$ denote the unique solution to the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b\,, y(a) = \alpha \qquad (5.12)$$

and $u_0, u_1, \ldots, u_N$ be the approximations obtained using (5.11). If $|\delta_i| < \delta$ for each i = 0, 1, ... , N and the hypotheses of Theorem 5.9 hold for (5.12), then

$$|y(t_i) - u_i| \le \frac{1}{L}\left(\frac{hM}{2} + \frac{\delta}{h}\right)[e^{L(t_i - a)} - 1] + |\delta_0| e^{L(t_i - a)} \qquad (5.13)$$

for each i = 0, 1, ... , N.

The error bound (5.13) is no longer linear in h. In fact, since

$$\lim_{h \to 0}\left(\frac{hM}{2} + \frac{\delta}{h}\right) = \infty$$

Letting $E(h) = (hM/2) + (\delta/h)$ implies that $E'(h) = (M/2) - (\delta/h^2)$

Letting $E(h) = (hM/2) + (\delta/h)$ implies that $E'(h) = (M/2) - (\delta/h^2)$

If $h < \sqrt{2\delta/M}$,then $E'(h) < 0$ and $E(h)$ is decreasing.

If $h > \sqrt{2\delta/M}$,then $E'(h) > 0$ and $E(h)$ is increasing.

The minimal value of E(h) occurs when

$$h = \sqrt{\frac{2\delta}{M}} \qquad (5.14)$$

- Decreasing h beyond this value tends to increase the total error in the approximation.
- Normally, however, the value of δ is sufficiently small that this lower bound for h does not affect the operation of Euler's method.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-3

Higher-Order Taylor Methods

高阶泰勒方法

Introduction

we need a means for comparing the efficiency of various approximation methods. The first device we consider is called **the local truncation error** of the method.

- The local truncation error at a specified step measures the amount by which the exact solution to the differential equation fails to satisfy the difference equation being used for the approximation at that step.

- The local truncation will serve quite well to determine not only the local error of a method but the actual approximation error.

Definition 5.11 The difference method

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + h\phi(t_i, \omega_i), \qquad \text{for each } i = 0,1,\ldots,N-1,$$

has **local truncation error**

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i)$$

for each i = 0, 1, ... , N − 1, where y_i and y_{i+1} denote the solution at t_i and t_{i+1}, respectively.

For example Euler's method has local truncation error at the ith step

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i), \qquad \text{for each } i = 0, 1, \ldots, N-1$$

This error is a local error because it measures the accuracy of the method at a specific step, assuming that the method was exact at the previous step. It depends on the differential equation, the step size, and the particular step in the approximation.

we see that Euler's method has

$$\tau_{i+1}(h) = \frac{h}{2} y''(\xi_i), \qquad \text{for some } \xi_i \in (t_i, t_{i+1})$$

When $y''(t)$ is known to be bounded by a constant M on [a,b],this implies

$$|\tau_{i+1}(h)| \leq \frac{h}{2} M$$

so the local truncation error in Euler's method is O(h).

- One way to select difference-equation methods for solving ordinary differential equations is in such a manner that their local truncation errors are $O(h^p)$ for as large a value of p as possible, while keeping the number and complexity of calculations of the methods within a reasonable bound.
- Suppose the solution $y(t)$ to the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

we expand the solution, $y(t)$, in terms of its nth Taylor polynomial about t_i and evaluate at t_{i+1}, we obtain

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \ldots + \frac{h^n}{n!}y^{(n)}(t_i) + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_i) \qquad \xi_i \in (t_i, t_{i+1}) \quad (5.15).$$

- Successive differentiation of the solution, y(t), gives

$$y'(t) = f(t, y(t)), y''(t) = f'(t, y(t)),$$
$$\text{and, generally, } y^{(k)}(t) = f^{(k-1)}(t, y(t)),$$

Substituting these results into Eq. (5.15) gives

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} f'(t_i, y(t_i)) + \cdots$$
$$+ \frac{h^n}{n!} f(t_i, y(t_i)) + \frac{h^{n+1}}{(n+1)!} f^{(n)}(\xi_i, y(\xi_i)) \quad (5.16)$$

The difference-equation method corresponding to Eq. (5.16) is obtained by deleting the remainder term involving ξ_i.

Taylor method of order n

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + hT^{(n)}(t_i, \omega_i), \qquad \text{for each } i = 0,1,\ldots,N-1,$$

where

$$T^{(n)}(t_i, \omega_i) = f(t_i, \omega_i) + \frac{h}{2} f'(t_i, \omega_i) + \ldots + \frac{h^{n-1}}{n!} f^{(n-1)}(t_i, \omega_i)$$

Euler's method is Taylor's method of order one.

Example 1 Apply Taylor's method of orders (a) two and (b) four with N = 10 to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

Theorem 5.12 If Taylor's method of order n is used to approximate the solution to

$$y'(t) = f(t, y(t)), a \le t \le b, \qquad y(a) = \alpha$$

with step size h and if $y \in C^{n+1}[a, b]$, then the local truncation error is $O(h^n)$.

Proof Note that Eq. (5.16) on page 277 can be rewritten

$$y_{i+1} - y_i - hf(t_i, y_i) - \frac{h^2}{2} f'(t_i, y_i) - \ldots - \frac{h^n}{n!} f^{(n-1)}(t_i y_i)$$

$$= \frac{h^{n+1}}{(n+1)!} f^{(n)}(\xi_i, y(\xi_i))$$

for some ξ_i in (t_i, t_{i+1}). So the local truncation error is

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - T^{(n)}(t_i, y_i) = \frac{h^n}{(n+1)!} f^{(n)}(\xi_i, y(\xi_i))$$

Since $y \in C^{n+1}[a,b]$, we have $y^{(n+1)}(t) = f^{(n)}(t, y(t))$ bounded on $[a,b]$ and

$\tau_i(h) = O(h^n)$, for each i=1,2,...,N

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-4

Runge-Kutta Methods

Introduction

- **Higher-Order Taylor Methods**

1. The Taylor methods have the desirable property of highorder local truncation error.

2. The disadvantage of requiring the computation and evaluation of the derivatives of $f(t, y)$.

3. The Taylor methods are seldom used in practice.

Example

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \quad 0 \le x \le 1,\ h = 0.1$$

- **Runge-Kutta Methods**

1. Runge-Kutta methods have the high-order local truncation error of the Taylor methods.

2. Runge-Kutta methods eliminate the need to compute and evaluate the derivatives of $f(t, y)$.

Theorem 5.13 Suppose that $f(t,y)$ and all its partial derivatives of order less than or equal to $n+1$ are continuous on $D=\{(t,y)|a\le t\le b, c\le y\le d\}$, and let $(t_0,y_0)\in D$. For every $(t,y)\in D$, there exists ξ between t and t_0 and μ between y and y_0 with

$$f(t,y)=P_n(t,y)+R_n(t,y)$$

where

$$P_n(t,y)=f(t_0,y_0)+[(t-t_0)\frac{\partial f}{\partial t}(t_0,y_0)+(y-y_0)\frac{\partial f}{\partial y}(t_0,y_0)]+[\frac{(t-t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0,y_0)+(t-t_0)(y-y_0)\frac{\partial^2 f}{\partial t\partial y}(t_0,y_0)$$

$$+\frac{(y-y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0,y_0)]+\cdots+[\frac{1}{n!}\sum_{j=0}^{n}C_n^j(t-t_0)^{n-j}(y-y_0)^j\frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0)]$$

and

$$R_n(t,y) = \frac{1}{(n+1)!}\sum_{j=0}^{n+1}\binom{n+1}{j}(t-t_0)^{n+1-j}(y-y_0)^j\frac{\partial^{n+1}f}{\partial t^{n+1-j}\partial y^j}(\xi,\mu)$$

The function $P_n(t,y)$ is called the n**th Taylor polynomial in two variables** for the function f about(t_0,y_0), and $R_n(t,y)$ is the remainder term associated with $P_n(t,y)$.

Taylor method of order n

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + hT^{(n)}(t_i, \omega_i), \qquad \text{for each } i = 0,1,\ldots,N-1,$$

where

$$T^{(n)}(t_i, \omega_i) = f(t_i, \omega_i) + \frac{h}{2} f'(t_i, \omega_i) + \ldots + \frac{h^{n-1}}{n!} f^{(n-1)}(t_i, \omega_i)$$

Euler's method is Taylor's method of order one.

Runge-Kutta Methods of Order Two

The first step in deriving a Runge-Kutta method is to determine values for a_1, α_1, and β_1 with the property that $a_1 f(t+\alpha_1, y+\beta_1)$ approximates

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y)$$

with error no greater than $O(h^2)$, which is same as the order of the local truncation error for the Taylor method of order two.

$$f'(t,y) = \frac{df}{dt}(t,y) = \frac{\partial f}{\partial t}(t,y) + \frac{\partial f}{\partial y}(t,y) \cdot y'(t) \quad and \quad y'(t) = f(t,y)$$

we have

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t}(t,y) + \frac{h}{2}\frac{\partial f}{\partial y}(t,y)\cdot f(t,y) \qquad (5.18)$$

Expanding $f(t+\alpha_1, y+\beta_1)$ in its Taylor polynomial of degree one about (t,y) gives

$$a_1 f(t+\alpha_1, y+\beta_1) = a_1 f(t,y) + a_1\alpha_1 \frac{\partial f}{\partial t}(t,y) + a_1\beta_1 \frac{\partial f}{\partial y}(t,y) + a_1 R_1(t+\alpha_1, y+\beta_1) \qquad (5.19)$$

where

$$R_1(t+\alpha_1, y+\beta_1) = \frac{\alpha_1^2}{2}\frac{\partial^2 f}{\partial t^2}(\xi,\mu) + \alpha_1\beta_1 \frac{\partial^2 f}{\partial t \partial y}(\xi,\mu) + \frac{\beta_1^2}{2}\frac{\partial^2 f}{\partial y^2}(\xi,\mu) \qquad (5.20)$$

for some ξ between t and $t+\alpha_1$ and μ between y and $y+\beta_1$.

also

$$f(t,y): a_1 = 1; \quad \frac{\partial f}{\partial t}(t,y): a_1\alpha_1 = \frac{h}{2}; \quad \frac{\partial f}{\partial y}(t,y): a_1\beta_1 = \frac{h}{2}f(t,y)$$

The parameters a_1,α_1, and β_1 are therefore

$$a_1 = 1; \quad \alpha_1 = \frac{h}{2}; \quad \beta_1 = \frac{h}{2}f(t,y)$$

so

$$T^{(2)}(t,y) = f(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)) - R_1(t+\frac{h}{2}, y+\frac{h}{2}f(t,y))$$

and from Eq. (5.20)

$$R_1(t+\frac{h}{2}, y+\frac{h}{2}f(t,y))$$
$$= \frac{h^2}{8}\frac{\partial^2 f}{\partial t^2}(\xi,\mu) + \frac{h^2}{4}f(t,y)\frac{\partial^2 f}{\partial t\partial y}(\xi,\mu) + \frac{h^2}{8}(f(t,y))^2\frac{\partial^2 f}{\partial y^2}(\xi,\mu)$$

If all the second-order partial derivatives of f are bounded, then

$$R_1\left(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)\right)$$

is $O(h^2)$. As a consequence:

- The order of error for this new method is the same as that of the Taylor method of order two.

The difference-equation method resulting from replacing $T^{(2)}(t,y)$ in Taylor's method of order two by $f(t+(h/2), y+(h/2)f(t,y))$ is a specific Runge-Kutta method known as the *Midpoint method*.

Midpoint Method

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + hf(t_i + \frac{h}{2}, \omega_i + \frac{h}{2} f(t_i, \omega_i)), \qquad for\ i = 0,1,\ldots,N-1$$

a more complicated form is required to satisfy the conditions for any of the higher-order Taylor methods.

The most appropriate four-parameter form for approximating

$$T^{(3)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y) + \frac{h^2}{6} f''(t,y)$$

$$a_1 f(t,y) + a_2\, f(t + \alpha_2, y + \delta_2 f(t,y)); \qquad (5.21)$$

and even with this, there is insufficient flexibility to match the term

and even with this, there is insufficient flexibility to match the term

$$\frac{h^2}{6}\left[\frac{\partial f}{\partial y}(t,y)\right]^2 f(t,y)$$

resulting from the expansion of $(h^2/6)f''(t,y)$.

(5.21) has four parameters, so it gives a flexibility in their choice, so a number of $O(h^2)$ methods can be derived. The most important is **Modified Euler method**, which corresponds to choosing $a_1 = a_2 = \frac{1}{2}$ and $\alpha_2 = \delta_2 = h$. It has the following difference-equation form.

Modified Euler Method

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + \frac{h}{2}[f(t_i, \omega_i) + f(t_{i+1}, \omega_i + hf(t_i, \omega_i))], \qquad for\ i = 0,1,\ldots,N-1$$

Example 2 Use the Midpoint method and the Modified Euler method with $N = 10, h = 0.2, t_i = 0.2i$ and $\omega_0 = 0.5$ to approximate the solution to our usual example,

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

Solution: The difference equations produced from the various formulas are

Midpoint method: $\omega_{i+1} = 1.22\omega_i - 0.0088i^2 - 0.008i + 0.218$

Modified Euler method: $\omega_{i+1} = 1.22\omega_i - 0.0088i^2 - 0.008i + 0.216$

Continued

The first two steps of these methods give

Midpoint method: $\omega_1 = 1.22(0.5) - 0.0088(0)^2 - 0.008(0) + 0.218 = 0.828$

Modified Euler method: $\omega_1 = 1.22(0.5) - 0.0088(0)^2 - 0.008(0) + 0.216 = 0.826$

and

Midpoint method: $\omega_2 = 1.22(0.828) - 0.0088(0.2)^2 - 0.008(0.2) + 0.218 = 1.21136$

Modified Euler method: $\omega_1 = 1.22(0.826) - 0.0088(0.2)^2 - 0.008(0.2) + 0.216 = 1.20692$

Table 5.6 lists all the results of the calculations

Table 5.6

t_i	$y(t_i)$	Midpoint Method	Error	Modified Euler Method	Error
0.0	0.5000000	0.5000000	0	0.5000000	0
0.2	0.8292986	0.8280000	0.0012986	0.8260000	0.0032986
0.4	1.2140877	1.2113600	0.0027277	1.2069200	0.0071677
0.6	1.6489406	1.6446592	0.0042814	1.6372424	0.0116982
0.8	2.1272295	2.1212842	0.0059453	2.1102357	0.0169938
1.0	2.6408591	2.6331668	0.0076923	2.6176876	0.0231715
1.2	3.1799415	3.1704634	0.0094781	3.1495789	0.0303627
1.4	3.7324000	3.7211654	0.0112346	3.6936862	0.0387138
1.6	4.2834838	4.2706218	0.0128620	4.2350972	0.0483866
1.8	4.8151763	4.8009586	0.0142177	4.7556185	0.0595577
2.0	5.3054720	5.2903695	0.0151025	5.2330546	0.0724173

Runge-Kutta methods are also options within theMaple command *InitialValueProblem*.

Higher-Order Runge-Kutta Methods

The term $T^{(3)}(t,y)$ can be approximated with error $O(h^3)$ by an expression of the form

$$f(\mathrm{t}+\alpha_1, \mathrm{y}+\delta_1 f(\mathrm{t}+\alpha_2, \mathrm{y}+\delta_2 f(t,y)))$$

The most common $O(h^3)$ is Heun's method, given by

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + \frac{h}{4}\left(f(t_i,\omega_i) + 3f\left(t_i + \frac{2h}{3}, \omega_i + \frac{2h}{3} f\left(t_i + \frac{h}{3}, \omega_i + \frac{h}{3} f(t_i,\omega_i)\right)\right)\right),$$

$$for\ i = 0,1,\ldots,N-1$$

Illustration

Applying Heun's method with $N = 10, h = 0.2, t_i = 0.2i \; and \; \omega_0 = 0.5$ to approximate the solution to our usual example,

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

gives the values in Table 5.7

Table 5.7

t_i	$y(t_i)$	Heun's Method	Error
0.0	0.5000000	0.5000000	0
0.2	0.8292986	0.8292444	0.0000542
0.4	1.2140877	1.2139750	0.0001127
0.6	1.6489406	1.6487659	0.0001747
0.8	2.1272295	2.1269905	0.0002390
1.0	2.6408591	2.6405555	0.0003035
1.2	3.1799415	3.1795763	0.0003653
1.4	3.7324000	3.7319803	0.0004197
1.6	4.2834838	4.2830230	0.0004608
1.8	4.8151763	4.8146966	0.0004797
2.0	5.3054720	5.3050072	0.0004648

- The most common RungeKutta method in use is of order four in difference-equation form

Runge-Kutta Order Four

$$\omega_0 = \alpha$$
$$k_1 = hf(t_i, \omega_i)$$
$$k_2 = hf(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1)$$
$$k_3 = hf(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_2)$$
$$k_4 = hf(t_{i+1}, \omega_i + k_3)$$
$$\omega_{i+1} = \omega_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

This method has local truncation error $O(h^4)$, provided the solution $y(t)$ has five continuous derivatives.

ALGORITHM 5.2 Runge-Kutta (Order Four)

To approximate the solution of the initial-value problem

$$y' = f(t, y), a \le t \le b, \qquad y(a) = \alpha$$

at $(N + 1)$ equally spaced numbers in the interval $[a, b]$:

INPUT *endpoints a, b; integer N; initial condition α.*

OUTPUT *approximation ω to y at the (N + 1) values of t.*

Step 1 *Set h= (b-a)/N;*

$t = a.$

$\omega = \alpha$

OUTPUT (t,ω)

Step 2 For *i=1,2,...,N do Steps 3-5.*

Step 3 *Set* $K_1 = hf(t, \omega)$

$$K_2 = hf(t + h/2, \omega + K_1/2);$$

$$K_3 = hf(t + h/2, \omega + K_2/2);$$

$$K_4 = hf(t + h/2, \omega + K_3);$$

Step 4 *Set* $\omega = \omega + (K_1 + 2K_2 + 2K_3 + K_4)/6; (Compute\ \omega_i.)$

$$t = a + ih(Compute\ t_i.)$$

Step 5 OUTPUT (t,ω)

Step 6 *STOP.*

Example 3 Use the Runge-Kutta method of order four with $N = 10, h = 0.2, t_i = 0.2i$ to obtain approximations to the solution of the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

Solution: The approximation to y(0.2) is obtained by

$$\omega_0 = 0.5$$

$$k_1 = 0.2f(0,0.5) = 0.2(1.5) = 0.3$$

$$k_2 = 0.2f(0.1,0.65) = 0.328$$

$$k_3 = 0.2f(0.1,0.664) = 0.3308$$

$$k_4 = 0.2f(0.2,0.8308) = 0.35816$$

$$\omega_1 = 0.5 + \frac{1}{6}(0.3 + 2(0.328) + 2(0.3308) + 0.35816) = 0.8292933$$

The remaining results and their errors are listed in Table 5.8.

Table 5.8

t_i	Exact $y_i = y(t_i)$	Runge-Kutta Order Four w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272027	0.0000269
1.0	2.6408591	2.6408227	0.0000364
1.2	3.1799415	3.1798942	0.0000474
1.4	3.7324000	3.7323401	0.0000599
1.6	4.2834838	4.2834095	0.0000743
1.8	4.8151763	4.8150857	0.0000906
2.0	5.3054720	5.3053630	0.0001089

- To obtain Runge-Kutta order 4 method results with *InitialValueProblem* use the option

method = rungekutta, submethod = rk4.

C := InitialValueProblem(deq, y(0) = 0.5, t = 2, method =rungekutta,submethod =rk4, numsteps = 10, output = information, digits = 8)

Computational Comparisons

The main computational effort in applying the Runge-Kutta methods is the evaluation of f .

- In the second-order methods, the local truncation error is $O(h^2)$, and the cost is two function evaluations per step.
- The Runge-Kutta method of order four requires 4 evaluations per step, and the local truncation error is $O(h^4)$.
- Butcher has established the relationship between the number of evaluations per step and the order of the local truncation error shown in Table 5.9.

Table 5.9

Evaluations per step	2	3	4	$5 \le n \le 7$	$8 \le n \le 9$	$10 \le n$
Best possible local truncation error	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^{n-1})$	$O(h^{n-2})$	$O(h^{n-3})$

One measure of comparing the lower-order Runge-Kutta methods is described as follows:

The Runge-Kutta method of order four requires four evaluations per step, whereas Euler's method requires only one evaluation. Hence if the Runge-Kutta method of order four is to be superior it should give more accurate answers than Euler's method with one-fourth the step size. Similarly, if the Runge-Kutta method of order four is to be superior to the second-order Runge-Kutta methods, which require two evaluations per step, it should give more accuracy with step size h than a second-order method with step size h/2.

Illustration

For the problem

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

- Euler's method with h = 0.025, the Midpoint method with h = 0.05, and the RungeKutta fourth-order method with h = 0.1 are compared at the common mesh points of these methods 0.1, 0.2, 0.3, 0.4, and 0.5.

Table 5.10

t_i	Exact	Euler $h = 0.025$	Modified Euler $h = 0.05$	Runge-Kutta Order Four $h = 0.1$
0.0	0.5000000	0.5000000	0.5000000	0.5000000
0.1	0.6574145	0.6554982	0.6573085	0.6574144
0.2	0.8292986	0.8253385	0.8290778	0.8292983
0.3	1.0150706	1.0089334	1.0147254	1.0150701
0.4	1.2140877	1.2056345	1.2136079	1.2140869
0.5	1.4256394	1.4147264	1.4250141	1.4256384

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-5

Error Control and the Runge-Kutta-Fehlberg Method

Introduction

The appropriate use of varying step sizes for integral approximations produced efficient methods.

1. They have a feature that makes them worthwhile. They incorporate in the step-size procedure an estimate of the truncation error that does not require the approximation of the higher derivatives of the function. These methods are called adaptive

2. There are adaptive methods for approximating the solutions to initial-value problems and that these methods are not only efficient, but also incorporate the control of error.

Any one-step method for approximating the solution, y(t), of the initial-value problem

$$y' = f(t, y), \quad \text{for } a \le t \le b, \quad \text{with } y(a) = \alpha$$

can be expressed in the form

$$\omega_{i+1} = \omega_i + h_i \phi(t_i, \omega_i, h_i), \qquad \text{for } i = 0,1,\ldots,N-1$$

An ideal difference-equation method

$$\omega_{i+1} = \omega_i + h_i \phi(t_i, \omega_i, h_i), \qquad \text{for } i = 0,1,\ldots,N-1$$

for approximating the solution, y(t), to the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad \text{with } y(a) = \alpha$$

would have the property that, given a tolerance ε > 0, a minimal number of mesh points could be used to ensure that the global error, $|y(t_i) - \omega_i|$, did not exceed ε for any i =0, 1, ... , N.

- In this section we examine techniques used to control the error of a difference equation method in an efficient manner by the appropriate choice of mesh points.

- By using methods of differing order we can predict the local truncation error and, using this prediction, choose a step size that will keep it and the global error in check.

To illustrate the technique, suppose that we have two approximation techniques.

The first techniques

The first is obtained from an nth-order Taylor method of the form

$$y(t_{i+1}) = y(t_i) + h\phi(t_i, y(t_i), h) + O(h^{n+1})$$

and produces approximations with local truncation error $\tau_{i+1}(h) = O(h^n)$.It is given by

$$\omega_0 = \alpha$$

$$\omega_{i+1} = \omega_i + h\phi(t_i, \omega_i, h), \qquad \text{for } i > 0$$

The second techniques

It comes from an (n + 1)st-order Taylor method of the form

$$y(t_{i+1}) = y(t_i) + h_i\tilde{\phi}(t_i, y(t_i), h) + O(h^{n+2})$$

and produces approximations with local truncation error $\tilde{\phi}_{i+1}(h) = O(h^{n+1})$.It is given by

$$\omega_0 = \alpha$$

$$\tilde{\omega}_{i+1} = \tilde{\omega}_i + h\tilde{\phi}(t_i, \tilde{\omega}_i, h), \qquad \text{for } i > 0$$

We first make the assumption that $\omega_i \approx y(t_i) \approx \tilde{\omega}_i$ and choose a fixed step size h to generate the approximations ω_{i+1} and $\tilde{\omega}_{i+1}$ to $y(t_{i+1})$

Then

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h)$$
$$= \frac{y(t_{i+1}) - \omega_i}{h} - \phi(t_i, \omega_i, h)$$
$$= \frac{y(t_{i+1}) - [\omega_i + h\phi(t_i, \omega_i, h)]}{h}$$
$$= \frac{1}{h}(y(t_{i+1}) - \omega_{i+1})$$

In a similar manner, we have

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{\omega}_{i+1})$$

As a consequence, we have

$$\tau_{i+1}(h) = \frac{1}{h}(\mathrm{y}(t_{i+1}) - \omega_{i+1})$$

$$= \frac{1}{h}[(\mathrm{y}(t_{i+1}) - \tilde{\omega}_{i+1}) + (\tilde{\omega}_{i+1} - \omega_{i+1})$$

$$= \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{\omega}_{i+1} - \omega_{i+1})$$

But $\tau_{i+1}(h)$ is $O(h^n)$ and $\tilde{\tau}_{i+1}(h)$ is $O(h^{n+1})$, so the significant portion of $\tau_{i+1}(h)$ must come from

$$\frac{1}{h}(\tilde{\omega}_{i+1} - \omega_{i+1})$$

This gives us an easily computed approximation for the local truncation error of the $O(h^n)$ method:

$$\tau_{i+1}(h) \approx \frac{1}{h}(\tilde{\omega}_{i+1} - \omega_{i+1})$$

we now assume that since $\tau_{i+1}(h)$ is $O(h^n)$, a number K, independent of h, exists with

$$\tau_{i+1}(h) \approx Kh^n$$

Then

$$\tau_{i+1}(qh) \approx K(qh)^n = q^n(Kh^n) \approx q^n\tau_{i+1}(h) \approx \frac{q^n}{h}(\widetilde{\omega}_{i+1} - \omega_{i+1})$$

To bound $\tau_{i+1}(qh)$ by ε, we choose q so that

$$\frac{q^n}{h}|\widetilde{\omega}_{i+1} - \omega_{i+1}| \approx |\tau_{i+1}(qh)| \leq \varepsilon;$$

that is, so that

$$q \leq \left(\frac{\varepsilon h}{|\widetilde{\omega}_{i+1} - \omega_{i+1}|}\right)^{1/n} \qquad (5.22)$$

Runge-Kutta-Fehlberg Method(费尔伯格)

- One popular technique that uses Inequality (5.22) for error control is the **Runge-KuttaFehlberg method**
- This technique uses a Runge-Kutta method with local truncation error of order five

$$\tilde{\omega}_{i+1} = \omega_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6$$

to estimate the local error in a Runge-Kutta method of order four given by

$$\omega_{i+1} = \omega_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5$$

where the coefficient equations are

$$k_1 = hf(t_i, \omega_i)$$

$$k_2 = hf(t_i + \frac{h}{4}, \omega_i + \frac{1}{4}k_1)$$

$$k_3 = hf(t_i + \frac{3h}{8}, \omega_i + \frac{3}{32}k_1 + \frac{9}{32}k_2)$$

$$k_4 = hf(t_i + \frac{12h}{13}, \omega_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3)$$

$$k_5 = hf(t_i + h, \omega_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4)$$

$$k_6 = hf(t_i + \frac{h}{2}, \omega_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5)$$

advantage

1. An advantage to this method is that only six evaluations of f are required per step.
2. the Runge-Kutta Fehlberg method has at least a 40% decrease in the number of function evaluations over the use of a pair of arbitrary fourth- and fifth-order methods.

In the error-control theory, an initial value of h at the ith step is used to find the first values of ω_{i+1} and $\tilde{\omega}_{i+1}$, which leads to the determination of q for that step, and then the calculations were repeated.
The value of q determined at the ith step is used for two purposes:

The value of q determined at the ith step is used for two purposes:

- When $q < 1$: to reject the initial choice of h at the ith step and repeat the calculations using qh, and
- When $q \geq 1$: to accept the computed value at the ith step using the step size h, but change the step size to qh for the (i + 1)st step.

In fact, for the Runge-Kutta-Fehlberg method with n = 4, a common choice is

$$q \leq \left(\frac{\varepsilon h}{2|\tilde{\omega}_{i+1} - \omega_{i+1}|}\right)^{1/4} = 0.84\left(\frac{\varepsilon h}{|\tilde{\omega}_{i+1} - \omega_{i+1}|}\right)^{1/4}$$

➢ In Algorithm 5.3 for the Runge-Kutta-Fehlberg method, Step 9 is added to eliminate large modifications in step size.

ALGORITHM 5.3 Runge-Kutta-Fehlberg

To approximate the solution of the initial-value problem

$$y' = f(t, y), a \leq t \leq b, \qquad y(a) = \alpha$$

with local truncation error within a given tolerance:

INPUT *endpoints a, b; initial condition α.tolerance TOL; maximum step size hmax;minimum step size hmin.*

OUTPUT t, ω, h *where ω approximates y(t) and the step size h was used, or a message that the minimum step size was exceeded.*

Step 1 *Set* $t = a$;

$$\omega = \alpha$$

h=hmax;

FLAG=1;

OUTPUT (t,ω)

Step 2 *While (FLAG = 1) do Steps 3–11.*

Step 3 *Set* $K_1 = hf(t, \omega)$

$$K_2 = hf(t_i + \frac{h}{4}, \omega + \frac{1}{4}K_1)$$

$$K_3 = hf(T + \frac{3h}{8}, \omega + \frac{3}{32}K_1 + \frac{9}{32}K_2)$$

$$K_4 = hf(t + \frac{12h}{13}, \omega + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3)$$

$$K_5 = hf(t+h, \omega + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4)$$

$$K_6 = hf(t+\frac{h}{2}, \omega - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - K_5)$$

Step 4 *Set* $R = \frac{1}{h}\left|\frac{1}{360}K_1 - \frac{128}{4275}K_3 - \frac{2197}{75240}K_4 + \frac{1}{50}K_5 + \frac{2}{55}K_6\right|$

(Note: $R = \frac{1}{h}|\tilde{\omega}_{i+1} - \omega_{i+1}|$. *)*

Step 5 *If* $R \leq TOL$ *then do Steps 6 and 7.*

Step 6 *Set* $t = t+h$*;(Approximation accepted.)*

$$\omega = \omega + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5$$

Step 7 OUTPUT (t, ω, h)

Step 8 *Set* $\delta = 0.84(TOL/R)^{1/4}$

Step 9 *If* $\delta < 0.1$ *then set h=0.1h*

else if $\delta \geq 4$ *then set h=4h*

else set h=δh. *(Calculate new h.)*

Step 10 *If* $h > hmax$ *then set h=hmax*

Step 11 *If* $\mathbf{t} \geq \mathbf{b}$ *then set FLAG=0*

else if $t + h > b$ *then set* $h = b - t$

else if $h < hmin$ *then*

set FLAG = 0;

OUTPUT ('minimum h exceeded').

(Procedure completed unsuccessfully.)

Step 12 *(The procedure is complete.)*

STOP.

Example 1 Use the Runge-Kutta-Fehlberg method with a tolerance TOL = 10^{-5}, a maximum step size hmax = 0.25, and a minimum step size hmin = 0.01 to approximate the solution to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

and compare the results with the exact solution $y(t) = (t + 1)^2 - 0.5e^t$

Solution We will work through the first step of the calculations and then apply Algorithm 5.3 to determine the remaining results. The initial condition gives $t_0 = 0$ and $w_0 = 0.5$. To determine w_1 using $h = 0.25$, the maximum allowable stepsize, we compute

$$k_1 = hf(t_0, w_0) = 0.25\left(0.5 - 0^2 + 1\right) = 0.375;$$

$$k_2 = hf\left(t_0 + \frac{1}{4}h, w_0 + \frac{1}{4}k_1\right) = 0.25\left(\frac{1}{4}0.25, 0.5 + \frac{1}{4}0.375\right) = 0.3974609;$$

$$k_3 = hf\left(t_0 + \frac{3}{8}h, w_0 + \frac{3}{32}k_1 + \frac{9}{32}k_2\right)$$

$$= 0.25\left(0.09375, 0.5 + \frac{3}{32}0.375 + \frac{9}{32}0.3974609\right) = 0.4095383;$$

$$k_4 = hf\left(t_0 + \frac{12}{13}h, w_0 + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right)$$

$$= 0.25\left(0.2307692, 0.5 + \frac{1932}{2197}0.375 - \frac{7200}{2197}0.3974609 + \frac{7296}{2197}0.4095383\right)$$

$$= 0.4584971;$$

$$k_5 = hf\left(t_0 + h, w_0 + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right)$$

$$= 0.25\left(0.25, 0.5 + \frac{439}{216}0.375 - 8(0.3974609) + \frac{3680}{513}0.4095383 - \frac{845}{4104}0.4584971\right)$$

$$= 0.4658452;$$

$$k_6 = hf\left(t_0 + \frac{1}{2}h, w_0 - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right)$$

$$= 0.25\left(0.125, 0.5 - \frac{8}{27}0.375 + 2(0.3974609) - \frac{3544}{2565}0.4095383\right.$$

$$\left. + \frac{1859}{4104}0.4584971 - \frac{11}{40}0.4658452\right)$$

$$= 0.4204789.$$

The two approximations to $y(0.25)$ are then found to be

$$\tilde{w}_1 = w_0 + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6$$

$$= 0.5 + \frac{16}{135}0.375 + \frac{6656}{12825}0.4095383 + \frac{28561}{56430}0.4584971 - \frac{9}{50}0.4658452$$

$$+ \frac{2}{55}0.4204789$$

$$= 0.9204870,$$

and

$$
\begin{aligned}
w_1 &= w_0 + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \\
&= 0.5 + \frac{25}{216}0.375 + \frac{1408}{2565}0.4095383 + \frac{2197}{4104}0.4584971 - \frac{1}{5}0.4658452 \\
&= 0.9204886.
\end{aligned}
$$

This also implies that

$$
\begin{aligned}
R &= \frac{1}{0.25}\left|\frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right| \\
&= 4\left|\frac{1}{360}0.375 - \frac{128}{4275}0.4095383 - \frac{2197}{75240}0.4584971 + \frac{1}{50}0.4658452 + \frac{2}{55}0.4204789\right| \\
&= 0.00000621388,
\end{aligned}
$$

$$q = 0.84\left(\frac{\varepsilon}{R}\right)^{1/4} = 0.84\left(\frac{0.00001}{0.00000621388}\right)^{1/4} = 0.9461033291.$$

Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25)$ but we should adjust the step size for the next iteration to $h = 0.9461033291(0.25) \approx 0.2365258$. However, only the leading 5 digits of this result would be expected to be accurate because R has only about 5 digits of accuracy. Because we are effectively subtracting the nearly equal numbers w_i and $\tilde{w}_i$ when we compute R, there is a good likelihood of round-off error. This is an additional reason for being conservative when computing q.

The results from the algorithm are shown in Table 5.11.

The last two columns in Table 5.11 show the results of the fifth-order method.

Table 5.11

		RKF-4				RKF-5	
t_i	$y_i = y(t_i)$	w_i	h_i	R_i	$\lvert y_i - w_i \rvert$	$\hat{w}_i$	$\lvert y_i - \hat{w}_i \rvert$
0	0.5	0.5			0.5		
0.2500000	0.9204873	0.9204886	0.2500000	6.2×10^{-6}	1.3×10^{-6}	0.9204870	2.424×10^{-7}
0.4865522	1.3964884	1.3964910	0.2365522	4.5×10^{-6}	2.6×10^{-6}	1.3964900	1.510×10^{-6}
0.7293332	1.9537446	1.9537488	0.2427810	4.3×10^{-6}	4.2×10^{-6}	1.9537477	3.136×10^{-6}
0.9793332	2.5864198	2.5864260	0.2500000	3.8×10^{-6}	6.2×10^{-6}	2.5864251	5.242×10^{-6}
1.2293332	3.2604520	3.2604605	0.2500000	2.4×10^{-6}	8.5×10^{-6}	3.2604599	7.895×10^{-6}
1.4793332	3.9520844	3.9520955	0.2500000	7×10^{-7}	1.11×10^{-5}	3.9520954	1.096×10^{-5}
1.7293332	4.6308127	4.6308268	0.2500000	1.5×10^{-6}	1.41×10^{-5}	4.6308272	1.446×10^{-5}
1.9793332	5.2574687	5.2574861	0.2500000	4.3×10^{-6}	1.73×10^{-5}	5.2574871	1.839×10^{-5}
2.0000000	5.3054720	5.3054896	0.0206668		1.77×10^{-5}	5.3054896	1.768×10^{-5}

- For small values of t, the error is less than the error in the fourth-order method.
- The error exceeds that of the fourth-order method when t increases.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-6

Multistep Methods
多步方法

Introduction

One-step methods: the approximation for the mesh point t_{i+1} involves information from only one of the previous mesh points, t_i.

- All the information used by these methods is obtained within the subinterval over which the solution is being approximated.
- The approximate solution is available at each of the mesh points $t_0, t_1, \ldots, t_i$ before the approximation at t_{i+1} is obtained.

Multistep methods:Methods using the approximation at more than one previous mesh point to determine the approximation at the next point . .

Definition 5.14 An **m-step multistep method** for solving the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

has a difference equation for finding the approximation ω_{i+1} at the mesh point t_{i+1} represented by the following equation, where m is an integer greater than 1:

$$\begin{aligned}&\omega_{i+1}\\&= a_{m-1}\omega_i + a_{m-2}\omega_{i-1} + \ldots + a_0\omega_{i+1-m} + h[b_m f(t_{i+1}, \omega_{i+1})\\&+ b_{m-1} f(t_i, \omega_i) + \ldots + b_0 f(t_{i+1-m}, \omega_{i+1-m})] \qquad (5.24)\end{aligned}$$

for $i = m-1, m, \ldots, N-1$, where $h = (b-a)/N$, the $a_0, a_1, \ldots, a_{m-1}$ and $b_0, b_1, .., b_m$ are constants, and the starting values

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2, \qquad \ldots, \qquad \omega_{m-1} = \alpha_{m-1}$$

are specified.

Relevant contents

- When $b_m = 0$ the method is called **explicit**, or **open**, because Eq. (5.24) then gives ω_{i+1} explicitly in terms of previously determined values.
- When $b_m \neq 0$ the method is called **implicit**, or **closed**, because ω_{i+1} occurs on both sides of Eq. (5.243), so ω_{i+1} is specified only implicitly.

For example

the equations $\omega_0 = \alpha,\ \ \omega_1 = \alpha_1,\ \ \omega_2 = \alpha_2,\ \omega_3 = \alpha_3,$

$$\omega_{i+1} = \omega_i + \frac{h}{24}[55f(t_i,\omega_i) - 59f(t_{i-1},\omega_{i-1}) + 37f(t_{i-2},\omega_{i-2}) - 9f(t_{i-3},\omega_{i-3})] \qquad (5.25)$$

for each $i = 3,4,\ldots,N-1$, define an *explicit* four-step method known as the **fourth-order Adams-Bashforth technique.**

For example

the equations

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2,$$

$$\omega_{i+1} = \omega_i + \frac{h}{24}[9f(t_{i+1}, \omega_{i+1}) + 19f(t_i, \omega_i) - 5f(t_{i-1}, \omega_{i-1}) + f(t_{i-2}, \omega_{i-2})] \qquad (5.26)$$

for each $i = 2,3,\ldots,N-1$, define an *implicit* three-step method known as the **fourth-order Adams-Moulton technique.**

Similarities and differences

- The starting values in either (5.25) or (5.26) must be specified, generally by assuming $\omega_0 = \alpha$ and generating the remaining values by either a Runge-Kutta or Taylor method.

- the implicit methods are generally more accurate then the explicit methods.

- To apply an implicit method such as (5.25) directly, we must solve the implicit equation for ω_{i+1}. This is not always possible.

Example 1 In Example 3 of Section 5.4 (see Table 5.8 on page 289) we used the Runge-Kutta method of order four with h = 0.2 to approximate the solutions to the initial value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

The first four approximations were found to be $y(0) = \omega_0 = 0.5$, $y(0.2) \approx \omega_1 = 0.8292933$, $y(0.4) \approx \omega_2 = 1.2140762$, and $y(0.6) \approx \omega_3 = 1.6489220$. Use these as starting values for the fourth-order Adams-Bashforth method to compute new approximations for $y(0.8)$ and $y(1.0)$, and compare these new approximations to those produced by the Runge-Kutta method of order four.

Solution For the fourth-order Adams-Bashforth we have

$$y(0.8) \approx \omega_4$$
$$= \omega_3 + \frac{0.2}{24}(55f(0.6, \omega_3) - 59f(0.4, \omega_2) + 37f(0.2, \omega_1) - 9f(0, \omega_0))$$
$$= 1.6489220 + \frac{0.2}{24}(55f(0.6,1.6489220) - 59f(0.4,1.2140762)$$
$$+ 37f(0.2,0.8292933) - 9f(0,0.5))$$
$$= 1.6489220 + 0.0083333(55(2.2889220) - 59(2.0540762)$$
$$+ 37(1.7892933) - 9(1.5)) = 2.2172892$$

and

$$y(1.0) \approx \omega_5 = \omega_4 + \frac{0.2}{24}(55f(0.8, \omega_4) - 59f(0.6, \omega_3) + 37f(0.4, \omega_2) -$$
$$9f(0.2, \omega_1)) = 2.1272892 + \frac{0.2}{24}(55f(0.8,2.1272892) - 59f(0.6,1.6489220) +$$

$$37f(0.4,1.2140762) - 9f(0.2,0.8292933))$$
$$= 2.1272892 + 0.0083333(55(2.4872892) - 59(2.2889220) + 37(2.0540762)$$
$$- 9(1.7892933)) = 2.6410533$$

The error for these approximations at t = 0.8 and t = 1.0 are, respectively

$$|2.1272295 - 2.1272892| = 5.97 \times 10^{-5}$$
$$|2.6410533 - 2.6408591| = 1.94 \times 10^{-4}$$

The corresponding Runge-Kutta approximations had errors

$$|2.1272027 - 2.1272892| = 2.69 \times 10^{-5}$$
$$|2.6408227 - 2.6408591| = 3.64 \times 10^{-4}$$

To begin the derivation of a multistep method, note that the solution to the initial-value problem

$$y' = f(t, y)y - t^2 + 1, \quad a \le t \le b, \quad y(a) = \alpha$$

if integrated over the interval $[t_i, t_{i+1}]$

$$y(t_{i+1}) - y(t_i) = \int_{t_i}^{t_{i+1}} y'(t)dt = \int_{t_i}^{t_{i+1}} f(t, y(t))dt$$

Consequently,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))dt \qquad (5.27)$$

we instead integrate an interpolating polynomial $P(t)$ to $f(t, y(t))$.When we assume, in addition, that $y(t_i) \approx \omega_i$, Eq. (5.27) becomes

$$y(t_i) \approx \omega_i + \int_{t_i}^{t_{i+1}} P(t)dt \qquad (5.28)$$

it is most convenient to use the Newton backward-difference formulad for the derivation.

To derive an Adams-Bashforth explicit m-step technique, we form the backward-difference polynomial $P_{m-1}(t)$ through

$$(t_i, f(t_i, y(t_i))), (t_{i-1}, f(t_{i-1}, y(t_{i-1}))), \ldots,$$
$$(t_{i+1-m}, f(t_{i+1-m}, y(t_{i+1-m}))).$$

Since $P_{m-1}(t)$ is an interpolatory polynomial of degree $m-1$, some number ξ_i in (t_{i+1-m}, t_i) exists with

$$f(t, y(t)) = P_{m-1}(t) + \frac{f^{(m)}(\xi_i, f(\xi_i))}{m!}(t - t_i)(t - t_{i-1}) \cdots (t - t_{i+1-m})$$

Introducing the variable substitution $t = t_i + sh$, with $dt = h\,ds$, into

$P_{m-1}(t)$and the error term implies that

$$\int_{t_i}^{t_{i+1}} f(t, y(t))dt = \int_{t_i}^{t_{i+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f(t_i, y(t_i)))dt$$

$$+ \int_{t_i}^{t_{i+1}} \frac{f^{(m)}(\xi_i, f(\xi_i))}{m!} (t - t_i)(t - t_{i-1}) \cdots (t - t_{i+1-m})\, dt$$

$$= \sum_{k=0}^{m-1} \nabla^k f(t_i, y(t_i)) h (-1)^k \int_0^1 \binom{-s}{k} ds$$

$$+ \frac{h^{m+1}}{m!} \int_0^1 s(s+1) \cdots (s+m+1) f^{(m)}(\xi_i, f(\xi_i)) ds$$

The integrals $(-1)^k \int_0^1 \binom{-s}{k} ds$ for various values of k are easily evaluated and are listed in Table 5.12.

Table 5.12

k	$\int_0^1 \binom{-s}{k} ds$
0	1
1	$\frac{1}{2}$
2	$\frac{5}{12}$
3	$\frac{3}{8}$
4	$\frac{251}{720}$
5	$\frac{95}{288}$

As a consequence,

$$\int_{t_i}^{t_{i+1}} f(t, y(t))dt$$

$$= h[f(t_i, y(t_i)) + \frac{1}{2}\nabla f(t_i, y(t_i)) + \frac{5}{12}\nabla^2 f(t_i, y(t_i)) + \ldots]$$

$$+ \frac{h^{m+1}}{m!}\int_0^1 s(s+1)\cdots(s+m+1)f^{(m)}(\xi_i, f(\xi_i))ds. \quad (5.29)$$

where $t_{i+1-m} < \mu_i < t_{i+1}$, the error term in Eq. (5.29) becomes

$$\frac{h^{m+1}}{m!}\int_0^1 s(s+1)\cdots(s+m+1)f^{(m)}(\xi_i, f(\xi_i))ds$$

$$= \frac{h^{m+1}f^{(m)}(\mu_i, y(\mu_i))}{m!}\int_0^1 s(s+1)\cdots(s+m+1)ds\,.$$

Hence the error in (5.29) simplifies to

$$h^{m+1}f^{(m)}(\mu_i, y(\mu_i))(-1)^m\int_0^1 \binom{-s}{m}ds \qquad (5.30)$$

But $y(t_{i-1}) - y(t_i) = \int_{t_i}^{t_{i+1}} f(t, y(t))dt$, so Eq. (5.27) can be written as

$$y(t_{i-1}) = y(t_i) + h[f(t_i, y(t_i)) + \frac{1}{2}\nabla f(t_i, y(t_i)) + \frac{5}{12}\nabla^2 f(t_i, y(t_i)) + \ldots] + h^{m+1} f^{(m)}(\mu_i, y(\mu_i))(-1)^m \int_0^1 \binom{-s}{m} ds. \quad (5.31)$$

Example 2 Use Eq. (5.31) with m = 3 to derive the three-step Adams-Bashforth technique.

Solution We have

$$y(t_{i-1}) \approx y(t_i) + h[f(t_i, y(t_i)) + \frac{1}{2}\nabla f(t_i, y(t_i)) + \frac{5}{12}\nabla^2 f(t_i, y(t_i)]$$

But

$$= y(t_i) + h\{f(t_i, y(t_i)) + \frac{1}{2}[f(t_i, y(t_i)) - f(t_{i-1}, y(t_{i-1}))] + \frac{5}{12}[f(t_i, y(t_i)) - 2f(t_{i-1}, y(t_{i-1})) + f(t_{i-2}, y(t_{i-2}))]\}$$

$$= y(t_i) + \frac{h}{12}[23f(t_i, y(t_i)) - 16f(t_{i-1}, y(t_{i-1}))] + 5f(t_{i-2}, y(t_{i-2}))]$$

The three-step Adams-Bashforth method is, consequently

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2,$$

$$\omega_{i+1} = \omega_i + \frac{h}{12}[23f(t_i, \omega_i) - 16f(t_{i-1}, \omega_{i-1})] + 5f(t_{i-2}, \omega_{i-2})] \quad ,$$

$$\text{for } i = 2,3,\ldots,N-1$$

The local truncation error for multistep methods is defined analogously to that of one-step methods.

Definition 5.15 If $y(t)$ is the solution to the initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

and

$$\begin{aligned}&\omega_{i+1}\\&= a_{m-1}\omega_i + a_{m-2}\omega_{i-1} + \ldots + a_0\omega_{i+1-m} + h[b_m f(t_{i+1}, \omega_{i+1})\\&+ b_{m-1} f(t_i, \omega_i) + \ldots + b_0 f(t_{i+1-m}, \omega_{i+1-m})]\end{aligned}$$

is the $(i+1)st$ step in a multistep method, the local truncation error at this step is

$$\tau_{i+1}(h) = \frac{y(t_{i-1}) - a_{m-1}y(t_i) - \ldots - a_0 y(t_{i+1-m})}{h}$$
$$-[b_m f(t_{i+1}, y(t_{i+1})) + \ldots + b_0 f(t_{i+1-m}, y(t_{i+1-m}))] \qquad (5.32)$$

for $each = m-1, m, \ldots, N-1,$

Example 3 Determine the local truncation error for the three-step Adams-Bashforth method derived in Example 2.

Solution Considering the form of the error given in Eq. (5.30) and the appropriate entry in Table 5.12 gives

$$h^4 f^{(3)}(\mu_i, y(\mu_i))(-1)^3 \int_0^1 \binom{-s}{3} ds = \frac{3h^4}{8} f^{(3)}(\mu_i, y(\mu_i))$$

Using the fact that $f^{(3)}(\mu_i, y(\mu_i))=y^{(4)}(\mu_i)$, and the difference equation derived in Example 2, we have

$$\tau_{i+1}(h) = \frac{y(t_{i-1})-y(t_i)}{h} - \frac{1}{12}[23f(t_i, y(t_i)) - 16f(t_{i-1}, y(t_{i-1})) + 5f(t_{i-2}, y(t_{i-2}))] = \frac{1}{h}[\frac{3h^4}{8} f^{(3)}(\mu_i, y(\mu_i))]=\frac{3h^4}{8} y^{(4)}(\mu_i),$$

$for\ some\ \mu_i \in (t_{i-2}, t_{i+1})$

Adams-Bashforth Explicit Methods

- Some of the explicit multistep methods together with their required starting values and local truncation errors are as follows.
- The derivation of these techniques is similar to the procedure in Examples 2 and 3.

Adams-Bashforth Two-Step Explicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1,$$

$$\omega_{i+1} = \omega_i + \frac{h}{2}[3f(t_i, \omega_i) - f(t_{i-1}, \omega_{i-1})] \quad , \qquad (5.33)$$

where $i = 1,2,\ldots,N-1$.The local truncation error is $\tau_{i+1}(h) = \frac{5}{12}y'''(\mu_i)h^2$, for some $\mu_i \in (t_{i-1}, t_{i+1})$

Adams-Bashforth Three-Step Explicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2$$

$$\omega_{i+1} = \omega_i + \frac{h}{12}[23f(t_i, \omega_i) - 16f(t_{i-1}, \omega_{i-1}) + 5f(t_{i-2}, \omega_{i-2})] \tag{5.34}$$

where $i = 2,3, \ \ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = \frac{3}{8}y^{(4)}(\mu_i)h^3$, for some $\mu_i \in (t_{i-2}, t_{i+1})$

Adams-Bashforth Four-Step Explicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2, \qquad \omega_3 = \alpha_3$$

$$\omega_{i+1} = \omega_i + \frac{h}{24}[55f(t_i, \omega_i) - 59f(t_{i-1}, \omega_{i-1}) + 37f(t_{i-2}, \omega_{i-2}) - 9f(t_{i-3}, \omega_{i-3})] \tag{5.35}$$

where $i = 3,4\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = \frac{251}{720} y^{(5)}(\mu_i)h^4$, for some $\mu_i \in (t_{i-3}, t_{i+1})$

Adams-Bashforth Five-Step Explicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2, \qquad \omega_3 = \alpha_3, \qquad \omega_4 = \alpha_4$$

$$\omega_{i+1} = \omega_i + \frac{h}{720}[1901f(t_i, \omega_i) - 2774f(t_{i-1}, \omega_{i-1}) + 2616f(t_{i-2}, \omega_{i-2}) - 1274f(t_{i-3}, \omega_{i-3}) + 251f(t_{i-4}, \omega_{i-4})] \tag{5.36}$$

where $i = 4,5\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = \frac{95}{288} y^{(6)}(\mu_i)h^5$, for some $\mu_i \in (t_{i-4}, t_{i+1})$

Adams-Moulton Implicit Methods

Implicit methods are derived by using $(t_{i+1}, f(t_{i+1}, y(t_{i+1})))$as an additional interpolation node in the approximation of the integral

$$\int_{t_i}^{t_{i+1}} f(t, y(t))dt.$$

Some of the more common implicit methods are as follows.

Adams-Moulton Two-Step Implicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1,$$

$$\omega_{i+1} = \omega_i + \frac{h}{12}[5f(t_{i+1}, \omega_{i+1}) + 8f(t_i, \omega_i) - f(t_{i-1}, \omega_{i-1})] \tag{5.37}$$

where $i = 1,2\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = -\frac{1}{24}y^{(4)}(\mu_i)h^3$, for some $\mu_i \in (t_{i-1}, t_{i+1})$

Adams-Moulton Three-Step Implicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2$$

$$\omega_{i+1} = \omega_i + \frac{h}{24}[9f(t_{i+1}, \omega_{i+1}) + 19f(t_i, \omega_i) - 5f(t_{i-1}, \omega_{i-1}) + f(t_{i-2}, \omega_{i-2})] \tag{5.38}$$

where $i = 2, \ 3\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = -\frac{19}{720}y^{(5)}(\mu_i)h^4$, for some $\mu_i \in (t_{i-2}, t_{i+1})$

Adams-Moulton Three-Step Implicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2$$

$$\omega_{i+1} = \omega_i + \frac{h}{24}[9f(t_{i+1}, \omega_{i+1}) + 19f(t_i, \omega_i) - 5f(t_{i-1}, \omega_{i-1}) + f(t_{i-2}, \omega_{i-2})] \tag{5.38}$$

where $i = 2,3\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = -\frac{19}{720}y^{(5)}(\mu_i)h^4$, for some $\mu_i \in (t_{i-2}, t_{i+1})$

Adams-Moulton Four-Step Implicit Method

$$\omega_0 = \alpha, \qquad \omega_1 = \alpha_1, \qquad \omega_2 = \alpha_2, \qquad \omega_3 = \alpha_3$$

$$\omega_{i+1} = \omega_i + \frac{h}{720}[251f(t_{i+1}, \omega_{i+1}) + 646f(t_i, \omega_i)$$
$$-264f(t_{i-1}, \omega_{i-1}) + 106f(t_{i-2}, \omega_{i-2}) - 19f(t_{i-3}, \omega_{i-3})] \tag{5.39}$$

where $i = 3,4\ldots, N-1$.The local truncation error is $\tau_{i+1}(h) = -\frac{3}{160}y^{(6)}(\mu_i)h^5$, for some $\mu_i \in (t_{i-3}, t_{i+1})$

Example 4 Consider the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

Use the exact values given from $y(t) = (t+1)^2 - 0.5e^t$ as starting values and h = 0.2 to compare the approximations from **(a)** by the explicit Adams-Bashforth four-step method and **(b)** the implicit Adams-Moulton three-step method.

Solution **(a)** The Adams-Bashforth method has the difference equation

$$\omega_{i+1} = \omega_i + \frac{h}{24}[55f(t_i, \omega_i) - 59f(t_{i-1}, \omega_{i-1}) + 37f(t_{i-2}, \omega_{i-2}) - 9f(t_{i-3}, \omega_{i-3})]$$

for i = 3, 4, ... , 9. When simplified using $f(t, y) = y - t^2 + 1$, h = 0.2, and $t_i = 0.2i$, it becomes

$$\omega_{i+1} = \frac{1}{24}[35\omega_i - 11.8\omega_{i-1} + 7.4\omega_{i-2} - 1.8\omega_{i-3} - 0.192i^2 - 0.192\mathrm{i} + 4.736]$$

Solution **(b)**The Adams-Bashforth method has the difference equation

$$\omega_{i+1} = \omega_i + \frac{h}{24}[9f(t_{i+1}, \omega_{i+1}) + 19f(t_i, \omega_i) - 5f(t_{i-1}, \omega_{i-1}) + f(t_{i-2}, \omega_{i-2})]$$

for i = 2,3, 4, ... , 9. This reduces to

$$\omega_{i+1} = \frac{1}{24}[1.8\omega_{i+1} + 27.8\omega_i - \omega_{i-1} + 0.2\omega_{i-2} - 0.192i^2 - 0.192\mathrm{i} + 4.736]$$

To use this method explicitly, we meed to solve the equation explicitly solve for ω_{i+1}.

This gives

$$\omega_{i+1} = \frac{1}{22.2}[27.8\omega_i - \omega_{i-1} + 0.2\omega_{i-2} - 0.192i^2 - 0.192\mathrm{i} + 4.736]$$

for i = 2,3, ... , 9.

Solution The results in Table 5.13 were obtained using the exact values from $y(t) = (t+1)^2 - 0.5e^t$ for $\alpha, \alpha_1, \alpha_2$ and α_3 in the explicit Adams-Bashforth case and for α, α_1 and α_2 in the implicit Adams-Moulton case.

Table 5.13

		Adams-Bashforth		Adams-Moulton	
t_i	Exact	w_i	Error	w_i	Error
0.0	0.5000000				
0.2	0.8292986				
0.4	1.2140877				
0.6	1.6489406			1.6489341	0.0000065
0.8	2.1272295	2.1273124	0.0000828	2.1272136	0.0000160
1.0	2.6408591	2.6410810	0.0002219	2.6408298	0.0000293
1.2	3.1799415	3.1803480	0.0004065	3.1798937	0.0000478
1.4	3.7324000	3.7330601	0.0006601	3.7323270	0.0000731
1.6	4.2834838	4.2844931	0.0010093	4.2833767	0.0001071
1.8	4.8151763	4.8166575	0.0014812	4.8150236	0.0001527
2.0	5.3054720	5.3075838	0.0021119	5.3052587	0.0002132

Predictor-Corrector Methods

Although this is generally the case, the implicit methods have the inherent weakness of first having to convert the method algebraically to an explicit representation for ω_{i+1}.

$$y' = e^y, \qquad 0 \le t \le 0.25, \qquad y(0) = 1$$

Because $f(t, y) = e^y$, the three-step Adams-Moulton method has

$$\omega_{i+1} = \omega_i + \frac{h}{24}[9e^{\omega_{i+1}} + 19e^{\omega_i} - 5e^{\omega_{i-1}} + e^{\omega_{i-2}}]$$

as its difference equation, and this equation cannot be algebraically solved for ω_{i+1}.

In practice, implicit multistep methods are not used as described above.

- predictor-corrector method:The combination of an explicit method to predict and an implicit to improve the prediction.

Consider the following fourth-order method for solving an initial-value problem.

- The first step is to calculate the starting values $\omega, \omega_1, \omega_2$ and ω_3 for the four-step explicit Adams-Bashforth method.
- The next step is to calculate an approximation,ω_{4p}, to $y(t_4)$ using the explicit Adams-Bashforth method as predictor

$$\omega_{4\mathrm{p}} = \omega_3 + \frac{h}{24}[55f(t_3, \omega_3) - 59f(t_2, \omega_2) + 37f(t_1, \omega_1) - 9f(t_0, \omega_0)]$$

This approximation is improved by inserting w4p in the right side of the three-step implicit Adams-Moulton method and using that method as a corrector. This gives

$$\omega_4 = \omega_3 + \frac{h}{24}[9f(t_4, \omega_4) + 19f(t_3, \omega_3) - 5f(t_2, \omega_2) + f(t_1, \omega_1)]$$

The only new function evaluation required in this procedure is $f(t_4, \omega_{4\mathrm{p}})$ in the corrector equation; all the other values of f have been calculated for earlier approximations.

ALGORITHM 5.4 Adams Fourth-Order Predictor-Corrector

To approximate the solution of the initial-value problem

$$y' = f(t, y), a \leq t \leq b, \qquad y(a) = \alpha$$

at $(N + 1)$ equally spaced numbers in the interval $[a, b]$:

INPUT *endpoints a, b;integer N; initial condition α.*

OUTPUT *approximation ω to y at the (N + 1) values of t.*

Step 1 *Set* $h = (b - a)/N$;

$t_0 = a$;

$\omega_0 = \alpha$;

OUTPUT (t_0, ω_0).

Step 2 For i = 1, 2, 3, do Steps 3–5.

(Compute starting values using Runge-Kutta method.)

Step 3 *Set* $K_1 = hf(t_{i-1}, \omega_{i-1})$;

$$K_2 = hf(t_{i-1} + h/2, \omega_{i-1} + K_1/2)$$

$$K_3 = hf(t_{i-1} + h/2, \omega_{i-1} + K_2/2)$$

$$K_4 = hf(t_{i-1} + h, \omega_{i-1} + K_3)$$

Step 4 *Set* $\omega_i = \omega_{i-1} + (K_1 + 2K_2 + 2K_3 + K_4)/6$

$$t_i = a + ih$$

Step 5 *OUTPUT* (t_i, ω_i).

Step 6 *For i = 4, … , N do Steps 7–10*

Step 7 *Set t = a+ih;*

$$\mathrm{w} = \omega_3 + \mathrm{h}\frac{[55f(t_3,\omega_3) - 59f(t_2,\omega_2) + 37f(t_1,\omega_1) - 9f(t_0,\omega_0)]}{24};\ (\text{Predict } \omega_{\mathrm{i}}.)\omega$$

$$= \omega_3 + \mathrm{h}[9f(\mathrm{t},\omega) + 19f(t_3,\omega_3) - 5f(t_2,\omega_2) + f(t_1,\omega_1)$$

$$- 9f(t_0,\omega_0)]/24;\ (\text{Correct } \omega_{\mathrm{i}}.)$$

Step 8 OUTPUT (t,ω)

Step 9 *For j = 0, 1, 2*

set $t_j = t_{j+1}$; *(Prepare for next iteration.)*

$\omega_j = \omega_{j+1}$.

Step 10 *set* $t_3 = t;$

$$\omega_3 = \omega.$$

Step 11 ***STOP.***

Example 5 Apply the Adams fourth-order predictor-corrector method with h = 0.2 and starting values from the Runge-Kutta fourth order method to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

Solution This is continuation and modification of the problem considered in Example 1 at the beginning of the section. In that example we found that the starting approximations from Runge-Kutta are

$$y(0) = \omega_0 = 0.5, y(0.2) \approx \omega_1 = 0.8292933, y(0.4) \approx \omega_2 = 1.2140762,$$
$$and y(0.6) \approx \omega_3 = 1.6489220$$

and the fourth-order Adams-Bashforth method gave

$$y(0.8) \approx \omega_{4\mathrm{p}}$$
$$= \omega_3 + \frac{0.2}{24}[55f(0.6, \omega_3) - 59f(t_2, \omega_2) + 37f(t_1, \omega_1) - 9f(0,0.5)]$$
$$== 2.1272892.$$

We will now use $\omega_{4\mathrm{p}}$ as the predictor of the approximation to y(0.8)

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \quad y(0) = 0.5$$

$$y(0.8) \approx \omega_{4\mathrm{p}}$$

$$= \omega_3 + \frac{0.2}{24}[9f(0.8, \omega_{4\mathrm{p}}) + 19f(0.6, \omega_3) - 5f(0.4, \omega_2) + f(0.2, \omega_1)]$$

$$= 2.1272056.$$

Now we use this approximation to determine the predictor, $\omega_{5\mathrm{p}}$, for y(1.0) as

$$y(1.0) \approx \omega_{5\mathrm{p}}$$

$$= \omega_4 + \frac{0.2}{24}[55f(0.8, \omega_4) - 59f(0.6, \omega_3) + 37f(0.4, \omega_2) - 9f(0.2, \omega_1)]$$

$$== 2.6409314.$$

and correct this with

and correct this with

$$y(1.0) \approx \omega_5$$
$$= \omega_4 + \frac{0.2}{24}[9f(1.0, \omega_{5\mathrm{p}}) + 19f(0.8, \omega_4) - 5f(0.6, \omega_3) + f(0.4, \omega_2) =$$
$$= \ 2.6408286$$

In Example 1 we found that using the explicit Adams-Bashforth method alone produced results that were inferior to those of Runge-Kutta. However, these approximations to y(0.8) and y(1.0) are accurate to within

$$|2.1272295 - 2.1272056| = 2.39 \times 10^{-5}$$
$$|2.6408286 - 2.6408591| = 3.05 \times 10^{-5}$$

respectively, compared to those of Runge-Kutta, which were accurate, respectively, to within

$$|2.1272027 - 2.1272892| = 2.69 \times 10^{-5}$$
$$|2.6408227 - 2.6408591| = 3.64 \times 10^{-5}$$

The remaining predictor-corrector approximations were generated using Algorithm 5.4 .

Table 5.14

t_i	$y_i = y(t_i)$	w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272056	0.0000239
1.0	2.6408591	2.6408286	0.0000305
1.2	3.1799415	3.1799026	0.0000389
1.4	3.7324000	3.7323505	0.0000495
1.6	4.2834838	4.2834208	0.0000630
1.8	4.8151763	4.8150964	0.0000799
2.0	5.3054720	5.3053707	0.0001013

When an interpolating polynomial is integrated over $[t_{i-3}, t_{i+1}]$, the result is the explicit **Milne's method**:

$$\omega_{i+1} = \omega_{i-3} + \frac{4h}{3}[2\mathrm{f}(t_i, \omega_i) - \mathrm{f}(t_{i-1}, \omega_{i-1}) + 2\mathrm{f}(t_{i-2}, \omega_{i-2})]$$

which has local truncation error $\frac{14}{45}h^4 y^{(5)}(\xi_i)$, for some $\xi_i \in (t_{i-3}, t_{i+1})$.

Milne's method is occasionally used as a predictor for the implicit **Simpson's method**

$$\omega_{i+1} = \omega_{i-1} + \frac{h}{3}[\mathrm{f}(t_{i+1}, \omega_{i+1}) + 4\mathrm{f}(t_i, \omega_i) + \mathrm{f}(t_{i-1}, \omega_{i-1})]$$

which has local truncation error $-\frac{h^4}{90}y^{(5)}(\xi_i)$, for some $\xi_i \in (t_{i-1}, t_{i+1})$, and is obtained by integrating an interpolating polynomial over $[t_{i-1}, t_{i+1}]$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 5-9

Higher-Order Equations and Systems of Differential Equations

高阶方程和微分方程组

Introduction

- This section contains an introduction to the numerical solution of higher-order initial-value problems.
- Before discussing the transformation procedure, some remarks are needed concerning systems that involve first-order differential equations.

An ***m*th-order system** of first-order initial-value problems has the form

$$\frac{du_1}{dt} = f_1(t, u_1, u_2, \ldots, u_m),$$
$$\frac{du_2}{dt} = f_2(t, u_1, u_2, \ldots, u_m),$$
$$\ldots\ldots$$
$$\frac{du_m}{dt} = f_m(t, u_1, u_2, \ldots, u_m), \qquad (5.45)$$

for a ≤ t ≤ b, with the initial conditions

$$u_1(a) = \alpha_1, \qquad u_2(a) = \alpha_2, \ldots, u_m(a) = \alpha_m \qquad (5.46)$$

The object is to find m functions $u_1(t), u_2(t), \ldots, u_m(t)$ that satisfy each of the differential equations together with all the initial conditions.

By using the Mean Value Theorem, it can be shown that if f and its first partial derivatives are continuous on D and if

$$\left|\frac{\partial f(t, u_1, u_2, \ldots, u_m)}{\partial u_i}\right| \leq L$$

for each $i = 1,2,..,m$ and all $(t, u_1, u_2, \ldots, u_m)$ in D, then f satisfies a Lipschitz condition on D with Lipschitz constant L.

Theorem 5.17 Suppose that

$$D = \{(t, u_1, u_2, \ldots, u_m) | a \le t \le b \text{ and } -\infty \le u_i \le \infty, \text{for each } i = 1,2,..,m\}$$

and let $f_i(t, u_1, u_2, \ldots, u_m)$,for each $i = 1,2,..,m$,be continuous and satisfy a Lipschitz condition on D. The system of first-order differential equations (5.45), subject to the initial conditions (5.46), has a unique solution $u_1(t), \ldots, u_m(t)$,for $a \le t \le b$.

- Methods to solve systems of first-order differential equations are generalizations of the methods for a single first-order equation presented earlier in this chapter.

For example

the classical Runge-Kutta method of order four given by

$$\omega_0 = \alpha,$$
$$k_1 = hf(t_i, \omega_i),$$
$$k_2 = hf(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1),$$
$$k_3 = hf(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_2),$$
$$k_4 = hf(t_{i+1}, \omega_i + k_3),$$

$$\omega_{i+1} = \omega_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \qquad \text{for each } i = 0,1,\ldots,N-1,$$

used to solve the first-order initial-value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

is generalized as follows.

Let an integer N > 0 be chosen and set h = (b−a)/N. Partition the interval [a, b] into N subintervals with the mesh points

$$t_j = a + jh, \qquad for\ each\ j = 0,1,\ldots,N$$

Use the notation ω_{ij}, for each $j = 0,1,\ldots,N$ and i = 1, 2, ... , m, to denote an approximation to $u_i(t_j)$. That is, ω_{ij} approximates the ith solution $u_i(t)$ of (5.45) at the jth mesh point t_j. For the initial conditions, set (see Figure 5.6)

$$\omega_{1,0} = \alpha_1, \qquad \omega_{2,0} = \alpha_2, \ldots, \omega_{m,0} = \alpha_m \qquad (5.48)$$

Figure 5.6

Suppose that t the values $\omega_{1,j}, \omega_{2,j}, \ldots, \omega_{m,j}$ have been computed. We obtain $\omega_{1,j+1}, \omega_{2,j+1}, \ldots, \omega_{m,j+1}$ by first calculating

$$k_{1,i} = hf_i(t_j, \omega_{1,j}, \omega_{2,j}, \ldots, \omega_{m,j}), for\ each\ i = 1,2,..,m; \qquad (5.49)$$

$$k_{2,i} = hf_i(t_j + \frac{h}{2}, \omega_{1,\mathrm{j}} + \frac{1}{2}k_{1,1}, \omega_{2,\mathrm{j}} + \frac{1}{2}k_{1,2}, \ldots, \omega_{\mathrm{m,j}} + \frac{1}{2}k_{1,m}), \text{for each } i = 1,2,..,m; \quad (5.50)$$

$$k_{3,i} = hf_i(t_j + \frac{h}{2}, \omega_{1,\mathrm{j}} + \frac{1}{2}k_{2,1}, \omega_{2,\mathrm{j}} + \frac{1}{2}k_{2,2}, \ldots, \omega_{\mathrm{m,j}} + \frac{1}{2}k_{2,m}), \text{for each } i = 1,2,..,m; \quad (5.51)$$

$$k_{4,i} = hf_i(t_j + h, \omega_{1,\mathrm{j}} + k_{3,1}, \omega_{2,\mathrm{j}} + k_{3,2}, \ldots, \omega_{\mathrm{m,j}} + k_{3,m}), \text{for each } i = 1,2,..,m; \quad (5.52)$$

$$\omega_{\mathrm{i,j+1}} = \omega_{\mathrm{i,j}} + \frac{1}{6}(k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}), \text{for each } i = 1,2,..,m; \quad (5.53)$$

- Algorithm 5.7 implements the Runge-Kutta fourth-order method for systems of initial-value problems.

ALGORITHM 5.7
Runge-Kutta Method for Systems of Differential Equations

To approximate the solution of the *m*th-order system of first-order initial-value problems

$$u_j' = f_j(t, u_1, u_2, \ldots, u_m), a \le t \le b, with\ u_j(a) = \alpha_j,$$

$for\ each\ j = 1,2,..,m$; at $(N+1)$ equally spaced numbers in the interval $[a, b]$:

INPUT *endpoints a, b;number of equations m;integer N; initial condition*

$\alpha_1, \alpha_2, \ldots, \alpha_m$.

OUTPUT *approximation* ω_j *to* $u_j(t)$ *at the (N + 1) values of t.*

Step 1 *Set* $h = (b - a)/N$;

$$t = a;$$

Step 2 *For* $j = 1, 2, \ldots, m$ *set* $w_j = \alpha_j$.

Step 3 *OUTPUT*$(t, \omega_1, \omega_2, \ldots, \omega_m)$.

Step 4 *For* $i = 1, 2, \ldots, N$ *do steps 5–11.*

Step 5 *For* $j = 1, 2, \ldots, m$ *set*

$$k_{1,j} = hf_j(t, \omega_1, \omega_2, \ldots, \omega_m).$$

Step 6 *For* $j = 1, 2, \ldots, m$ *set*

$$k_{2,j} = hf_j(t + \frac{h}{2}, \omega_1 + \frac{1}{2}k_{1,1}, \omega_2 + \frac{1}{2}k_{1,2}, \ldots, \omega_\mathrm{m} + \frac{1}{2}k_{1,m})$$

Step 7 *For j = 1, 2, … , m set*

$$k_{3,j} = hf_j(t + \frac{h}{2}, \omega_1 + \frac{1}{2}k_{2,1}, \omega_2 + \frac{1}{2}k_{2,2}, \ldots, \omega_\mathrm{m} + \frac{1}{2}k_{2,m})$$

Step 8 *For j = 1, 2, … , m set*

$$k_{4,j} = hf_j(t + h, \omega_1 + k_{3,1}, \omega_2 + k_{3,2}, \ldots, \omega_\mathrm{m} + k_{3,m})$$

Step 9 *For j = 1, 2, … , m set*

$$\omega_\mathrm{j} = \omega_\mathrm{j} + (k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6$$

Step 10 *Set* $t = a + ih$.

Step 11 $OUTPUT(t, \omega_1, \omega_2, \ldots, \omega_m)$.

Step 12 *STOP.*

Higher-Order Differential Equations

- By relabeling the variables, we can reduce a higher-order differential equation into a system of first-order differential equations and then apply one of the methods we have already discussed.

A general mth-order initial-value problem

$$y^{(m)}(t) = f(t, y, y', \ldots, y^{(m-1)}) a \leq t \leq b,$$

Let $u_1(t) = y(t), u_2(t) = y'(t), \ldots,$and $u_m(t) = y^{(m-1)}(t)$.This produces the first-order system

$$\frac{du_1}{dt} = \frac{dy}{dt} = u_2, \frac{du_2}{dt} = \frac{dy'}{dt} = u_3, \qquad \ldots, \qquad \frac{du_{m-1}}{dt} = \frac{dy^{(m-2)}}{dt} = u_m,$$

and

$$\frac{du_m}{dt} = \frac{dy^{(m-1)}}{dt} = y^{(m)} = f(t, y, y', \ldots, y^{(m-1)}) = f(t, u_1, u_2, \ldots, u_m),$$

with initial conditions

$$u_1(a) = y(a) = \alpha_1, u_2(a) = y'(a) = \alpha_2, \ldots, \text{and } u_m(a) = y^{(m-1)}(a) = \alpha_m.$$

Example 1 Transform the the second-order initial-value problem

$$y'' - 2y' + 2y = e^{2t} \sin t, \quad for\ 0 \le t \le 1, \ with\ y(0) = -0.4, y'(0) = -0.6$$

into a system of first order initial-value problems, and use the Runge-Kutta method with h = 0.1 to approximate the solution.

Solution Let $u_1(t) = y(t), u_2(t) = y'(t)$.This transforms the second-order equation into the system

$$u_1'(t) = u_2(t)$$
$$u_2'(t) = e^{2t} \sin t - 2u_1(t) + 2u_2(t)$$

with initial conditions $u_1(0) = -0.4$, $u_2(0) = -0.6$.

The initial conditions give $\omega_{1,0} = -0.4$ and $\omega_{2,0} = -0.6$.

We have

$$k_{1,1} = hf_1(t_0, \omega_{1,0}, \omega_{2,0}) = h\omega_{2,0} = -0.06$$

$$k_{1,2} = hf_2(t_0, \omega_{1,0}, \omega_{2,0}) = h[e^{2t_0} sin\, t_0 - 2\omega_{1,0} + 2\omega_{2,0}] = -0.04$$

$$k_{2,1} = hf_1(t_0 + \frac{1}{2}h, \omega_{1,0} + \frac{1}{2}k_{1,1}, \omega_{2,0} + \frac{1}{2}k_{1,2}) = h[\omega_{2,0} + \frac{1}{2}k_{1,2}] = -0.062$$

$$k_{2,2} = hf_2(t_0 + \frac{1}{2}h, \omega_{1,0} + \frac{1}{2}k_{1,1}, \omega_{2,0} + \frac{1}{2}k_{1,2})$$

$$= h[e^{2(t_0+0.05)} sin(\, t_0 + 0.05) - 2(\omega_{1,0} + \frac{1}{2}k_{1,1}) + 2(\omega_{2,0} + \frac{1}{2}k_{1,2})]$$

$$= -0.03247644757$$

$$k_{3,1} = h[\omega_{2,0} + \frac{1}{2}k_{2,2}] = -0.06162832238,$$

$$k_{3,2} = -0.03152409237$$

$$k_{4,1} = -0.06315240924,$$

$k_{4,2} = -0.02178637298,$

So

$$\omega_{1,1} = \omega_{1,0} + \frac{1}{6}(k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1}) = -0.4617333423$$

$$\omega_{2,1} = \omega_{2,0} + \frac{1}{6}(k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2}) = -0.6316312421$$

- The value $\omega_{1,1}$ approximates $u_1(0.1) = y(0.1) = 0.2e^{2(0.1)}(sin0.1 - 2cos0.1)$,and $\omega_{2,1}$ approximates $u_2(0.1) = y'(0.1) = 0.2e^{2(0.1)}(4sin0.1 - 3cos0.1)$.

- The set of values $\omega_{1,j}$ and $\omega_{2,j}$,for j = 0, 1, ... , 10, are presented in Table 5.20 and are compared to the actual values of $u_1(t) = 0.2e^{2t}(sint - 2cost)$ and $u_2(t) = 0.2e^{2t}(4sint - 4cost)$

Table 5.20

t_j	$y(t_j) = u_1(t_j)$	$w_{1,j}$	$y'(t_j) = u_2(t_j)$	$w_{2,j}$	$\lvert y(t_j) - w_{1,j}\rvert$	$\lvert y'(t_j) - w_{2,j}\rvert$
0.0	−0.40000000	−0.40000000	−0.6000000	−0.60000000	0	0
0.1	−0.46173297	−0.46173334	−0.6316304	−0.63163124	3.7×10^{-7}	7.75×10^{-7}
0.2	−0.52555905	−0.52555988	−0.6401478	−0.64014895	8.3×10^{-7}	1.01×10^{-6}
0.3	−0.58860005	−0.58860144	−0.6136630	−0.61366381	1.39×10^{-6}	8.34×10^{-7}
0.4	−0.64661028	−0.64661231	−0.5365821	−0.53658203	2.03×10^{-6}	1.79×10^{-7}
0.5	−0.69356395	−0.69356666	−0.3887395	−0.38873810	2.71×10^{-6}	5.96×10^{-7}
0.6	−0.72114849	−0.72115190	−0.1443834	−0.14438087	3.41×10^{-6}	7.75×10^{-7}
0.7	−0.71814890	−0.71815295	0.2289917	0.22899702	4.05×10^{-6}	2.03×10^{-6}
0.8	−0.66970677	−0.66971133	0.7719815	0.77199180	4.56×10^{-6}	5.30×10^{-6}
0.9	−0.55643814	−0.55644290	1.534764	1.5347815	4.76×10^{-6}	9.54×10^{-6}
1.0	−0.35339436	−0.35339886	2.578741	2.5787663	4.50×10^{-6}	1.34×10^{-5}

- The other one-step methods can be extended to systems in a similar way.
- The multistep methods and predictor-corrector techniques can also be extended to systems.

Example Find a value of α to make the following technique

$$w_{i+1} + \alpha(w_i - w_{i-1}) - w_{i-2} = \frac{1}{2}(3+\alpha)h(f(t_i, w_i) + f(t_{i-1}, w_{i-1}))$$

is a four-order method.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 6-1

Direct Methods for Solving Linear Systems

Question

How to solve a linear system of n equations in n variables by direct methods?

For example, a system has the form

$$\begin{cases} E_1: & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ E_2: & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ & \vdots \\ E_n: & a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \qquad (6.1)$$

In this system we are given the constants a_{ij}, for each $i, j = 1, 2, \ldots, n$, and b_i,for each $i = 1, 2, \ldots, n$, and we need to determine the unknowns $x_1, \ldots, x_n$.

The solution of systems of this type will be considered in this chapter.

Direct techniques are methods that theoretically give the exact solution to the system in a finite number of steps. In practice, the solution obtained will be contaminated by the round-off error that is involved with the arithmetic being used. Analyzing the effect of this round-off error and determining ways to keep it under control will be a major component of this chapter.

$$\begin{cases} E_1: & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ E_2: & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ & \vdots \\ E_n: & a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \qquad (6.1)$$

We use three operations to simplify the linear system given in (6.1):

(1) Equation E_i can be multiplied by any nonzero constant λ with the resulting equation used in place of E_i. This operation is denoted $(\lambda E_i) \rightarrow (E_i)$.

(2) Equation E_j can be multiplied by any constant λ and added to equation E_i with the resulting equation used in place of E_i. This operation is denoted $(E_i + \lambda E_j) \rightarrow (E_i)$.

(3) Equations E_i and E_j can be transposed in order. This operation is denoted $(E_i) \leftrightarrow (E_j)$.

Illustration

The four equations

$$\begin{aligned} E_1 &: x_1 + x_2 + 3x_4 = 4, \\ E_2 &: 2x_1 + x_2 - 3x_3 + x_4 = 1, \\ E_3 &: 3x_1 - x_2 - x_3 + 2x_4 = -3, \\ E_4 &: -x_1 + 2x_2 + 3x_3 - x_4 = 4. \end{aligned} \quad (6.2)$$

will be solved for x_1, x_2, x_3, and x_4. We first use equation E_1 to eliminate the unknown x_1 from equations E_2, E_3, and E_4 by performing $(E_2 - 2E_1) \rightarrow (E_2)$, $(E_3 - 3E_1) \rightarrow (E_3)$, and $(E_4 + E_1) \rightarrow (E_4)$.For simplicity, the new equations are again labeled E_1, E_2, E_3, and E_4.

In the new system, E_2 is used to eliminate the unknown x_2 from E_3 and E_4 by performing $(E_3 - 4E_2) \rightarrow (E_3)$ and $(E_4 + 3E_2) \rightarrow (E_4)$.

Then, this results in:

$$\begin{aligned} E_1 &: x_1 + x_2 \quad + \; 3x_4 = \quad 4, \\ E_2 &: \quad -x_2 - x_3 - \; 5x_4 = -7, \\ E_3 &: \qquad\quad 3x_3 + 13x_4 = \quad 13, \\ E_4 &: \qquad\qquad -13x_4 = -13. \end{aligned} \qquad (6.3)$$

The system of equations (6.3) is now in *triangular (or reduced)* form and can be solved for the unknowns by a *backward-substitution process*.

The solution to system (6.3), is therefore, $x_1 = -1, x_2 = 2$, $x_3 = 0$, and $x_4 = 1$.

Matrices and Vectors

When performing the calculations in the Illustration, we would not need to write out the full equations at each step or to carry the variables x_1, x_2, x_3, and x_4 through the calculations, if they always remained in the same column.

The *only variation* from system to system occurs in the coefficients of the unknowns and in the values on the right side of the equations. For this reason, a linear system is often replaced by a matrix,which contains all the information about the system that is necessary to determine its solution.

Definition 6.1

An $n \times m$ (n by m) *matrix* is a rectangular array of elements with n rows and m columns in which not only is the value of an element important, but also its position in the array.

The notation for an n × m matrix will be a capital letter such as A for the matrix and lowercase letters with double subscripts, such as a_{ij}, to refer to the entry at the intersection of the ith row and jth column; that is,

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

$$AX = b, A = [a_{ij}], X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{cases} E_1: & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ E_2: & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ & \vdots \\ E_n: & a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \quad (6.1)$$

The 1 × n matrix $A = [a_{11}, a_{12}, \cdots a_{1n}]$ is called an *n-dimensional row vector*, and an n× 1 matrix $A = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix}$ is called an *n-dimensional column vector.*

An n × (n + 1) matrix can be used to represent the linear system (6.1),by first constructing $A = [a_{ij}] = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$,and $\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ .Then, The array [A, b] is called an *augmented matrix.*

$$[A, b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

$$\begin{aligned} E_1 &: x_1 + x_2 + 3x_4 = 4, \\ E_2 &: 2x_1 + x_2 - 3x_3 + x_4 = 1, \\ E_3 &: 3x_1 - x_2 - x_3 + 2x_4 = -3, \\ E_4 &: -x_1 + 2x_2 + 3x_3 - x_4 = 4. \end{aligned} \tag{6.2}$$

Repeating the operations involved in (6.2) with the matrix notation results in first considering the augmented matrix:

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 3 & 4 \\ 2 & 1 & -1 & 1 & 1 \\ 3 & -1 & -1 & 2 & -3 \\ -1 & 2 & 3 & -1 & 4 \end{array}\right]$$

Performing the operations as described in that example produces the augmented matrices

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 3 & 4 \\ 0 & -1 & -1 & -5 & -7 \\ 0 & -4 & -1 & -7 & -15 \\ 0 & 3 & 3 & 2 & 8 \end{array}\right] \textbf{and} \left[\begin{array}{cccc|c} 1 & 1 & 0 & 3 & 4 \\ 0 & -1 & -1 & -5 & -7 \\ 0 & 0 & 3 & 13 & 13 \\ 0 & 0 & 0 & -13 & -13 \end{array}\right]$$

Gaussian elimination with backward substitution

The final matrix can now be transformed into its corresponding linear system, and solutions for x_1, x_2, x_3, and x_4, can be obtained. The procedure is called *Gaussian elimination with backward substitution.*

Applied to the linear system:

$$\begin{aligned} E_1 &: a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ E_2 &: a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ E_n &: a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n, \end{aligned} \tag{6.4}$$

First form the augmented matrix Ã:

$$\tilde{A} = [A, b] = \left[\begin{array}{cccc:c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{array}\right] \qquad (6.5)$$

where A denotes the matrix formed by the coefficients. The entries in the (n + 1)st column are the values of b; that is, $a_{i,n+1} = b_i$ for each i = 1, 2, ⋯ , n.

Provided $a_{11} \neq 0$, we perform the operations corresponding to

$$(E_j - (a_{j1}/a_{11})E_1) \rightarrow (E_j) \text{ for each } j = 2, 3, \ldots, n$$

to eliminate the coefficient of x_1 in each of these rows. Although the entries in rows 2, 3, ... , n are expected to change, for ease of notation we again denote the entry in the *i*th row and the *j*th column by a_{ij}.

With this in mind, we follow a sequential procedure for i = 2, 3, ... , n−1 and perform the operation

$$(E_j - (a_{ji}/a_{ii})E_i) \rightarrow (E_j) \text{ for each } j = i + 1, i + 2, \ldots, n$$

provided $a_{ii} \neq 0$. This eliminates (changes the coefficient to zero) x_i in each row below the ith for all values of i = 1, 2, ... , n − 1.

The resulting matrix has the form:

$$\tilde{\tilde{A}} = \left[\begin{array}{cccc:c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nn} & a_{n,n+1} \end{array}\right]$$

Gaussian elimination with backward substitution

where, except in the first row, the values of a_{ij} are not expected to agree with those in the original matrix Ã. The matrix $\tilde{\tilde{A}}$ represents a linear system with the same solution set as the original system.

The new linear system is triangular,

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= a_{1,n+1}, \\
a_{22}x_2 + \cdots + a_{2n}x_n &= a_{2,n+1}, \\
\ddots \quad \vdots \quad &\quad \vdots \\
\ddots \quad \vdots \quad &\quad \vdots \\
a_{nn}x_n &= a_{n,n+1},
\end{aligned}
$$

So *backward substitution* can be performed.

Solving the nth equation for x_n gives

$$x_n = \frac{a_{n,n+1}}{a_{nn}}$$

Solving the (n − 1)st equation for x_{n-1} and using the known value for x_n yields

$$x_{n-1} = \frac{a_{n-1,n+1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

Continuing this process, we obtain

$$x_i = \frac{a_{i,n+1} - a_{i,n}x_n - a_{i,n-1}x_{n-1} - \cdots - a_{i,i+1}x_{i+1}}{a_{ii}} = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$$

for each $i = n - 1, n - 2, \cdots, 2, 1$.

$$\tilde{A} = [A, b] = \left[\begin{array}{cccc:c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{array}\right] \qquad (6.5)$$

Gaussian elimination procedure is described more precisely, although more intricately,by forming a sequence of augmented matrices$\tilde{A}^{(1)}, \tilde{A}^{(2)}, \ldots, \tilde{A}^{(n)}$,where $\tilde{A}^{(1)}$ is the matrix $\tilde{A}$ given in (6.5) and $\tilde{A}^{(k)}$, for each k = 2, 3,..., n, has entries $a^{(k)}_{ij}$,where:

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)}, & when\ i = 1, 2, \ldots, k-1\ and\ j = 1, 2, \ldots, n+1, \\ 0, & when\ i = k, k+1, \ldots, n\ and\ i = 1, 2, \ldots, k-1, \\ a_{ij}^{(k-1)} - \dfrac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} a_{k-1,j}^{(k-1)}, & when\ i = k, k+1, \ldots, n\ and\ j = k, k+1, \ldots, n+1. \end{cases}$$

Thus

$$\tilde{A}^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & \vdots & a_{1,n+1}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2,k-1}^{(2)} & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} & \vdots & a_{2,n+1}^{(2)} \\ \vdots & \ddots & \ddots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} & \vdots & a_{k-1,n+1}^{(k-1)} \\ \vdots & & & \ddots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & \vdots & a_{k,n+1}^{(k)} \\ \vdots & & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & \vdots & a_{n,n+1}^{(k)} \end{bmatrix} \qquad (6.6)$$

represents the equivalent linear system for which the variable x_{k-1} has just been eliminated from equations E_k , E_{k+1}, ... , E_n.

Question: Will this process always work?

The answer is no.

The procedure will fail if one of the elements $a_{11}^{(1)}$,$a_{22}^{(2)}$, $a_{33}^{(3)}$, … , $a_{n-1.n-1}^{(n-1)}$, $a_{nn}^{(n)}$ is zero because the step

$$(E_i - \frac{a_{i,k}^{(k)}}{a_{kk}^{(k)}}(E_k)) \rightarrow E_i$$

either cannot be performed (this occurs if one of $a_{11}^{(1)}$, …, $a_{n-1.n-1}^{(n-1)}$ is zero), or the backward substitution cannot be accomplished (in the case $a_{nn}^{(n)} = 0$).

The system may still have a solution, but the technique for finding the solution must be altered. An illustration is given in the following example.

Example 2 Represent the linear system

$$\begin{aligned} E_1: \quad x_1 - x_2 + 2x_3 - x_4 &= -8, \\ E_2: 2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20, \\ E_3: \quad x_1 + x_2 + x_3 \qquad &= -2, \\ E_4: \quad x_1 - x_2 + 4x_3 + 3x_4 &= 4, \end{aligned}$$

as an augmented matrix and use *Gaussian Elimination* to find its solution

Solution The augmented matrix is

$$\tilde{A} = \tilde{A}^{(1)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array}\right].$$

Gaussian Elimination

$$\tilde{A} = \tilde{A}^{(1)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array}\right].$$

Performing the operations

$$(E_2 - 2E_1) \rightarrow (E_2),\ (E_3 - E_1) \rightarrow (E_3),\ \text{and}\ (E_4 - E_1) \rightarrow (E_4),$$

gives

$$\tilde{A}^{(2)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right].$$

The diagonal entry $a_{22}^{(2)}$, called *the pivot element*, is 0, so the procedure cannot continue in its present form.

But operations $(E_i) \leftrightarrow (E_j)$ are permitted, so a search is made of the elements $a_{32}^{(2)}$ and $a_{42}^{(2)}$ for the first nonzero element.

Gaussian Elimination

$$\tilde{A}^{(2)} = \left[\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right].$$

Since $a_{32}^{(2)} \neq 0$, the operation $(E_2) \leftrightarrow (E_3)$ is performed to obtain a new matrix,

$$\tilde{A}^{(2)'} = \left[\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 2 & 4 & 12 \end{array}\right].$$

Since x_2 is already eliminated from E_3 and E_4, $\tilde{A}^{(3)}$ will be $\tilde{A}^{(2)'}$, and the computations continue with the operation $(E_4 + 2E_3) \rightarrow (E_4)$, giving

$$\tilde{A}^{(4)} = \left[\begin{array}{cccc:c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array}\right].$$

Gaussian Elimination

Finally, the matrix is converted back into a linear system that has a solution equivalent to the solution of the original system and the backward substitution is applied:

$$x_4 = \frac{4}{2} = 2,$$

$$x_3 = \frac{[-4 - (-1)x_4]}{-1} = 2,$$

$$x_2 = \frac{[6 - x_4 - (-1)x_3]}{2} = 3,$$

$$x_1 = \frac{[-8 - (-1)x_4 - 2x_3 - (-1)x_2]}{1} = -7.$$

Example 2 illustrates what is done if $a_{kk}^{(k)} = 0$ for some k = 1, 2, ... , n − 1. The *kth* column of $\tilde{A}^{(k-1)}$ from the *kth* row to the *nth* row is searched for the first nonzero entry.

If $a_{pk}^{(k)} \neq 0$ for some p,with k + 1 ≤ p ≤ n, then the operation $(E_k) \leftrightarrow (E_p)$ is performed to obtain $\tilde{A}^{(k-1)'}$. The procedure can then be continued to form $\tilde{A}^{(k)}$, and so on.

If $a_{pk}^{(k)} = 0$ for each p, it can be shown that the linear system does not have a unique solution and the procedure stops.

Finally, if $a_{nn}^{(n)} = 0$, the linear system does not have a unique solution, and again the procedure stops.

Algorithm 6.1 summarizes *Gaussian elimination with backward substitution.*

The algorithm incorporates pivoting when one of the pivots $a_{kk}^{(k)}$ is 0 by interchanging the kth row with the pth row, where p is the smallest integer greater than k for which $a_{pk}^{(k)} \neq 0$.

To solve the $n \times n$ linear system

$$\begin{aligned}
E_1 &: a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1}, \\
E_2 &: a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1}, \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
E_n &: a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = a_{n,n+1},
\end{aligned}$$

INPUT number of unknowns and equations n; augmented matrix $A = [a_{ij}]$, where $1 \le i \le n$ and $1 \le j \le n + 1$.

OUTPUT solution $x_1, x_2, \ldots, x_n$ or message that the linear system has no unique solution.

Algorithm 6.1

Step 1 For i = 1, ... , n − 1 do Steps 2–4. (*Elimination process.*)

Step 2 Let p be the smallest integer with $i \le p \le n$ and $a_{pi} \neq 0$.

If no integer p can be found
then OUTPUT ('no unique solution exists');
STOP.

Step 3 If $p \neq i$ then perform $(E_p) \leftrightarrow (E_i)$

Step 4 For $j = i + 1, \ldots, n$ do Steps 5 and 6.

Step 5 Set $m_{ji} = a_{ji}/a_{ii}$.

Step 6 Perform $(E_j - m_{ji}E_i) \rightarrow (E_j)$;

Algorithm 6.1

Step 7 If $a_{nn} = 0$ then OUTPUT ('no unique solution exists');
STOP.

Step 8 Set $x_n = a_{n,n+1}/a_{nn}$. (*Start backward substitution*.)

Step 9 For $i = n - 1, \ldots, 1$ set $x_i = \left[a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j\right]/a_{ii}$

Step 10 OUTPUT $(x_1, \ldots, x_n)$; (*Procedure completed successfully*.)
STOP.

Illustration

The purpose of this illustration is to show what can happen if **Algorithm 6.1** fails. The computations will be done simultaneously on two linear systems:

$$\begin{aligned} x_1 + x_2 + x_3 &= 4, \\ 2x_1 + 2x_2 + x_3 &= 6, \\ x_1 + x_2 + 2x_3 &= 6, \end{aligned} \qquad \text{and} \qquad \begin{aligned} x_1 + x_2 + x_3 &= 4, \\ 2x_1 + 2x_2 + x_3 &= 4, \\ x_1 + x_2 + 2x_3 &= 6. \end{aligned}$$

These systems produce the augmented matrices

$$\tilde{A} = \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 2 & 2 & 1 & 6 \\ 1 & 1 & 2 & 6 \end{array}\right] \qquad \text{and} \qquad \tilde{A} = \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 2 & 2 & 1 & 4 \\ 1 & 1 & 2 & 6 \end{array}\right].$$

Illustration

$$\begin{aligned} x_1 + x_2 + x_3 &= 4, \\ 2x_1 + 2x_2 + x_3 &= 6, \\ x_1 + x_2 + 2x_3 &= 6, \end{aligned} \quad \text{and} \quad \begin{aligned} x_1 + x_2 + x_3 &= 4, \\ 2x_1 + 2x_2 + x_3 &= 4, \\ x_1 + x_2 + 2x_3 &= 6. \end{aligned}$$

Since $a_{11} = 1$, we perform $(E_2 - 2E_1) \rightarrow (E_2)$ and $(E_3 - E_1) \rightarrow (E_3)$ to produce

$$\tilde{A} = \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 1 & 2 \end{array}\right] \quad \text{and} \quad \tilde{A} = \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 0 & 0 & -1 & -4 \\ 0 & 0 & 1 & 2 \end{array}\right].$$

At this point, $a_{22} = a_{32} = 0$. The algorithm requires that the procedure be halted, and no solution to either system is obtained.Writing the equations for each system gives

$$\begin{aligned} x_1 + x_2 + x_3 &= 4, \\ -x_3 &= -2, \\ x_3 &= 2, \end{aligned} \quad \text{and} \quad \begin{aligned} x_1 + x_2 + x_3 &= 4, \\ -x_3 &= -4, \\ x_3 &= 2. \end{aligned}$$

Illustration

The first linear system has an infinite number of solutions, which can be described by $x_3 = 2$, $x_2 = 2 - x_1$, and x_1 arbitrary.

The second system leads to the contradiction $x_3 = 2$ and $x_3 = 4$, so no solution exists. In each case, however, there is no unique solution, as we conclude from Algorithm 6.1.

Although **Algorithm 6.1** can be viewed as the construction of the augmented matrices $\tilde{A}^{(1)}, \ldots, \tilde{A}^{(n)}$, the computations can be performed in a computer using only one $n \times (n+1)$ array for storage. At each step we simply replace the previous value of a_{ij} by the new one.

Algorithm 6.1

In addition, we can store the multipliers m_{ji} in the locations of a_{ji} because a_{ji} has the value 0 for each $i = 1, 2, \ldots, n-1$ and $j = i+1, i+2, \ldots, n$.

Thus A can be overwritten by the multipliers in the entries that are below the main diagonal (that is, the entries of the form a_{ji}, with $j > i$) and by the newly computed entries of $\tilde{A}^{(n)}$ on and above the main diagonal (the entries of the form a_{ij}, with $j \leq i$).

These values can be used to solve other linear systems involving the original matrix A, as we will see in Section 6.5.

Operation Counts

Both the amount of time required to complete the calculations and the subsequent round-off error depend on the number of floating-point arithmetic operations needed to solve a routine problem. In general, the amount of time required to perform a *multiplication or division* on a computer is approximately the same and is considerably greater than that required to perform an *addition or subtraction*.

The actual differences in execution time, however, depend on the particular computing system. To demonstrate the counting operations for a given method, we will count the operations required to solve a typical linear system of n equations in *n* unknowns using Algorithm 6.1.

Operation Counts

We will keep the count of the *additions/subtractions* separate from the count of the *multiplications/divisions* because of the time differential.

No arithmetic operations are performed until Steps 5 and 6 in the algorithm.
Step 5 requires that $(n - i)$ divisions be performed. The replacement of the equation E_j by $(E_j - m_{ji}E_i)$ in Step 6 requires that m_{ji} be multiplied by each term in E_i, resulting in a total of $(n - i)(n - i + 1)$ multiplications.

After this is completed, each term of the resulting equation is subtracted from the corresponding term in E_j.This requires $(n - i)(n - i + 1)$ subtractions. For each $i = 1, 2, \ldots, n - 1$, the operations required in Steps 5 and 6 are as follows.

Multiplications/divisions

$$(n-i)+(n-i)(n-i+1)=(n-i)(n-i+2).$$

Additions/subtractions

$$(n-i)(n-i+1).$$

The total number of operations required by Steps 5 and 6 is obtained by summing the operation counts for each i. Recalling from calculus that

$\sum_{j=1}^{m} 1 = m, \sum_{j=1}^{m} j = \frac{m(m+1)}{2}$, and $\sum_{j=1}^{m} j^2 = \frac{m(m+1)(2m+1)}{6}$

we have the following operation counts:

Multiplications/divisions

$$\sum_{i=1}^{n-1}(n-i)(n-i+2)=\sum_{i=1}^{n-1}(n^2-2ni+i^2+2n-2i)$$

$$=\sum_{i=1}^{n-1}(n-i)^2+2\sum_{i=1}^{n-1}(n-i)=\sum_{i=1}^{n-1}i^2+2\sum_{i=1}^{n-1}i$$

$$=\frac{(n-1)n(2n-1)}{6}+2\frac{(n-1)n}{2}=\frac{2n^3+3n^2-5n}{6}$$

Additions/subtractions

$$\sum_{i=1}^{n-1}(n-i)(n-i+1)=\sum_{i=1}^{n-1}(n^2-2ni+i^2+n-i)$$

$$=\sum_{i=1}^{n-1}(n-i)^2+\sum_{i=1}^{n-1}(n-i)=\sum_{i=1}^{n-1}i^2+\sum_{i=1}^{n-1}i$$

$$=\frac{(n-1)n(2n-1)}{6}+\frac{(n-1)n}{2}=\frac{n^3-n}{3}$$

The only other steps in **Algorithm 6.1** that involve arithmetic operations are those required for backward substitution, Steps 8 and 9. Step 8 requires one division. Step 9 requires $(n-i)$ multiplications and $(n-i-1)$ additions for each summation term and then one subtraction and one division. The total number of operations in Steps 8 and 9 is as follows.

Multiplications/divisions

$$1+\sum_{i=1}^{n-1}((n-i)+1)=1+(\sum_{i=1}^{n-1}(n-i))+n-1$$

$$=n+\sum_{i=1}^{n-1}(n-i)=n+\sum_{i=1}^{n-1}i=\frac{n+n^2}{2}$$

Additions/subtractions

$$\sum_{i=1}^{n-1}((n-i-1)+1)=\sum_{i=1}^{n-1}(n-i)=\sum_{i=1}^{n-1}i=\frac{n^2-n}{2}$$

The total number of arithmetic operations in **Algorithm 6.1** is, therefore:

Multiplications/divisions

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

Additions/subtractions

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

For large n, the total number of multiplications and divisions is approximately $n^3/3$, as is the total number of additions and subtractions. Thus the amount of computation and the time required increases with n in proportion to n^3, as shown in **Table 6.1.**

Table 6.1

n	Multiplications/Divisions	Additions/Subtractions
3	17	11
10	430	375
50	44,150	42,875
100	343,300	338,250

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 6-2

Pivoting Strategies

Algorithm 6.1

In deriving Algorithm 6.1, we found that a row interchange was needed when one of the pivot elements $a_{kk}^{(k)}$ is 0.

This row interchange has the form $(E_k) \leftrightarrow (E_p)$, where p is the smallest integer greater than k with $a_{pk}^{(k)} \neq 0$.

To reduce round-off error, it is often necessary to perform row interchanges even when the pivot elements are not zero.

If $a_{kk}^{(k)}$ is small in magnitude compared to $a_{jk}^{(k)}$, then the magnitude of the multiplier

$$m_{jk} = \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}}$$

will be much larger than 1.

Round-off error introduced in the computation of one of the terms $a_{kl}^{(k)}$ is multiplied by m_{jk} when computing $a_{kl}^{(k+1)}$, which compounds the original error.

Also, when performing the backward substitution for

$$x_k = \frac{a_{k,n+1}^{(k)} - \sum_{j=k-1}^{m} a_{kj}^{(k)}}{a_{kk}^{(k)}}$$

$$x_k = \frac{a_{k,n+1}^{(k)} - \sum_{j=k-1}^{m} a_{kj}^{(k)}}{a_{kk}^{(k)}}$$

with a small value of $a_{kk}^{(k)}$, any error in the numerator can be dramatically increased because of the division by $a_{kk}^{(k)}$.

In our next example, we will see that even for small systems, round-off error can dominate the calculations.

Example 1

Apply Gaussian elimination to the system

$$E_1 : 0.003000x_1 + 59.14x_2 = 59.17$$

$$E_2 : \quad 5.291x_1 - 6.130x_2 = 46.78,$$

using four-digit arithmetic with rounding, and compare the results to the exact solution $x_1 = 10.00$ and $x_2 = 1.000$.

Solution The first pivot element, $a_{11}^{(1)} = 0.003000$, is small, and its associated multiplier,

$$m_{21} = \frac{5.291}{0.003000} = 1763.\overline{66}$$

rounds to the large number 1764.

Example 1

$$E_1 : 0.003000x_1 + 59.14x_2 = 59.17$$
$$E_2 : \quad 5.291x_1 - 6.130x_2 = 46.78,$$

Performing $(E_2 - m_{21}E_1) \rightarrow (E_2)$ and the appropriate rounding gives the system

$$0.003000x_1 + 59.14x_2 \approx 59.17$$
$$-104300x_2 \approx 104400$$

instead of the exact system, which is

$$0.003000x_1 + 59.14x_2 = 59.17$$
$$-104309.37\overline{6}x_2 = 104309.37\overline{6}$$

The disparity in the magnitudes of $m_{21}a_{13}$ and a_{23} has introduced round-off error, but the round-off error has not yet been propagated.

Example 1

$$0.003000x_1 + 59.14x_2 \approx 59.17$$
$$-104300x_2 \approx 104400$$

Backward substitution yields

$$x_2 \approx 1.001,$$

which is a close approximation to the actual value, $x_2 = 1.000$.

However, because of the small pivot $a_{11} = 0.003000$,

$$x_1 \approx \frac{59.17 - (59.14)(1.001)}{0.003000} = -10.00$$

contains the small error of 0.001 multiplied by

$$\frac{59.14}{0.003000} \approx 20000$$

This ruins the approximation to the actual value $x_1 = 10.00$.

This is clearly a contrived example and the graph in **Figure 6.1**. shows why the error can so easily occur. For larger systems it is much more difficult to predict in advance when devastating round-off error might occur.

Figure 6.1

Partial Pivoting

Example 1 shows how difficulties can arise when the pivot element $a_{kk}^{(k)}$ is small relative to the entries $a_{ij}^{(k)}$, for $k \le i \le n$ and $k \le j \le n$.

To avoid this problem, pivoting is performed by selecting an element $a_{pq}^{(k)}$ with a *larger magnitude* as the pivot, and interchanging the *kth* and *pth* rows. This can be followed by the interchange of the *kth* and *qth* columns, if necessary.

The simplest strategy is to select an element in the same column that is below the diagonal and has the largest absolute value; specifically, we determine the smallest p ≥ k such that $\left|a_{pk}^{(k)}\right| = \max_{k \le i \le n} \left|a_{ik}^{(k)}\right|$ and perform $(E_k) \leftrightarrow (E_p)$.In this case no interchange of columns is used.

Example 2 Apply Gaussian elimination to the system

$$E_1 : 0.003000x_1 + 59.14x_2 = 59.17$$

$$E_2 : \quad 5.291x_1 - 6.130x_2 = 46.78,$$

using partial pivoting and four-digit arithmetic with rounding, and compare the results to the exact solution $x_1 = 10.00$ and $x_2 = 1.000$.

Solution The partial-pivoting procedure first requires finding

$$max\left\{\left|a_{11}^{(1)}\right|, \left|a_{21}^{(1)}\right|\right\} = max\{|0.003000|, |5.291|\} = |5.291| = \left|a_{21}^{(1)}\right|$$

This requires that the operation $(E_2) \leftrightarrow (E_1)$ be performed to produce the equivalent system

$$E_1 : \quad 5.291x_1 - 6.130x_2 = 46.78,$$

$$E_2 : 0.003000x_1 + 59.14x_2 = 59.17$$

Example 2

$$E_1: \quad 5.291x_1 - 6.130x_2 = 46.78,$$
$$E_2: 0.003000x_1 + 59.14x_2 = 59.17$$

The multiplier for this system is

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = 0.0005670$$

and the operation $(E_2 - m_{21}E_1) \rightarrow (E_2)$ reduces the system to

$$5.291x_1 - 6.130x_2 \approx 46.78,$$
$$59.14x_2 \approx 59.14.$$

The four-digit answers resulting from the backward substitution are the correct values $x_1 = 10.00$ and $x_2 = 1.000$.

The technique just described is called *partial pivoting* (or maximal column pivoting) and is detailed in Algorithm 6.2.

The actual row interchanging is simulated in the algorithm by interchanging the values of NROW in Step 5.

Gaussian Elimination with Partial Pivoting

To solve the $n \times n$ linear system

$$\begin{aligned}
E_1: &\quad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\
E_2: &\quad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\
&\qquad\vdots \qquad\qquad\qquad\qquad \vdots \\
E_n: &\quad a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = a_{n,n+1}
\end{aligned}$$

Algorithm 6.2

INPUT number of unknowns and equations n; augmented matrix $A = [a_{ij}]$ where $1 \le i \le n$ and $1 \le j \le n + 1$.

OUTPUT solution $x_1, \ldots, x_n$ or message that the linear system has no unique solution.

Step 1 For i = 1, ... , n set NROW(i) = i. (*Initialize row pointer.*)

Step 2 For i = 1, ... , $n - 1$ do Steps 3–6. (*Elimination process.*)

Step 3 Let p be the smallest integer with $i \le p \le n$ and

$$|a(NROW(p), i)| = \max_{i \le j \le n} |a(NROW(j), i)|.$$

(*Notation:* $a(NROW(i), j) \equiv a_{NROWi,\, j}$.)

Step 4 If $a(NROW(p), i) = 0$ then OUTPUT ('no unique solution exists'); STOP.

Algorithm 6.2

Step 5 If $NROW(i) \neq NROW(p)$ then set $NCOPY = NROW(i)$;
$NROW(i) = NROW(p)$;
$NROW(p) = NCOPY$.

(Simulated row interchange.)

Step 6 For $j = i + 1, \ldots, n$ do Steps 7 and 8.

Step 7 Set $m(NROW(j), i) = a(NROW(j), i)/a(NROW(i), i)$.

Step 8 Perform $(ENROW(j) - m(NROW(j), i) \cdot ENROW(i)) \rightarrow (ENROW(j))$

Step 9 If $a(NROW(n), n) = 0$ then OUTPUT ('no unique solution exists');
STOP.

Algorithm 6.2

Step 10 Set $x_n = a(NROW(n), n + 1)/a(NROW(n), n)$.

(Start backward substitution.)

Step 11 For $i = n - 1, \ldots, 1$

$$\text{set } x_i = \frac{a(NROW(i), n+1) - \sum_{j=i+1}^{n} a(NROW(i), j) \cdot xj}{a(NROW(i), i)}$$

Step 12 OUTPUT (x1, ... , xn); (*Procedure completed successfully.*)

STOP.

Each multiplier m_{ji} in the partial pivoting algorithm has magnitude less than or equal to 1. Although this strategy is sufficient for many linear systems,situations do arise when it is inadequate.

Illustration The linear system

$$E_1 : 30.00x_1 + 591400x_2 = 591700$$

$$E_2 : 5.291x_1 - \quad 6.130x_2 = 46.78,$$

is the same as that in **Examples 1 and 2** except that all the entries in the first equation have been multiplied by 10^4.

The partial pivoting procedure described in Algorithm 6.2 with four-digit arithmetic leads to the same results as obtained in Example 1.The maximal value in the first column is 30.00, and the multiplier

$$m_{21} = \frac{5.291}{30.00} = 0.1764,$$

Illustration

$$E_1 : 30.00x_1 + 591400x_2 = 591700$$
$$E_2 : 5.291x_1 - \quad 6.130x_2 = 46.78,$$

leads to the system

$$30.00x_1 + 591400x_2 \approx 591700$$
$$-104300x_2 \approx 104400$$

which has the same inaccurate solutions as in Example 1: $x_2 \approx 1.001$ and $x_1 \approx -10.00$.

Scaled Partial Pivoting

Scaled partial pivoting (or *scaled-column pivoting*) is needed for the system in the Illustration. It places the element in the pivot position that is largest relative to the entries in its row.

The first step in this procedure is to define a scale factor s_i for each row as

$$s_i = \max_{1 \le j \le n} |a_{ij}| ,$$

If we have $s_i = 0$ for some i, then the system has no unique solution since all entries in the *ith* row are 0.

Scaled Partial Pivoting

Assuming that this is not the case, the appropriate row interchange to place zeros in the first column is determined by choosing the least integer p with

The first step in this procedure is to define a scale factor s_i for each row as

$$\frac{|a_{p1}|}{s_p} = \max_{1 \le k \le n} \frac{|a_{k1}|}{s_k}$$

and performing $(E_1) \leftrightarrow (E_p)$.

The effect of scaling is to ensure that the largest element in each row has a relative magnitude of 1 before the comparison for row interchange is performed.

Scaled Partial Pivoting

In a similar manner, before eliminating the variable x_i using the operations

$$E_k - m_{ki}E_i, \text{ for } k = i + 1, \dots, n$$

we select the smallest integer $p \geq i$ with

$$\frac{|a_{pi}|}{s_p} = \max_{i \leq k \leq n} \frac{|a_{ki}|}{s_k}$$

and perform the row interchange $(E_i) \leftrightarrow (E_p)$ if $i \neq p$.

The scale factors $s_1, \dots, s_n$ are computed only once, at the start of the procedure. They are row dependent, so they must also be interchanged when row interchanges are performed.

$E_1 : 30.00x_1 + 591400x_2 = 591700$
$E_2 : 5.291x_1 - \quad 6.130x_2 = 46.78,$

Illustration Applying scaled partial pivoting to the previous Illustration gives

$$s_1 = \max\{|30.00|, |591400|\} = 591400$$

$$s_2 = \max\{|5.291|, |-6.130|\} = 6.130.$$

Consequently

$$\frac{|a_{11}|}{s_1} = \frac{30.00}{591400} = 0.5073 \times 10^{-4}, \qquad \frac{|a_{21}|}{s_2} = \frac{5.291}{6.130} = 0.8631$$

and the interchange $(E_1) \leftrightarrow (E_2)$ is made.

Applying *Gaussian elimination* to the new system

$$5.291x_1 - \quad 6.130x_2 = 46.78$$
$$30.00x_1 + 591400x_2 = 591700$$

produces the correct results: $x_1 = 10.00$ and $x_2 = 1.000$.

Algorithm 6.3 *Gaussian Elimination with Scaled Partial Pivoting*

Algorithm 6.3 implements scaled partial pivoting.

The only steps in this algorithm that differ from those of Algorithm 6.2 are:

Step 1 For $i = 1, \ldots, n-1$ t $s_i = max_{1 \le j \le n} |a_{ij}|$;

If $s_i = 0$ then OUTPUT ('no unique solution exists');
STOP.

Step 2 For $i = 1, \ldots, n-1$ do Steps 3–6. (*Elimination process*.)

Step 3 Let p be the smallest integer with $i \le p \le n$ and

$$\frac{|a(NROW(p), i)|}{s(NROW(p))} = \max_{1 \le j \le n} \frac{|a(NROW(j), i)|}{s(NROW(j))}$$

The first additional computations required for scaled partial pivoting result from the determination of the scale factors; there are $(n-1)$ comparisons for each of the n rows, for a total of $n(n-1)$ *comparisons*.

To determine the correct first interchange, n divisions are performed, followed by n−1 comparisons. So the first interchange determination adds *n divisions and* $(n-1)$ *comparisons*

The scaling factors are computed only once, so the second step requires $(n-1)$ *divisions and* $(n-2)$ *comparisons.*

We proceed in a similar manner until there are zeros below the main diagonal in all but the *nth* row. The final step requires that we perform *2 divisions and 1 comparison.*

As a consequence, scaled partial pivoting adds a total of

$$n(n-1)+\sum_{k=1}^{n-1}k = n(n-1)+\frac{(n-1)n}{2}=\frac{3}{2}n(n-1)\ \textit{comparisons} \qquad (6.7)$$

and

$$\sum_{k=2}^{n}k=\left(\sum_{k=1}^{n}k\right)-1=\frac{n(n+1)}{2}-1=\frac{1}{2}(n-1)(n+2) \qquad \textit{divisions}$$

to the Gaussian elimination procedure.

The time required to perform a *comparison* is about the same as an *addition/subtraction.*

Since the total time to perform the basic *Gaussian elimination* procedure is $O(n^3/3)$ multiplications/divisions and $O(n^3/3)$ additions/subtractions, scaled partial pivoting does not add significantly to the computational time required to solve a system for large values of n.

To emphasize the importance of choosing the scale factors only once, consider the amount of additional computation that would be required if the procedure were modified so that new scale factors were determined each time a row interchange decision was to be made.

$$n(n-1)+\sum_{k=1}^{n-1}k = n(n-1) + \frac{(n-1)n}{2} = \frac{3}{2}n(n-1) \ \textit{comparisons} \qquad (6.7)$$

In this case, the term $n(n-1)$ in Eq. (6.7) would be replaced by

$$\sum_{k=2}^{n} k(k-1) = \frac{1}{3}n(n^2-1)$$

As a consequence, this pivoting technique would add $O(n^3/3)$ comparisons, in addition to the $[n(n+1)/2]-1$ divisions.

Complete Pivoting

Pivoting can incorporate the interchange of both rows and columns.

Complete (or maximal) pivoting at the *kth* step searches all the entries a_{ij}, for $i = k, k+1, \ldots, n$ and $j = k, k+1, \ldots, n$, to find the entry with the largest magnitude.

Both row and column interchanges are performed to bring this entry to the pivot position.The first step of total pivoting requires that $n^2 - 1$ comparisons be performed, the second step requires $(n-1)^2 - 1$ comparisons, and so on.

Complete Pivoting

The total additional time required to incorporate complete pivoting into Gaussian elimination is $\sum_{k=2}^{n}(k^2-1) = \frac{n(n-1)(2n+5)}{6}$ *comparisons.*

Complete pivoting is, consequently, the strategy recommended only for systems where accuracy is essential and the amount of execution time needed for this method can be justified.

The key result relating nonsingularity, Gaussian elimination, linear systems, and determinants is that the following statements are equivalent.

Theorem 6.17 The following statements are equivalent for any $n \times n$ matrix A:

(i) The equation A**x** = **0** has the *unique solution* $\boldsymbol{x} = \boldsymbol{0}$.

(ii) The system A**x** = **b** has a *unique solution* for any n-dimensional column vector **b**.

(iii) The matrix A is *nonsingular*; that is, A^{-1} *exists*.

(iv) $\det A \neq 0$.

(v) Gaussian elimination with row interchanges can be performed on the system A**x** = **b** for any n-dimensional column vector **b**.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 6-5

Matrix Factorization

Gaussian elimination is the principal tool in the direct solution of linear systems of equations, so it should be no surprise that it appears in other guises.

In this section we will see that the steps used to solve a system of the form $A\mathbf{x} = \mathbf{b}$ can be used to factor a matrix.
The factorization is particularly useful when it has the form $A = LU$, where *L is lower triangular and U is upper triangular.*

Although not all matrices have this type of representation, many do that occur frequently in the application of numerical techniques.

In Section 6.1 we found that Gaussian elimination applied to an arbitrary linear system $A\boldsymbol{x} = \boldsymbol{b}$ requires $O(n^3/3)$ arithmetic operations to determine $\mathbf{x}$.

However, to solve a linear system that involves an upper-triangular system requires only backward substitution, which takes $O(n^2)$ operations.

The number of operations required to solve a lower-triangular systems is similar.

Suppose that A has been factored into the triangular form $A = LU$, where L is lower triangular and U is upper triangular.

Then we can solve for $\mathbf{x}$ more easily by using a two-step process.

First we let $\mathbf{y} = U\mathbf{x}$ and solve the lower triangular system $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$. Since L is triangular, determining $\mathbf{y}$ from this equation requires only $O(n^2)$ operations.

Once y is known, the upper triangular system $U\mathbf{x} = \mathbf{y}$ requires only an additional $O(n^2)$ operations to determine the solution $\mathbf{x}$.

Solving a linear system $\boldsymbol{A}\mathbf{x}=\mathbf{b}$ in factored form means that the number of operations needed to solve the system $\boldsymbol{A}\mathbf{x}=\mathbf{b}$ in factored form means that the number of operations is reduced from $O(n^3/3)$ to $O(2n^2)$.

Example 1

Compare the approximate number of operations required to determine the solution to a linear system using a technique requiring $O(n^3/3)$ operations and one requiring $O(2n^2)$ when $n = 20$, $n = 100$, and $n = 1000$.

Solution Table 6.3 gives the results of these calculations.

Table 6.3

n	$n^3/3$	$2n^2$	% Reduction
10	$3.\overline{3} \times 10^2$	2×10^2	40
100	$3.\overline{3} \times 10^5$	2×10^4	94
1000	$3.\overline{3} \times 10^8$	2×10^6	99.4

Not surprisingly, the reductions from the factorization come at a cost; determining the specific matrices L and U requires $O(n^3/3)$ operations.

But once the factorization is determined, systems involving the matrix A can be solved in this simplified manner for any number of vectors **b**.

To see which matrices have an LU factorization and to find how it is determined, first suppose that Gaussian elimination can be performed on the system $\boldsymbol{Ax} = \boldsymbol{b}$ without row interchanges. this is equivalent to having nonzero pivot elements $a_{ii}^{(i)}$, for each $i = 1, 2, \ldots , n$.

The first step in the *Gaussian elimination* process consists of performing, for each $j = 2, 3, \ldots, n$, the operations

$$\left(E_j - m_{j,1}E_1\right) \rightarrow \left(E_j\right), \qquad \text{where} \quad m_{j,1} = \frac{a_{j1}^{(1)}}{a_{11}^{(1)}} \qquad (6.8)$$

These operations transform the system into one in which all the entries in the first column below the diagonal are zero.

$$\left(E_j - m_{j,1}E_1\right) \to \left(E_j\right), \qquad \text{where} \quad m_{j,1} = \frac{a_{j1}^{(1)}}{a_{11}^{(1)}} \qquad (6.8)$$

The system of operations in (6.8) can be viewed in another way.
It is simultaneously accomplished by multiplying the original matrix A on the left by the matrix

$$M^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -m_{21} & 1 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -m_{n1} & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

This is called ***the first Gaussian transformation matrix***.
We denote the product of this matrix with $A^{(1)} \equiv A$ by $A^{(2)}$ and with b by $b^{(2)}$, so

$$A^{(2)}\mathbf{x} = M^{(1)}A\mathbf{x} = M^{(1)}\mathbf{b} = b^{(2)}.$$

In a similar manner we construct $M^{(2)}$, the identity matrix with the entries below the diagonal in the second column replaced by the negatives of the multipliers

$$m_{j,2} = \frac{a_{j2}^{(2)}}{a_{22}^{(2)}}$$

The product of this matrix with $A^{(2)}$ has zeros below the diagonal in the first two columns, and we let

$$A^{(3)}\mathbf{x} = M^{(2)}A^{(2)}\mathbf{x} = M^{(2)}M^{(1)}A\mathbf{x} = M^{(2)}M^{(1)}\mathbf{b} = \mathbf{b}^{(3)}.$$

In general, with $A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$ already formed, multiply by the *kth* ***Gaussian transformation matrix***

$$M^{(k)} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & 0 & \ddots & & & \vdots \\ \vdots & & \vdots & -m_{k+1,k} & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & -m_{n,k} & 0 & \cdots & 1 \end{bmatrix},$$

to obtain

$$A^{(k+1)}x = M^{(k)}A^{(k)}x = M^{(k)}\cdots M^{(1)}Ax = M^{(k)}b^{(k)} = b^{(k+1)} = M^{(k)}\cdots M^{(1)}b. \qquad (6.9)$$

The process ends with the formation of $\mathbf{A}^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$, where $A^{(n)}$ is the upper triangular matrix

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

given by $\quad A^{(n)} = M^{(n-1)}M^{(n-2)} \cdots M^{(1)}A$

This process forms the $U = A^{(n)}$ portion of the matrix factorization $A = LU$.

$$A^{(k+1)}x = M^{(k)}A^{(k)}x = M^{(k)}\cdots M^{(1)}Ax = M^{(k)}b^{(k)} = b^{(k+1)} = M^{(k)}\cdots M^{(1)}b. \qquad (6.9)$$

To determine the complementary lower triangular matrix L, first recall the multiplication of $A^{(k)}\mathbf{x} = b^{(k)}$ by the Gaussian transformation of $M^{(k)}$ used to obtain (6.9):

$$A^{(k+1)}\mathbf{x} = M^{(k)}A^{(k)}\mathbf{x} = M^{(k)}b^{(k)} = b^{(k+1)},$$

where $M^{(k)}$ generates the row operations

$$\left(E_j - m_{j,k}E_k\right) \rightarrow \left(E_j\right), \qquad \text{for} \qquad j = k+1, \ldots, n.$$

To reverse the effects of this transformation and return to $A^{(k)}$ requires that the operations $(E_j + m_{j,k}E_k) \rightarrow (E_j)$ be performed for each $j = k + 1, \ldots, n$.

This is equivalent to multiplying by the inverse of the matrix $M^{(k)}$, the matrix

$$L^{(k)} = [M^{(k)}]^{-1} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & m_{k+1,k} & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & m_{n,k} & 0 & \cdots & 1 \end{bmatrix}.$$

The lower-triangular matrix L in the factorization of A, then, is the product of the matrices $L^{(k)}$:

$$L = L^{(1)}L^{(2)}\cdots L^{(n-1)} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ m_{21} & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \\ m_{n1} & \cdots & \cdots & m_{n,n-1} & 1 \end{bmatrix},$$

since the product of L with the upper-triangular matrix $U = M^{(n-1)} \cdots M^{(2)}M^{(1)}A$

gives
$$\begin{aligned} LU &= L^{(1)}L^{(2)} \ldots L^{(n-3)}L^{(n-2)}L^{(n-1)}.M^{(n-1)}M^{(n-2)}M^{(n-3)} \ldots M^{(2)}M^{(1)}A \\ &= \left[M^{(1)}\right]^{-1}\left[M^{(2)}\right]^{-1} \ldots \left[M^{(n-2)}\right]^{-1}\left[M^{(n-1)}\right]^{-1}M^{(n-1)}M^{(n-2)}M^{(n-3)} \ldots M^{(2)}M^{(1)}A \\ &= A \end{aligned}$$

Theorem 6.19 follows from these observations.

Theorem 6.19

If Gaussian elimination can be performed on the linear system A**x** = **b** without row interchanges, then the matrix A can be factored into the product of a lower-triangular matrix L and an upper-triangular matrix U,

that is, A = LU, where $m_{ji} = a_{ji}^{(i)}/a_{ii}^{(i)}$,

$$U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix}, \quad \text{and} \quad L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{n,n-1} & 1 \end{bmatrix}.$$

Example 2

(a) Determine the LU factorization for matrix A in the linear system A**x** = **b**, where

$$A = \begin{bmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 4 \\ 1 \\ -3 \\ 4 \end{bmatrix}$$

(b) Then use the factorization to solve the system

$$\begin{aligned} x_1 + x_2 \phantom{{}-x_3} + 3x_4 &= 8, \\ 2x_1 + x_2 - x_3 + x_4 &= 7, \\ 3x_1 - x_2 - x_3 + 2x_4 &= 14, \\ -x_1 + 2x_2 + 3x_3 - x_4 &= -7. \end{aligned}$$

Solution

(a) The sequence of operations $(E_2 - 2E_1) \to (E_2)$, $(E_3 - 3E_1) \to (E_3)$, $(E_4 - (-1)E_1) \to (E_4)$,$(E_3 - 4E_2) \to (E_3)$, $(E_4 - (-3)E_2) \to (E_4)$ converts the system to the triangular system

$$\begin{aligned} x_1 + x_2 \qquad\quad + 3x_4 &= 4, \\ -x_2 - x_3 - 5x_4 &= -7, \\ 3x_3 + 13x_4 &= 13, \\ -13x_4 &= -13. \end{aligned}$$

The multipliers m_{ij} and the upper triangular matrix produce the factorization

$$A = \begin{bmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ -1 & -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{bmatrix} = LU$$

Solution

(b) To solve

$$Ax = LUx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ -1 & -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 7 \\ 14 \\ -7 \end{bmatrix}$$

we first introduce the substitution y = $U\mathbf{x}$. Then $\mathbf{b} = L(U\mathbf{x}) = L\mathbf{y}$. That is,

$$Ly = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ -1 & -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 7 \\ 14 \\ -7 \end{bmatrix}$$

Solution

This system is solved for **y** by a simple forward-substitution process:

$$
\begin{aligned}
y_1 &= 8; \\
2y_1 + y_2 &= 7, \qquad \text{so } y_2 = 7 - 2y_1 = -9; \\
3y_1 + 4y_2 + y_3 &= 14, \qquad \text{so } y_3 = 14 - 3y_1 - 4y_2 = 26; \\
-y_1 - 3y_2 + y_4 &= -7, \qquad \text{so } y_4 = -7 + y_1 + 3y_2 = -26.
\end{aligned}
$$

We then solve $U\mathbf{x} = \mathbf{y}$ for **x**, the solution of the original system; that is,

$$
\begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}
=
\begin{bmatrix} 8 \\ -9 \\ 26 \\ -26 \end{bmatrix}
$$

Using backward substitution we obtain $x_4 = 2$, $x_3 = 0$, $x_2 = -1$, $x_1 = 3$.

The factorization used in **Example 2** is called Doolittle's method and requires that 1s be on the diagonal of L, which results in the factorization described in **Theorem 6.19.**

A general procedure for factoring matrices into a product of triangular matrices is contained in **Algorithm 6.4**.

Although new matrices L and U are constructed, the generated values can replace the corresponding entries of A that are no longer needed.

Algorithm 6.4 LU Factorization

To factor the n×n matrix A = $[a_{ij}]$ into the product of the lower-triangular matrix L = $[l_{ij}]$and the upper-triangular matrix U = $[u_{ij}]$; that is, A = LU, where the main diagonal of either L or U consists of all ones:

INPUT dimension n; the entries a_{ij}, $1 \le i, j \le n$ of A; the diagonal $l_{11} = \cdots = l_{nn} = 1$ of L or the diagonal $u_{11} = \cdots = u_{nn} = 1$ of U.

OUTPUT the entries l_{ij}, $1 \le j \le i$, $1 \le i \le n$ of L and the entries, u_{ij}, $i \le j \le n$, $1 \le i \le n$ of U.

Step 1 Select l_{11} and u_{11} satisfying $l_{11}u_{11} = a_{11}$.

If $l_{11}u_{11} = 0$ then OUTPUT (*'Factorization impossible'*);
STOP.

Algorithm 6.4

Step 2 For $j = 2, \ldots, n$ set $u_{1j} = a_{1j}/l_{11}$; (*First row of U.*)

$l_{j1} = a_{j1}/u_{11}$. (*First column of L.*)

Step 3 For $i = 2, \ldots, n-1$ do Steps 4 and 5.

Step 4 Select l_{ii} and u_{ii} satisfying $l_{ii}u_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{ki}$

If $l_{ii}u_{ii} = 0$ then OUTPUT (*'Factorization impossible'*); STOP.

Step 5 For $j = i+1, \ldots, n$

set $u_{ij} = \frac{1}{l_{ii}}\left[a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}\right]$; (*ith row of U.*)

$l_{ji} = \frac{1}{u_{ii}}\left[a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki}\right]$. (*ith column of L.*)

Algorithm 6.4

Step 6 Select l_{nn} and u_{nn} satisfying $l_{nn}u_{nn}=a_{nn}-\sum_{k=1}^{i-1} l_{nk}u_{kn}$.

(Note: If $l_{nn}u_{nn} = 0$, then $A = LU$ but A is singular.)

Step 7 OUTPUT (l_{ij} for $j = 1, \ldots, i$ and $i = 1, \ldots, n$);

OUTPUT (u_{ij} for $j = i, \ldots, n$ and $i = 1, \ldots, n$);

STOP.

Once the matrix factorization is complete, the solution to a linear system of the form $A\mathbf{x} = LU\mathbf{x} = \mathbf{b}$ is found by first letting $\mathbf{y} = U\mathbf{x}$ and solving $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$. Since $\mathbf{L}$ is lower triangular, we have $y_1 = \frac{b_1}{l_{11}}$

and, for each i = 2, 3, ... , n, $y_i = \frac{1}{l_{ii}}\left[b_i - \sum_{j=1}^{i-1} l_{ij} y_j\right].$

After $\mathbf{y}$ is found by this forward-substitution process, the upper-triangular system U$\mathbf{x}$ = $\mathbf{y}$ is solved for $\mathbf{x}$ by backward substitution using the equations

$$x_n = \frac{y_n}{u_{nn}} \quad \text{and} \quad x_i = \frac{1}{u_{ii}}\left[y_i - \sum_{j=i+1}^{n} u_{ij} x_j\right].$$

Permutation Matrices

In the previous discussion we assumed that $A\boldsymbol{x} = \mathbf{b}$ can be solved using Gaussian elimination without row interchanges.

From a practical standpoint, this factorization is useful only when row interchanges are not required to control the round-off error resulting from the use of finite-digit arithmetic.

Fortunately, many systems we encounter when using approximation methods are of this type, but we will now consider the modifications that must be made when row interchanges are required.

We begin the discussion with the introduction of a class of matrices that are used to rearrange, or permute, rows of a given matrix.

An $n \times n$ **permutation matrix** $P = [p_{ij}]$ is a matrix obtained by rearranging the rows of I_n, the identity matrix.This gives a matrix with precisely one nonzero entry in each row and in each column, and each nonzero entry is a 1.

Illustration The matrix

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

is a 3×3 permutation matrix.

Illustration

For any 3 × 3 matrix A, multiplying on the left by P has the effect of interchanging the second and third rows of A:

$$PA = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}.$$

Similarly, multiplying A on the right by P interchanges the second and third columns of A.

Two useful properties of permutation matrices relate to Gaussian elimination, Suppose $k_1, \cdots, k_n$ is a permutation of the integers $1, \cdots, n$ and the permutation matrix $P = (p_{ij})$ is defined by

$$p_{ij} = \begin{cases} 1, & \text{if } j = k_i \\ 0, & \text{otherwise.} \end{cases}$$

Then, (1)PA permutes the rows of A; that is,

$$PA = \begin{bmatrix} a_{k_1 1} & a_{k_1 2} & \cdots & a_{k_1 n} \\ a_{k_2 1} & a_{k_2 2} & \cdots & a_{k_2 n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k_n 1} & a_{k_n 2} & \cdots & a_{k_n n} \end{bmatrix}.$$

(2)P^{-1} exists and $P^{-1} = P^t$.

For any nonsingular matrix A, the linear system $Ax = b$ can be solved by Gaussian elimination, with the possibility of row interchanges.

If we knew the row interchanges that were required to solve the system by Gaussian elimination, we could arrange the original equations in an order that would ensure that no row interchanges are needed.

Hence there is a rearrangement of the equations in the system that permits Gaussian elimination to proceed *without* row interchanges.

This implies that for any *nonsingular matrix* A, a permutation matrix P exists for which the system

$$PAx = Pb$$

can be solved without row interchanges. As a consequence, this matrix PA can be factored into

$$PA = LU,$$

where L is lower triangular and U is upper triangular. Because $P^{-1} = P^t$, this produces the factorization

$$A = P^{-1}LU = (P^tL)U.$$

The matrix U is still upper triangular, but P^tL is not lower triangular unless $P = I$.

Example 3 Determine a factorization in the form $A = (P^tL)U$ for the matrix

$$A = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 1 & 1 & -1 & 2 \\ -1 & -1 & 2 & 0 \\ 1 & 2 & 0 & 2 \end{bmatrix}.$$

Solution

The matrix A cannot have an LU factorization because $a_{11} = 0$. However, using the row interchange $(E_1) \leftrightarrow (E_2)$, followed by $(E_3 + E_1) \rightarrow (E_3)$ and $(E_4 - E_1) \rightarrow (E_4)$, produces

$$\begin{bmatrix} 1 & 1 & -1 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Then the row interchange $(E_2) \leftrightarrow (E_4)$, followed by $(E_4 + E_3) \rightarrow (E_4)$, gives the matrix

$$U = \begin{bmatrix} 1 & 1 & -1 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

The permutation matrix associated with the row interchanges $(E_1) \leftrightarrow (E_2)$ and $(E_2) \leftrightarrow (E_4)$ is

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \text{and} \quad PA = \begin{bmatrix} 1 & 1 & -1 & 2 \\ 1 & 2 & 0 & 2 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Gaussian elimination is performed on PA using the same operations as on A, except without the row interchanges.

That is, $(E_2 - E_1) \rightarrow (E_2)$, $(E_3 + E_1) \rightarrow (E_3)$, followed by $(E_4 + E_3) \rightarrow (E_4)$.

The nonzero multipliers for PA are consequently,

$$m_{21} = 1,\ m_{31} = -1, \text{ and } m_{43} = -1,$$

and the LU factorization of PA is

$$PA = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix} = LU.$$

Multiplying by $P^{-1} = P^t$ produces the factorization

$$A = P^{-1}(LU) = P^t(LU) = (P^tL)U = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 6-6

Special Types of Matrices

We now turn attention to two classes of matrices for which Gaussian elimination can be performed effectively without row interchanges.

Diagonally Dominant Matrices(对角占优矩阵)

Definition 6.20 The $n \times n$ matrix A is said to be diagonally dominant when

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \quad \text{holds for each } i = 1, 2, \cdots, n. \qquad (6.10)$$

A diagonally dominant matrix is said to be strictly diagonally dominant when the inequality in (6.10) is strict for each n,that is, when

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \quad \text{holds for each } i = 1, 2, \cdots, n.$$

Illustration Consider the matrices

$$A = \begin{bmatrix} 7 & 2 & 0 \\ 3 & 5 & -1 \\ 0 & 5 & -6 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 6 & 4 & -3 \\ 4 & -2 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

The nonsymmetric matrix A is strictly diagonally dominant because

$$|7| > |2|+|0|,\ |5| > |3| + |-1|, \text{ and } |-6| > |0|+|5|.$$

The symmetric matrix B is not strictly diagonally dominant because, for example, in the first row the absolute value of the diagonal element is $|6| < |4| + |-3| = 7$.

It is interesting to note that A^t is not strictly diagonally dominant, because the middle row of A^t is [2 5 5], nor, of course, is B^t because $B^t = B$.

The following theorem was used in Section 3.5 to ensure that there are unique solutions to the linear systems needed to determine cubic spline interpolants.

Theorem 6.21

A strictly diagonally dominant matrix A is *nonsingular*. Moreover, in this case, Gaussian elimination can be performed on any linear system of the form $A\mathbf{x} = \mathbf{b}$ to obtain its unique solution *without row or column interchanges*, and the computations will be stable with respect to the growth of round-off errors.

Proof

We first use proof by contradiction to show that A is nonsingular.

Consider the linear system described by A**x** = **0**, and suppose that a nonzero solution **x** = (x_i) to this system exists. Let k be an index for which

$$0 < |x_k| = \max_{1 \le j \le n} |x_j|$$

Because $\sum_{j=1}^{n} a_{ij}\, x_j = 0$ for each $i = 1, 2, \ldots, n$, we have, when $i = k$,

$$a_{kk} x_k = -\sum_{\substack{j=1 \\ j \neq k}}^{n} a_{kj} x_j .$$

From the triangle inequality we have

$$|a_{kk}||x_k| \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}||x_j|, \quad \text{so} \quad |a_{kk}| \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}| \frac{|x_j|}{|x_k|} \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}|$$

This inequality contradicts the strict diagonal dominance of A. Consequently, the only solution to $A\mathbf{x} = \mathbf{0}$ is $\mathbf{x} = \mathbf{0}$. This is shown in **Theorem 6.17** to be equivalent to the nonsingularity of A.

To prove that Gaussian elimination can be performed *without row interchanges*, we show that each of the matrices $A^{(2)}, A^{(3)}, \ldots, A^{(n)}$ generated by the Gaussian elimination process is strictly diagonally dominant.

This will ensure that at each stage of the Gaussian elimination process the pivot element is *nonzero*.

Since A is strictly diagonally dominant, $a_{11} \neq 0$ and $A^{(2)}$ can be formed.Thus for each i= 2, 3, ... , n,

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{1j}^{(1)} a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad \text{for} \quad 2 \leq j \leq n.$$

First, $a_{i1}^{(2)}$= 0. The triangle inequality implies that

$$\sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}^{(2)}| = \sum_{\substack{j=2 \\ j\neq i}}^{n} \left| a_{ij}^{(1)} - \frac{a_{1j}^{(1)} a_{i1}^{(1)}}{a_{11}^{(1)}} \right| \leq \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}^{(1)}| + \sum_{\substack{j=2 \\ j\neq i}}^{n} \left| \frac{a_{1j}^{(1)} a_{i1}^{(1)}}{a_{11}^{(1)}} \right| .$$

But since A is strictly diagonally dominant,

$$\sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}^{(1)}| < |a_{ii}^{(1)}| - |a_{i1}^{(1)}| \quad \text{and} \quad \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{1j}^{(1)}| < |a_{11}^{(1)}| - |a_{1i}^{(1)}|,$$

so,

$$\sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}^{(2)}| < |a_{ii}^{(1)}| - |a_{i1}^{(1)}| + \frac{|a_{i1}^{(1)}|}{|a_{11}^{(1)}|}(|a_{11}^{(1)}| - |a_{1i}^{(1)}|) = |a_{ii}^{(1)}| - \frac{|a_{i1}^{(1)}||a_{1i}^{(1)}|}{|a_{11}^{(1)}|}.$$

The triangle inequality also implies that

$$|a_{ii}^{(1)}| - \frac{|a_{i1}^{(1)}||a_{1i}^{(1)}|}{|a_{11}^{(1)}|} \leq |a_{ii}^{(1)} - \frac{a_{i1}^{(1)}a_{1i}^{(1)}}{a_{11}^{(1)}}| = |a_{ii}^{(2)}|$$

which gives

$$\sum_{\substack{j=2 \\ j\neq i}}^{n} \left|a_{ij}^{(2)}\right| < \left|a_{ii}^{(2)}\right|.$$

This establishes the strict diagonal dominance for rows 2, ... , n. But the first row of $A^{(2)}$ and A are the same, so $A^{(2)}$ is strictly diagonally dominant.

This process is continued inductively until the upper-triangular and strictly diagonally dominant $A^{(n)}$ is obtained. This implies that all the diagonal elements are *nonzero*, so Gaussian elimination can be performed without row interchanges.

Positive Definite Matrices

Definition 6.22

A matrix A is ***positive definite*** if it is symmetric and if $\mathbf{x}^t A\mathbf{x} > 0$ for every n-dimensional vector $\mathbf{x} \neq 0$.

To be precise, **Definition 6.22** should specify that the 1×1 matrix generated by the operation $\mathbf{x}^t A\mathbf{x}$ has a positive value for its only entry:

$$\mathbf{x}^t A\mathbf{x} = [x_1, x_2, \cdots, x_n] \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= [x_1, x_2, \cdots, x_n] \begin{bmatrix} \sum_{j=1}^n a_{1j}x_j \\ \sum_{j=1}^n a_{2j}x_j \\ \vdots \\ \sum_{j=1}^n a_{nj}x_j \end{bmatrix} = \left[\sum_{i=1}^n \sum_{j=1}^n a_{ij}x_i x_j \right].$$

Example 1 Show that the matrix

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

is positive definite.

Solution Suppose $\mathbf{x}$ is any three-dimensional column vector. Then

$$\begin{aligned}
\mathbf{x}^t A\mathbf{x} &= [x_1, x_2, x_3] \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\
&= [x_1, x_2, x_3] \begin{bmatrix} 2x_1 & - & x_2 & & \\ -x_1 & + & 2x_2 & - & x_3 \\ -x_2 & + & 2x_3 & & \end{bmatrix} \\
&= 2x_1^2 - 2x_1x_2 + 2x_2^2 - 2x_2x_3 + 2x_3^2.
\end{aligned}$$

Rearranging the terms gives

$$\begin{aligned}\mathbf{x}^t A\mathbf{x} &= x_1^2 + (x_1^2 - 2x_1x_2 + x_2^2) + (x_2^2 - 2x_2x_3 + x_3^2) + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2\end{aligned}$$

which implies that

$$x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 > 0$$

unless $x_1 = x_2 = x_3 = 0$.

It should be clear from the example that using the definition to determine if a matrix is positive definite can be difficult.

Fortunately, there are more easily verified criteria, which are presented in Chapter 9, for identifying members of this important class.

Theorem 6.23 If A is an n × n positive definite matrix, then

(i) A has an inverse;

(ii) $a_{ii} > 0$, for each $i = 1, 2, \ldots, n$;

(iii) $\max_{1\le k,j\le n} |a_{kj}| \le \max_{1\le i\le n} |a_{ii}|$;

(iv) $(a_{ij})^2 < a_{ii}a_{jj}$, for each $i \neq j$.

Proof

(i) If $\mathbf{x}$ satisfies $A\mathbf{x} = 0$, then $\mathbf{x}^t A\mathbf{x} = 0$. Since A is positive definite, this implies $\mathbf{x} = \mathbf{0}$. Consequently, $A\mathbf{x} = \mathbf{0}$ has only the zero solution. This is equivalent to A being nonsingular.

Proof

(ii) For a given i, let $\mathbf{x} = (x_j)$ be defined by $x_i = 1$ and $x_j = 0$, if $j \neq i$. Since $\mathbf{x} \neq \mathbf{0}$,

$$0 < \mathbf{x}^t A\mathbf{x} = a_{ii}.$$

(iii) For $k \neq j$, define $\mathbf{x} = (x_i)$ by

$$x_i = \begin{cases} 0, & \text{if } i \neq j \text{ and } i \neq k, \\ 1, & \text{if } i = j, \\ -1, & \text{if } i = k. \end{cases}$$

Since $\mathbf{x} \neq \mathbf{0}$,

$$0 < \mathbf{x}^t A\mathbf{x} = a_{ii} + a_{kk} - a_{jk} - a_{kj}.$$

But $A^t = A$, so $a_{jk} = a_{kj}$, which implies that

$$2a_{kj} < a_{jj} + a_{kk} . \qquad (6.11)$$

Now define $\mathbf{z} = (z_i)$ by

$$z_i = \begin{cases} 0, & \text{if } i \neq j \text{ and } i \neq k, \\ 1, & \text{if } i = j \text{ or } i = k. \end{cases}$$

Then $\mathbf{z}^t A\mathbf{z} > 0$, so

$$-2a_{kj} < a_{jj} + a_{kk} . \qquad (6.12)$$

Equations (6.11) and (6.12) imply that for each $k \neq j$,

$$|a_{kj}| < \frac{a_{kk} + a_{jj}}{2} \leq \max_{1 \leq i \leq n} |a_{ii}|, \quad \text{so} \quad \max_{1 \leq k, j \leq n} |a_{kj}| \leq \max_{1 \leq i \leq n} |a_{ii}|.$$

Proof

(iv) For $i \neq j$, define $\mathbf{x} = (x_k)$ by

$$x_k = \begin{cases} 0, & \text{if } k \neq j \text{ and } k \neq i, \\ \alpha, & \text{if } k = i, \\ 1, & \text{if } k = j, \end{cases}$$

where α represents an arbitrary real number. Because $\mathbf{x} \neq \mathbf{0}$,

$$0 < \mathbf{x}^t A \mathbf{x} = a_{ii}\alpha^2 + 2a_{ij}\alpha + a_{jj}$$

As a quadratic polynomial in α with no real roots, the discriminant of $P(\alpha) = a_{ii}\alpha^2 + 2a_{ij}\alpha + a_{jj}$ must be negative. Thus

$$4a_{ij}^2 - 4a_{ii}a_{jj} < 0 \quad \text{and} \quad a_{ij}^2 < a_{ii}a_{jj}.$$

The following notion will be used to provide a necessary and sufficient condition.

Definition 6.24

A leading principal submatrix of a matrix A is a matrix of the form

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix},$$

for some $1 \le k \le n$.

Theorem 6.25

A symmetric matrix A is positive definite if and only if each of its leading principal submatrices has a positive determinant.

Example 2 In Example 1 we used the definition to show that the symmetric matrix

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

is positive definite.

Solution

$$\det A_1 = [2] = 2 > 0, \det A_2 = \det \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = 4 - 1 = 3 > 0$$

$$\det A_3 = \det \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = 4$$

in agreement with **Theorem 6.25**.

The next result extends part **(i)** of Theorem 6.23 and parallels the strictly diagonally dominant results presented in Theorem 6.21.

Theorem 6.26

The symmetric matrix A is positive definite if and only if Gaussian elimination without row interchanges can be performed on the linear system A**x** = **b** **with all pivot elements positive**. Moreover, in this case, the computations are stable with respect to the growth of round-off errors.

Corollary 6.27

The matrix A is positive definite if and only if A can be factored in the form LDL^t, where L is lower triangular with 1s on its diagonal and D is a diagonal matrix with positive diagonal entries.

Corollary 6.28

The matrix A is positive definite if and only if A can be factored in the form LL^t, where L is lower triangular with nonzero diagonal entries.

The matrix L in this Corollary is not the same as the matrix L in Corollary 6.27. A relationship between them is presented in Exercise 32.

Algorithm 6.5 LDL^t Factorization

Algorithm 6.5 is based on the *LU Factorization Algorithm 6.4* and obtains the LDL^t factorization described in Corollary 6.27.

To factor the positive definite n×n matrix A into the form LDL^t, where L is a lower triangular matrix with *1s* along the diagonal and D is a diagonal matrix with positive entries on the diagonal:

INPUT the dimension n; the entries a_{ij}, $1 \le i, j \le n$ of A;

OUTPUT the entries l_{ij}, $1 \le j \le i$, $1 \le i \le n$ of L and d_i, for $1 \le i \le n$ of D.

Algorithm 6.5

Step 1 For $i = 1, \ldots, n$ do Steps 2–4.

Step 2 For $j = 1, \ldots, i-1$ set $v_j = l_{ij}d_j$.

Step 3 Set $d_i = a_{ii} - \sum_{j=1}^{i-1} l_{ij}v_j$.

Step 4 For $j = i+1, \ldots, n$ set $l_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} l_{jk}v_k\right)/d_i$

Step 5 OUTPUT (l_{ij} for $j = 1, \ldots, i$-1 and $i = 1, \ldots, n$);
OUTPUT (d_i for $j = i, \ldots, n$);
STOP.

Corollary 6.27 has a counterpart when A is symmetric but not necessarily positive definite. This result is widely applied because symmetric matrices are common and easily recognized.

Corollary 6.29 Let A be a **symmetric** $n \times n$ matrix for which Gaussian elimination can be applied without row interchanges. Then A can be factored into LDL^t , where L is lower triangular with 1s on its diagonal and D is the diagonal matrix with $a_{11}^{(1)}$, ... , $a_{nn}^{(n)}$ on its diagonal.

Example 3 Determine the LDL^t factorization of the positive definite matrix

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}.$$

Solution The LDL^t factorization has 1s on the diagonal of the lower triangular matrix L so we need to have

$$A = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} d_1 & d_1 l_{21} & d_1 l_{31} \\ d_1 l_{21} & d_2 + d_1 l_{21}^2 & d_2 l_{32} + d_1 l_{21} l_{31} \\ d_1 l_{31} & d_1 l_{21} l_{31} + d_2 l_{32} & d_1 l_{31}^2 + d_2 l_{32}^2 + d_3 \end{bmatrix}$$

Thus

$$a_{11}: 4 = d_1 \Rightarrow d_1 = 4, \qquad a_{21}: -1 = d_1 l_{21} \Rightarrow l_{21} = -0.25$$

$$a_{31}: 1 = d_1 l_{31} \Rightarrow l_{31} = 0.25, \qquad a_{22}: 4.25 = d_2 + d_1 l_{21}^2 \Rightarrow d_2 = 4$$

$$a_{32}: 2.75 = d_1 l_{21} l_{31} + d_2 l_{32} \Rightarrow l_{32} = 0.75, \quad a_{33}: 3.5 = d_1 l_{31}^2 + d_2 l_{32}^2 + d_3 \Rightarrow d_3 = 1.$$

and we have

$$A = LDL^t = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & 0.75 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -0.25 & 0.25 \\ 0 & 1 & 0.75 \\ 0 & 0 & 1 \end{bmatrix}.$$

Algorithm 6.5 is easily modified to factor the symmetric matrices described in Corollary 6.29. It simply requires adding a check to ensure that the diagonal elements are nonzero. **The Cholesky Algorithm 6.6** produces the LL^t factorization described in Corollary 6.28.

Algorithm 6.6 Cholesky

To factor the positive definite $n \times n$ matrix A into LL^t, where L is lower triangular:

INPUT the dimension n; entries a_{ij}, $1 \le i, j \le n$ of A;

OUTPUT the entries l_{ij}, $1 \le j \le i$, and $1 \le i \le n$ of L.(*The entries of* $U = L^t$ *are* $u_{ij} = l_{ji}$, *for* $i \le j \le n$ *and* $1 \le i \le n$.)

Step 1 Set $l_{11} = \sqrt{a_{11}}$.

Step 2 For $j = 2, \ldots, n$, set $l_{j1} = a_{j1}/l_{11}$.

Step 3 For $i = 2, \ldots, n-1$ do Steps 4 and 5.

Step 4 Set $l_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\right)^{1/2}$.

Algorithm 6.6

Step 5 For $j = i + 1, \ldots, n$. set $l_{ji} = \left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik}\right) / l_{ii}$

Step 6 set $l_{nn} = \left(a_{nn} - \sum_{k=1}^{n-1} l_{nk}^2\right)^{1/2}$.

Step 7 OUTPUT (l_{ij} for $j = 1, \ldots, i$ and $i = 1, \ldots, n$);
STOP.

Example 4 Determine the Cholesky LL^t factorization of the positive definite matrix

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}.$$

Solution

The LL^t factorization does not necessarily has 1s on the diagonal of the lower triangular matrix L so we need to have

$$A = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}$$

Thus

$$a_{11}: \quad 4 = l_{11}^2 \implies l_{11} = 2, \qquad a_{21}: \quad -1 = l_{11}l_{21} \implies l_{21} = -0.5$$

$$a_{31}: \quad 1 = l_{11}l_{31} \implies l_{31} = 0.5, \qquad a_{22}: \quad 4.25 = l_{21}^2 + l_{22}^2 \implies l_{22} = 2$$

$$a_{32}: \quad 2.75 = l_{21}l_{31} + l_{22}l_{32} \implies l_{32} = 1.5, \qquad a_{33}: \quad 3.5 = l_{31}^2 + l_{32}^2 + l_{33}^2 \implies l_{33} = 1,$$

and we have

$$A = LL^t = \begin{bmatrix} 2 & 0 & 0 \\ -0.5 & 2 & 0 \\ 0.5 & 1.5 & 1 \end{bmatrix} \begin{bmatrix} 2 & -0.5 & 0.5 \\ 0 & 2 & 1.5 \\ 0 & 0 & 1 \end{bmatrix}.$$

The LDL^t factorization described in Algorithm 6.5 requires

$$\frac{1}{6}n^3 + n^2 - \frac{7}{6}n \text{ multiplications/divisions}$$

and $$\frac{1}{6}n^3 - \frac{1}{6}n \text{ additions/subtractions.}$$

The LL^t Cholesky factorization of a positive definite matrix requires only

$$\frac{1}{6}n^3 + \frac{1}{2}n^2 - \frac{2}{3}n \text{ multiplications/divisions}$$

and $$\frac{1}{6}n^3 - \frac{1}{6}n \text{ additions/subtractions.}$$

This computational advantage of Cholesky's factorization is misleading, because it requires extracting n square roots.

However, the number of operations required for computing the n square roots is a linear factor of n and will decrease in significance as n increases.

Algorithm 6.5 provides a stable method for factoring a positive definite matrix into the form $A = LDL^t$, but it must be modified to solve the linear system $A\mathbf{x} = \mathbf{b}$.

To do this,we delete the STOP statement from Step 5 in the algorithm and add the following steps to solve the lower triangular system L**y** = **b**:

Algorithm 6.5

Step 6 Set $y_1 = b_1$.

Step 7 For $i = 2, \ldots, n$ set $y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j$.

The linear system D**z** = **y** can then be solved by

Step 8 For $i = 2, \ldots, n$ set $z_i = y_i / d_i$

Finally, the upper-triangular system L^t**x** = **z** is solved with the steps given by

Step 9 Set $x_n = z_n$.

Step 10 For $i = n - 1, \ldots, 1$ set $x_i = z_i - \sum_{j=i+1}^{n} l_{ji} x_j$.

Algorithm 6.5

Step 11 OUTPUT (x_i for $i = 1, \ldots, n$);
STOP.

Table 6.4 shows the additional operations required to solve the linear system.

Table 6.4

Step	Multiplications/Divisions	Additions/Subtractions
6	0	0
7	$n(n-1)/2$	$n(n-1)/2$
8	n	0
9	0	0
10	$n(n-1)/2$	$n(n-1)/2$
Total	n^2	$n^2 - n$

If the Cholesky factorization given in **Algorithm 6.6** is preferred, the additional steps for solving the system A**x** = **b** are as follows.

Algorithm 6.6

First delete the STOP statement from Step 7. Then add

Step 8 Set $y_1 = b_1/l_{11}$

Step 9 For $i = 2, \ldots, n$ set $y_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j\right)/l_{ii}$

Step 10 Set $x_n = b_n/l_{nn}$

Step 11 For $i = n - 1, \ldots, 1$ set $x_i = (y_i - \sum_{j=i+1}^{n} l_{ji} x_j)/l_{ii}$.

Step 12 OUTPUT (x_i for $i = 1, \ldots, n$);
STOP.

Steps 8–12 require $n^2 + n$ multiplications/divisions and $n^2 - n$ additions/subtractions.

Band Matrices

The last class of matrices considered are band matrices. In many applications, the band matrices are also strictly diagonally dominant or positive definite

Definition 6.30

An n × n matrix is called a *band matrix* if integers p and q, with $1 < p, q < n$, exist with the property that $a_{ij} = 0$ whenever $p \leq j - i$ or $q \leq i - j$. The *band width* of a band matrix is defined as $w = p + q - 1$.

The number p describes the number of diagonals above, and including, the main diagonal on which nonzero entries may lie.

The number q describes the number of diagonals below, and including, the main diagonal on which nonzero entries may lie.

For example, the matrix

$$A = \begin{bmatrix} 7 & 2 & 0 \\ 3 & 5 & -1 \\ 0 & -5 & -6 \end{bmatrix}$$

is a band matrix with $p = q = 2$ and bandwidth $2 + 2 - 1 = 3$.

The definition of band matrix forces those matrices to concentrate all their nonzero entries about the diagonal. Two special cases of band matrices that occur frequently have $p = q = 2$ and $p = q = 4$.

Tridiagonal Matrices

Matrices of bandwidth 3 occurring when $p = q = 2$ are called tridiagonal because they have the form

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \ddots & & \vdots \\ 0 & a_{32} & a_{33} & a_{34} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix}.$$

The case of $p = q = 4$ will be used for the solution of boundary-value problems when the approximating functions assume the form of cubic splines.

The factorization algorithms can be simplified considerably in the case of band matrices because a large number of *zeros* appear in these matrices in regular patterns.

It is particularly interesting to observe the form the Crout or Doolittle method assumes in this case.

To illustrate the situation, suppose a tridiagonal matrix A can be factored into the triangular matrices L and U.Then A has at most $(3n-2)$ nonzero entries.

Then there are only $(3n-2)$ conditions to be applied to determine the entries of L and U, provided, of course, that the zero entries of A are also obtained.

Suppose that the matrices L and U also have tridiagonal form, that is,

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & \cdots & 0 \\ l_{21} & l_{22} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & l_{n,n-1} & l_{nn} \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 1 & u_{12} & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}.$$

There are (2n−1) undetermined entries of L and (n−1) undetermined entries of U, which totals (3n −2), the number of possible nonzero entries of A. The 0 entries of A are obtained automatically.

The multiplication involved with $A = LU$ gives, in addition to the 0 entries,

$$a_{11} = l_{11};$$
$$a_{i,i-1} = l_{i,i-1}, \text{ for each } i = 2, 3, \ldots, n; \qquad (6.13)$$

$$a_{ii} = l_{i,i-1}u_{i-1,i} + l_{ii}, \text{ for each } i = 2, 3, \ldots, n; \qquad (6.14)$$

and

$$a_{i,i+1} = l_{ii}u_{i,i+1}, \text{ for each } i = 1, 2, \ldots, n-1. \qquad (6.15)$$

A solution to this system is found by first using Eq. (6.13) to obtain all the nonzero offdiagonal terms in L and then using Eqs. (6.14) and (6.15) to alternately obtain the remainder of the entries in U and L.

Once an entry L or U is computed, the corresponding entry in A is not needed. So the entries in A can be overwritten by the entries in L and U with the result that no new storage is required.

Algorithm 6.7 solves an $n \times n$ system of linear equations whose coefficient matrix is tridiagonal. This algorithm requires only $(5n - 4)$ multiplications/divisions and $(3n - 3)$additions/subtractions.

Consequently, it has considerable computational advantage over the methods that do not consider the tridiagonality of the matrix.

Algorithm 6.7 Crout Factorization for Tridiagonal Linear Systems

To solve the $n \times n$ linear system

$$
\begin{array}{lllllll}
E_1: & a_{11}x_1 + a_{12}x_2 & & & & & = a_{1,n+1}, \\
E_2: & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & & & & & = a_{2,n+1}, \\
\vdots & & \vdots & & & & \vdots \\
E_{n-1}: & & a_{n-1,n-2}x_{n-2} & + a_{n-1,n-1}x_{n-1} & + a_{n-1,n}x_n & & = a_{n-1,n+1}, \\
E_n: & & & a_{n,n-1}x_{n-1} & + a_{nn}x_n & & = a_{n,n+1},
\end{array}
$$

which is assumed to have a unique solution:

Algorithm 6.7 Crout Factorization for Tridiagonal Linear Systems

INPUT the dimension n; the entries of A;

OUTPUT the solution $x_1, \ldots, x_n$.
(*Steps 1–3 set up and solve* $L\mathbf{z} = \boldsymbol{b}$.)

Step 1 Set $l_{11} = a_{11}$;
$u_{12} = a_{12}/l_{11}$;
$z_1 = a_{1,n+1}/l_{11}$.

Step 2 For $i = 2, \ldots, n-1$, set $l_{i,i-1} = a_{i,i-1}$; (*ith row of L.*)
$l_{ii} = a_{ii} - l_{i,i-1}u_{i-1,i}$;
$u_{i,i+1} = a_{i.i+1}/l_{ii}$; (*(i + 1)th column of U.*)
$z_i = (a_{i,n+1} - l_{i,i-1}z_{i-1})/l_{ii}$.

Algorithm 6.7

Step 3 Set $l_{n,n-1} = a_{n,n-1}$; *(nth row of L.)*

$$l_{nn} = a_{nn} - l_{n,n-1}u_{n-1,n};$$

$$z_n = (a_{n,n+1} - l_{n,n-1}z_{n-1})/l_{nn}.$$

(Steps 4 and 5 solve U**x** = **z**.)

Step 4 Set $x_n = z_n$.

Step 5 For $i = n - 1, \ldots, 1$ set $x_i = z_i - u_{i,i+1}x_{i+1}$.

Step 6 OUTPUT $(x_1, \ldots, x_n)$;
STOP.

Example 5 Determine the Crout factorization of the symmetric tridiagonal matrix

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}.$$

and use this factorization to solve the linear system

$$\begin{aligned} 2x_1 - x_2 \qquad\qquad &= 1, \\ -x_1 + 2x_2 - x_3 \qquad &= 0, \\ - x_2 + 2x_3 - x_4 &= 0, \\ - x_3 + 2x_4 &= 1. \end{aligned}$$

Solution The LU factorization of A has the form

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ 0 & l_{32} & l_{33} & 0 \\ 0 & 0 & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & 0 & 0 \\ 0 & 1 & u_{23} & 0 \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} l_{11} & l_{11}u_{12} & 0 & 0 \\ l_{21} & l_{22} + l_{21}u_{12} & l_{22}u_{23} & 0 \\ 0 & l_{32} & l_{33} + l_{32}u_{23} & l_{33}u_{34} \\ 0 & 0 & l_{43} & l_{44} + l_{43}u_{34} \end{bmatrix}.$$

Thus

$$a_{11}: \quad 2 = l_{11} \implies l_{11} = 2, \qquad a_{12}: \quad -1 = l_{11}u_{12} \implies u_{12} = -\tfrac{1}{2},$$

$$a_{21}: \quad -1 = l_{21} \implies l_{21} = -1, \qquad a_{22}: \quad 2 = l_{22} + l_{21}u_{12} \implies l_{22} = -\tfrac{3}{2},$$

$$a_{23}: \quad -1 = l_{22}u_{23} \implies u_{23} = -\tfrac{2}{3}, \qquad a_{32}: \quad -1 = l_{32} \implies l_{32} = -1,$$

$$a_{33}: \quad 2 = l_{33} + l_{32}u_{23} \implies l_{33} = \tfrac{4}{3}, \qquad a_{34}: \quad -1 = l_{33}u_{34} \implies u_{34} = -\tfrac{3}{4},$$

$$a_{43}: \quad -1 = l_{43} \implies l_{43} = -1, \qquad a_{44}: \quad 2 = l_{44} + l_{43}u_{34} \implies l_{44} = \tfrac{5}{4}.$$

This gives the Crout factorization

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & \frac{3}{2} & 0 & 0 \\ 0 & -1 & \frac{4}{3} & 0 \\ 0 & 0 & -1 & \frac{5}{4} \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 1 & -\frac{2}{3} & 0 \\ 0 & 0 & 1 & -\frac{3}{4} \\ 0 & 0 & 0 & 1 \end{bmatrix} = LU.$$

Solving the system

$$L\mathbf{z} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & \frac{3}{2} & 0 & 0 \\ 0 & -1 & \frac{4}{3} & 0 \\ 0 & 0 & -1 & \frac{5}{4} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{gives} \quad \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{4} \\ 1 \end{bmatrix},$$

and then solving

$$U\mathbf{x} = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 1 & -\frac{2}{3} & 0 \\ 0 & 0 & 1 & -\frac{3}{4} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{4} \\ 1 \end{bmatrix} \quad \text{gives} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

The Crout Factorization Algorithm can be applied whenever $l_{ii} \neq 0$ for each $i = 1, 2, \ldots, n$. Two conditions, either of which ensure that this is true, are that the coefficient matrix of the system is *positive definite* or that it is *strictly diagonally dominant.*

An additional condition that ensures this algorithm can be applied is given in the next theorem.

Theorem 6.31

Suppose that $A = [a_{ij}]$ is tridiagonal with $a_{i,i-1}a_{i,i+1} \neq 0$, for each $i = 2, 3, \ldots, n-1$. If $|a_{11}| > |a_{12}|$, $|a_{ii}| \geq |a_{i,i-1}| + |a_{i,i+1}|$, for each $i = 2, 3, \ldots, n-1$, and $|a_{nn}| > |a_{n,n-1}|$, then A is *nonsingular* and the values of l_{ii} described in the Crout Factorization Algorithm are *nonzero* for each $i = 1, 2, \ldots, n$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Chapter 7

Iterative Techniques in Matrix Algebra

Matrices with a high percentage of zero entries are called sparse and are often solved using iterative, rather than direct, techniques.

The methods presented in Chapter 6 used direct techniques to solve a system of $n \times n$ linear equations of the form $A\mathbf{x} = \mathbf{b}$. In this chapter, we present iterative methods to solve a system of this type.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 7-1

Norms of Vectors and Matrices

The objective is to find a way to minimize the difference between the approximations and the exact solution.

To discuss iterative methods for solving linear systems, we first need to determine a way to measure the distance between n-dimensional column vectors.

This will permit us to determine whether a sequence of vectors converges to a solution of the system.

Vector Norms

Let R^n denote the set of all n-dimensional column vectors with real-number components. To define a distance in R^n we use the notion of a *norm*, which is the generalization of the absolute value on R, the set of real numbers.

Definition 7.1 A **vector norm** on R^n is a function, $\|\cdot\|$, from R^n into R with the following properties:

(i) $\|\mathbf{x}\| \geq 0$ for all $\mathbf{x} \in R^n$,

(ii) $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$,

(iii) $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ for all $\alpha \in \mathrm{R}$ and $\mathbf{x} \in R^n$,,

(iv) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in R^n$,

Vectors in R^n are column vectors, and it is convenient to use the *transpose notation* presented in Section 6.3 when a vector is represented in terms of its components.

For example, the vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ will be written $\mathbf{x} = (x_1, x_2, \cdots, x_n)^t$.

We will need only two specific norms on R^n, although a third norm on R^n is presented in Exercise 2.

Definition 7.2

The l_2 and l_∞ norms for the vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)^t$ are defined by

$$\|\mathbf{x}\|_2 = \left\{\sum_{i=1}^{n} x_i^2\right\}^{1/2} \quad \text{and} \quad \|\mathbf{x}\|_\infty = \max_{1 \le i \le n} |x_i|.$$

Note that each of these norms reduces to the absolute value in the case $n = 1$.

The l_2 norm is called the *Euclidean norm* of the vector x because it represents the usual notion of distance from the origin in case **x** is in $R^1 \equiv R$, R^2, or R^3.

For example, the l_2 norm of the vector $\mathbf{x} = (x_1, x_2, x_3)^t$ gives the length of the straight line joining the points (0, 0, 0) and (x_1, x_2, x_3).

Figure 7.1

Figure 7.2

The vectors in $\mathbb{R}^2$ with l_∞ norm less than 1 are inside this figure.

The vectors in the first octant of $\mathbb{R}^3$ with l_∞ norm less than 1 are inside this figure.

Example 1 Determine the l_2 norm and the l_∞ norm of the vector $\mathbf{x} = (-1,1,-2)^t$

Solution The vector $\mathbf{x} = (-1,1,-2)^t$ in R^3 has norms

$$\|\mathbf{x}\|_2 = \sqrt{(-1)^2 + (1)^2 + (-2)^2} = \sqrt{6}$$

and

$$\|\mathbf{x}\|_\infty = \max\{|-1|, |1|, |-2|\} = 2$$

It is easy to show that the properties in **Definition 7.1** hold for the l_∞ norm because they follow from similar results for *absolute values*.

The only property that requires much demonstration is **(iv)**, and in this case if $\mathbf{x} = (x_1, x_2, \cdots, x_n)^t$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^t$,

Definition 7.1 **(iv)** $\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in R^n$,

then

$$\|\mathbf{x}+\mathbf{y}\|_\infty = \max_{1\le i\le n}|x_i + y_i| \le \max_{1\le i\le n}(|x_i| + |y_i|) \le \max_{1\le i\le n}|x_i| + \max_{1\le i\le n}|y_i| = \|\mathbf{x}\|_\infty + \|\mathbf{y}\|_\infty$$

The first three conditions also are easy to show for the l_2 norm.But to show that

$$\|\mathbf{x}+\mathbf{y}\|_2 \le \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2 \quad \text{for each } \mathbf{x}, \mathbf{y} \in R^n$$

Theorem 7.3 (Cauchy-Bunyakovsky-Schwarz Inequality for Sums)

For each $\mathbf{x} = (x_1, x_2, \cdots, x_n)^t$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^t$ in R^n,

$$\mathbf{x}^t\mathbf{y} = \sum_{i=1}^n x_i y_i \le \left\{\sum_{i=1}^n x_i^2\right\}^{1/2}\left\{\sum_{i=1}^n y_i^2\right\}^{1/2} = \|\mathbf{x}\|_2 \cdot \|\mathbf{y}\|_2 \qquad (7.1)$$

With this result we see that for each $\mathbf{x}, \mathbf{y} \in R^n$,

$$\|\mathbf{x}+\mathbf{y}\|_2^2 = \sum_{i=1}^{n}(x_i+y_i)^2 = \sum_{i=1}^{n} {x_i}^2 + 2\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} {y_i}^2$$

$$\leq \|\mathbf{x}\|_2^2 + 2\|\mathbf{x}\|_2\|\mathbf{y}\|_2 + \|\mathbf{y}\|_2^2,$$

which gives norm property **(iv)**:

$$\|\mathbf{x}+\mathbf{y}\|_2 \leq (\|\mathbf{x}\|_2^2 + 2\|\mathbf{x}\|_2\|\mathbf{y}\|_2 + \|\mathbf{y}\|_2^2)^{1/2} = \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2.$$

Distance between Vectors in R^n

The norm of a vector gives a measure for the distance between an arbitrary vector and the zero vector, just as the absolute value of a real number describes its distance from 0.

Similarly, the *distance between two vectors* is defined as the norm of the difference of the vectors just as distance between two real numbers is the absolute value of their difference.

Definition 7.4

If $\mathbf{x} = (x_1, x_2, \cdots, x_n)^t$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^t$ are vectors in R^n, the l_2 and l_∞ distances between x and y are defined by

$$\|\mathbf{x} - \mathbf{y}\|_2 = \{\textstyle\sum_{i=1}^{n}(x_i - y_i)^2\}^{1/2} \quad \text{and} \quad \|\mathbf{x} - \mathbf{y}\|_\infty = \max_{1 \le i \le n} |x_i - y_i| .$$

Example 2 The linear system

$$3.3330\,x_1 + 15920\,x_2 - 10.333\,x_3 = 15913,$$
$$2.2220x_1 + 16.710x_2 + 9.6120x_3 = 28.544,$$
$$1.5611x_1 + 5.1791x_2 + 1.6852x_3 = 8.4254.$$

has the exact solution $\mathbf{x} = (x_1, x_2, x_3)^t = (1,\ 1,\ 1)^t$, and Gaussian elimination performed using five-digit rounding arithmetic and partial pivoting (**Algorithm 6.2**), produces the approximate solution

$$\tilde{\mathbf{x}} = \left(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3\right)^t = (1.2001,\ 0.99991,\ 0.92538)^t$$

Determine the l_2 and l_∞ distances between the exact and approximate solutions.

Solution Measurements of $\mathbf{x} - \tilde{\mathbf{x}}$ are given by

$$\|\mathbf{x}-\tilde{\mathbf{x}}\|_\infty = \max\{|1-1.2001|,|1-0.99991|,|1-0.92538|\}$$
$$= \max\{0.2001, 0.00009, 0.07462\} = 0.2001$$

and

$$\|\mathbf{x}-\tilde{\mathbf{x}}\|_2 = \left[(1-1.2001)^2 + (1-0.99991)^2 + (1-0.92538)^2\right]^{1/2}$$
$$= \left[(0.2001)^2 + (0.00009)^2 + (0.07462)^2\right]^{1/2} = 0.21356.$$

Although the components $\tilde{x}_2$ and $\tilde{x}_3$ are good approximations to x_2 and x_3, the component $\tilde{x}_1$ is a poor approximation to x_1, and $|x_1 - \tilde{x}_1|$ dominates both norms.

The concept of distance in R^n is also used to define a limit of a sequence of vectors in this space.

Definition 7.5

A sequence $\{\mathbf{x}^{(k)}\}_{k=1}^{\infty}$ of vectors in R^n is said to *converge* to $\mathbf{x}$ with respect to the norm $\|\cdot\|$ if, given any $\varepsilon > 0$, there exists an integer $N(\varepsilon)$ such that

$$\|\mathbf{x}^{(k)} - \mathbf{x}\| < \varepsilon, \text{for all } k \geq N(\varepsilon).$$

Theorem 7.6

The sequence of vectors $\{\mathbf{x}^{(k)}\}$ converges to $\mathbf{x}$ in R^n with respect to the l_∞ norm if and only if $\lim_{k\to\infty} x_i^{(k)} = x_i$, for each $i = 1, 2, \ldots, n$.

Proof

Suppose $\{\mathbf{x}^{(k)}\}$ converges to x with respect to the l_∞ norm. Given any $\varepsilon > 0$, there exists an integer N(ε) such that for all $k \geq N(\varepsilon)$,

$$\max_{i=1,2,\ldots,n} \left|x_i^{(k)} - x_i\right| = \left\|\mathbf{x}^{(k)} - \mathbf{x}\right\|_\infty < \varepsilon.$$

This result implies that $\left|x_i^{(k)} - x_i\right| < \varepsilon$, for each i = 1, 2, ... , n, so $\lim_{k\to\infty} x_i^{(k)} = x_i$ for each i.

Conversely, suppose that $\lim_{k\to\infty} x_i^{(k)} = x_i$, for every $i = 1, 2, \ldots, n$. For a given $\varepsilon > 0$, let $N_i(\varepsilon)$ for each i represent an integer with the property that

$$\left|x_i^{(k)} - x_i\right| < \varepsilon$$

whenever $k \geq N_i(\varepsilon)$.

Proof

Define $N(\varepsilon) = \max_{i=1,2,\ldots,n} N_i(\varepsilon)$. If $k \geq N(\varepsilon)$, then

$$\max_{i=1,2,\ldots,n} \left| x_i^{(k)} - x_i \right| = \left\| \mathbf{x}^{(k)} - \mathbf{x} \right\|_\infty < \varepsilon.$$

This implies that $\{\mathbf{x}^{(k)}\}$ converges to $\mathbf{x}$ with respect to the l_∞ norm.

Example 3

Show that

$$\mathbf{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)} x_4^{(k)} \right)^t = \left(1{,}2 + \frac{1}{k}, \frac{3}{k^2}, e^{-k} sink \right)^t.$$

converges to $\mathbf{x} = (1, 2, 0, 0)^{\mathrm{t}}$ with respect to the l_∞ norm.

Theorem 7.6 implies that the sequence $\{\mathbf{x}^{(k)}\}$ converges to $(1, 2, 0, 0)^{\mathrm{t}}$ with respect to the l_∞ norm.

To show directly that the sequence in Example 3 converges to $(1, 2, 0, 0)^{\mathrm{t}}$ with respect to the l_2 norm is quite complicated. It is better to prove the next result and apply it to this special case.

Theorem 7.7

For each $\mathbf{x} \in R^n$, $\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \sqrt{n}\|\mathbf{x}\|_\infty$

Proof

Let x_j be a coordinate of $\mathbf{x}$ such that $\|\mathbf{x}\|_\infty = \max_{1\le i\le n}|x_j|$ .Then

$$\|\mathbf{x}\|_\infty^2 = |x_j|^2 = x_j^2 \le \sum_{i=1}^{n} x_i^2 = \|\mathbf{x}\|_2^2$$

and

$$\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2$$

So

$$\|\mathbf{x}\|_2^2 = \sum_{i=1}^{n} x_i^2 \le \sum_{i=1}^{n} x_j^2 = nx_j^2 = n\|\mathbf{x}\|_\infty^2$$

and $\|\mathbf{x}\|_2 \le \sqrt{n}\|\mathbf{x}\|_\infty$.

Figure 7.3 illustrates this result when n = 2.

Figure 7.3

Example 4 In Example 3, we found that the sequence $\{\mathbf{x}^{(k)}\}$, defined by

$$\mathbf{x}^{(k)} = \left(1, 2 + \frac{1}{k}, \frac{3}{k^2}, e^{-k}\sin k\right)^t,$$

converges to $\mathrm{x} = (1, 2, 0, 0)^{\mathrm{t}}$ with respect to the l_∞ norm. Show that this sequence also converges to $\mathbf{x}$ with respect to the l_2 norm.

Solution

Given any $\varepsilon > 0$, there exists an integer $\mathrm{N}(\varepsilon/2)$ with the property that

$$\left\|\mathbf{x}^{(k)} - \mathbf{x}\right\|_\infty < \frac{\varepsilon}{2}$$

whenever $k \geq N(\varepsilon/2)$.

Theorem 7.7

For each $\mathbf{x} \in R^n$, $\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \sqrt{n}\|\mathbf{x}\|_\infty$

By Theorem 7.7, this implies that

$$\left\|\mathbf{x}^{(k)} - \mathbf{x}\right\|_2 \le \sqrt{4}\left\|\mathbf{x}^{(k)} - \mathbf{x}\right\|_\infty \le 2(\varepsilon/2) = \varepsilon$$

when $k \ge N(\varepsilon/2)$.

So $\{\mathbf{x}^{(k)}\}$ also converges to $\mathbf{x}$ with respect to the l_2 norm.

It can be shown that all norms on R^n are equivalent with respect to convergence; that is, if $\|\cdot\|$ and $\|\cdot\|'$ are any two norms on R^n and $\{\mathbf{x}^{(k)}\}_{k=1}^{\infty}$ has the limit $\mathbf{x}$ with respect to $\|\cdot\|$, then $\{\mathbf{x}^{(k)}\}_{k=1}^{\infty}$ also has the limit $\mathbf{x}$ with respect to $\|\cdot\|'$.

Matrix Norms and Distances

Definition 7.8

A **matrix norm** on the set of all $n \times n$ matrices is a real-valued function, $\|\cdot\|$, defined on this set, satisfying for all $n \times n$ matrices A and B and all real numbers α:

(i) $\|A\| \geq 0$;

(ii) $\|A\| = 0$, if and only if A is O, the matrix with all 0 entries;

(iii) $\|\alpha A\| = |\alpha|\|A\|$;

(iv) $\|A + B\| \leq \|A\| + \|B\|$;

(v) $\|AB\| \leq \|A\|\|B\|$.

The **distance** between $n \times n$ matrices A and B with respect to this matrix norm is $\|A - B\|$.

Although matrix norms can be obtained in various ways, the norms considered most frequently are those that are natural consequences of the vector norms l_2 and l_∞.

These norms are defined using the following theorem.

Theorem 7.9

If $\|\cdot\|$ is a vector norm on R^n, then

$$\|A\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| \qquad (7.2)$$

is a matrix norm.

Matrix norms defined by vector norms are called the *natural*, or induced, *matrix norm* associated with the vector norm.

For any $\mathrm{z} \neq 0$, the vector $\mathbf{x} = \mathbf{z}/\|\mathbf{z}\|$ is a unit vector. Hence

$$\max_{\|\mathbf{x}\|=1} \|A\mathrm{x}\| = \max_{\mathbf{z}\neq\mathbf{0}} \left\| A\left(\frac{\mathbf{z}}{\|\mathbf{z}\|}\right)\right\| = \max_{\mathbf{z}\neq\mathbf{0}} \frac{\|A\mathbf{z}\|}{\|\mathbf{z}\|},$$

and we can alternatively write

$$\|A\| = \max_{\mathbf{z}\neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{z}\|}. \qquad (7.3)$$

The following corollary to Theorem 7.9 follows from this representation of $\|A\|$.

Corollary 7.10

For any vector z $\neq 0$, matrix A, and any natural norm $\|\cdot\|$, we have

$$\|A\mathbf{z}\| \leq \|A\| \cdot \|\mathbf{z}\|.$$

The measure given to a matrix under a natural norm describes how the matrix stretches unit vectors relative to that norm. The *maximum stretch* is the norm of the matrix. The matrix norms we will consider have the forms

$$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|A\mathbf{x}\|_\infty, \qquad \text{the } l_\infty \text{ norm,}$$

and

$$\|A\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2, \qquad \text{the } l_2 \text{ norm.}$$

n = 2 matrix $A = \begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix}$

Figure 7.4

x_2
$\|\mathbf{x}\|_2 = 1$
1
−1
x
1
x_1
−1
x_2
3
$A\mathbf{x}$ for
$\|\mathbf{x}\|_2 = 1$
$A\mathbf{x}$
$\|A\|_2$
1
−2
−1
1
2
x_1
−1
−3

Figure 7.5

The l_∞ norm of a matrix can be easily computed from the entries of the matrix.

Theorem 7.11 If $A = (a_{ij})$ is an $n \times n$ matrix, then

$$\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|.$$

Proof

First we show that $\|A\|_\infty \le \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|.$

Let $\mathbf{x}$ be an n-dimensional vector with $1 = \|\boldsymbol{x}\|_\infty = \max_{1\le i\le n} |x_i|$. Since $A\mathbf{x}$ is also an n-dimensional vector,

$$\|A\mathbf{x}\|_\infty = \max_{1\le i\le n} |(A\mathbf{x})_i| = \max_{1\le i\le n} \left|\sum_{j=1}^{n} a_{ij}x_j\right| \le \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \max_{1\le j\le n} |x_j|.$$

But $\max_{1\le j\le n}|x_j| = \|\mathbf{x}\|_\infty = 1$,so

$$\|Ax\|_\infty \le \max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|.$$

and consequently

$$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1}\|A\mathbf{x}\|_\infty \le \max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}| \qquad (7.4)$$

Now we will show the opposite inequality. Let p be an integer with

$$\sum_{j=1}^{n}|a_{pj}| = \max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|,$$

and $\mathbf{x}$ be the vector with components

$$x_j = \begin{cases} 1, & \text{if } a_{pj} \ge 0, \\ -1, & \text{if } a_{pj} < 0. \end{cases}$$

Then $\|\mathbf{x}\|_\infty = 1$ and $a_{pj}x_j = |a_{pj}|$, for all $j = 1, 2, \ldots, n$, so

$$\|A\mathbf{x}\|_\infty = \max_{1\le i\le n}\left|\sum_{j=1}^{n} a_{ij}x_j\right| \ge \left|\sum_{j=1}^{n} a_{pj}x_j\right| = \left|\sum_{j=1}^{n}\left|a_{pj}\right|\right| = \max_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right|.$$

This result implies that

$$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1}\|A\mathbf{x}\|_\infty \ge \max_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right|$$

Putting this together with Inequality (7.4) gives $\|A\|_\infty = \max\limits_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right|$.

Example 5 Determine $\|A\|_\infty$ for the matrix

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & -1 \\ 5 & -1 & 1 \end{bmatrix}.$$

Solution We have

$$\sum_{j=1}^{3} |a_{1j}| = |1| + |2| + |-1| = 4, \quad \sum_{j=1}^{3} |a_{2j}| = |0| + |3| + |-1| = 4,$$

and

$$\sum_{j=1}^{3} |a_{3j}| = |5| + |-1| + |1| = 7.$$

So Theorem 7.11 implies that $\|A\|_\infty = \max\{4, 4, 7\} = 7$

In the next section, we will discover an alternative method for finding the l_2 norm of a matrix.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 7-2

Eigenvalues and Eigenvectors

Introduction

An $n \times m$ matrix can be considered as a function that uses *matrix multiplication* to take m-dimensional column vectors into n-dimensional column vectors.

So an $n \times m$ matrix is actually a linear function from R^m to R^n. A square matrix A takes the set of n-dimensional vectors into itself, which gives a linear function from R^n to R^n.

In this case, certain nonzero vectors $\mathbf{x}$ might be parallel to $A\mathbf{x}$, which means that a constant λ exists with $A\mathbf{x} = \lambda\mathbf{x}$. For these vectors, we have $(A - \lambda I)\mathbf{x} = \mathbf{0}$.

There is a close connection between these numbers λ and the likelihood that an iterative method will converge. We will consider this connection in this section.

Definition 7.12

If A is a square matrix, the characteristic polynomial of A is defined by

$$p(\lambda) = det(A - \lambda I).$$

It is not difficult to show that p is an *nth-degree polynomial* and, consequently, has at most *n distinct zeros*, some of which might be complex.

If λ is a zero of p, then, since $\det(A - \lambda I) = 0$, **Theorem 6.17** implies that the linear system defined by $(A - \lambda I)\mathbf{x} = \mathbf{0}$ has a solution with $\mathbf{x} \neq \mathbf{0}$.

We wish to study the zeros of p and the nonzero solutions corresponding to these systems.

Definition 7.13

If p is the characteristic polynomial of the matrix A, the zeros of p are *eigenvalues*, or characteristic values, of the matrix A.

If λ is an eigenvalue of A and $\mathbf{x} \neq \mathbf{0}$ satisfies $(A - \lambda I)\mathbf{x} = \mathbf{0}$, then $\mathbf{x}$ is an *eigenvector*, or characteristic vector, of A corresponding to the eigenvalue λ.

To determine the eigenvalues of a matrix, we can use the fact that

* λ is an eigenvalue of A if and only if $\det(A - \lambda I) = 0$.

Once an eigenvalue λ has been found a corresponding eigenvector $\mathbf{x} \neq \mathbf{0}$ is determined by solving the system

* $(A - \lambda I)\mathbf{x} = \mathbf{0}$

Example 1 Show that there are no nonzero vectors x in R^2 with $A\mathbf{x}$ parallel to $\mathbf{x}$ if

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

Solution

The eigenvalues of A are the solutions to the characteristic polynomial

$$0 = \det(A - \lambda I) = \det\begin{bmatrix} -\lambda & 1 \\ -1 & -\lambda \end{bmatrix} = \lambda^2 + 1,$$

so the eigenvalues of A are the complex numbers $\lambda_1 = i$ and $\lambda_2 = -i$. A corresponding eigenvector $\mathbf{x}$ for λ_1 needs to satisfy

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -i & 1 \\ -1 & -i \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -ix_1 + x_2 \\ -x_1 - ix_2 \end{bmatrix},$$

that is, $0 = -ix_1 + x_2$, so $x_2 = ix_1$, and $0 = -x_1 - ix_2$.

Hence if **x** is an eigenvector of A, then exactly one of its components is real and the other is complex.As a consequence, there are no nonzero vectors **x** in R^2 with $A\mathbf{x}$ parallel to **x**.

If **x** is an eigenvector associated with the real eigenvalue λ, then $A\mathbf{x} = \lambda\mathbf{x}$, so the matrix A takes the vector x into a scalar multiple of itself.

(1) If λ is real and $\lambda > 1$, then A has the effect of stretching **x** by a factor of λ, as illustrated in Figure 7.6(a).

(2) If $0 < \lambda < 1$, then A shrinks **x** by a factor of λ (see Figure 7.6(b)).

(3) If $\lambda < 0$, the effects are similar (see Figure 7.6(c) and (d)), although the direction of $A\mathbf{x}$ is reversed.

Figure 7.6

(a) $\lambda > 1$
(b) $1 > \lambda > 0$
(c) $\lambda < -1$
(d) $-1 < \lambda < 0$
$A\mathbf{x}$
$\mathbf{x}$
$A\mathbf{x} = \lambda\mathbf{x}$

Notice also that if **x** is an eigenvector of A associated with the eigenvalue λ and α is any nonzero constant, then α**x** is also an eigenvector since

$$A(\alpha\mathbf{x}) = \alpha(A\mathbf{x}) = \alpha(\lambda\mathbf{x}) = \lambda(\alpha\mathbf{x})$$

An important consequence of this is that for any vector norm $||\cdot||$ we could choose the constant $\alpha = \pm\|x\|^{-1}$, which would result in αx being an eigenvector with norm 1. So

* For every eigenvalue and any vector norm there are eigenvectors with norm 1.

Example 2 Determine the eigenvalues and eigenvectors for the matrix

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & -1 & 4 \end{bmatrix}.$$

Solution The characteristic polynomial of A is

$$p(\lambda) = \det(A - \lambda I) = \det \begin{bmatrix} 2-\lambda & 0 & 0 \\ 1 & 1-\lambda & 2 \\ 1 & -1 & 4-\lambda \end{bmatrix}$$

$$= -\left(\lambda^3 - 7\lambda^2 + 16\lambda - 12\right) = -(\lambda - 3)(\lambda - 2)^2,$$

so there are two eigenvalues of A: $\lambda_1 = 3$ and $\lambda_2 = 2$.

An eigenvector $\mathbf{x_1}$ corresponding to the eigenvalue $\lambda_1 = 3$ is a solution to the vector-matrix equation $(A - 3 \cdot I)\mathbf{x}_1 = \mathbf{0}$, so

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -2 & 2 \\ 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

which implies that $x_1 = 0$ and $x_2 = x_3$.

Any nonzero value of x_3 produces an eigenvector for the eigenvalue $\lambda_1 = 3$.

For example, when $x_3 = 1$ we have the eigenvector $\mathrm{x}_1 = (0, 1, 1)^{\mathrm{t}}$, and any eigenvector of A corresponding to $\lambda = 3$ is a nonzero multiple of x_1.

An eigenvector $\mathbf{x} \neq \mathbf{0}$ of A associated with $\lambda_2 = 2$ is a solution of the system $(A - 2 \cdot I)\mathbf{x} = \mathbf{0}$, so

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 2 \\ 1 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

In this case the eigenvector has only to satisfy the equation

$$x_1 - x_2 + 2x_3 = 0$$

which can be done in various ways.

For example, when $x_1 = 0$ we have $x_2 = 2x_3$, so one choice would be $x_2 = (0, 2, 1)^t$. We could also choose $x_2 = 0$, which requires that $x_1 = -2x_3$.

Hence $x_3 = (-2, 0, 1)^t$ gives a second eigenvector for the eigenvalue $\lambda_2 = 2$ that is not a multiple of x_2.

The eigenvectors of A corresponding to the eigenvalue $\lambda_2 = 2$ generate an entire plane. This plane is described by all vectors of the form

$$\alpha \mathbf{x}_2 + \beta \mathbf{x}_3 = (-2\beta, 2\alpha, \alpha + \beta)^t$$

for arbitrary constants α and β, provided that at least one of the constants is nonzero.

Spectral Radius

Definition 7.14 The *spectral radius* $\rho(A)$ of a matrix A is defined by

$$\rho(A) = max|\lambda|, \quad \text{where } \lambda \text{ is an eigenvalue of } A.$$

(For complex $\lambda = \alpha + \beta i$, we define $|\lambda| = (\alpha^2 + \beta^2)^{1/2}$.)

For the matrix considered in **Example 2**, $\rho(A) = \max\{2,3\} = 3$.

The spectral radius is closely related to the norm of a matrix, as shown in the following theorem.

Theorem 7.15 If A is an $n \times n$ matrix, then

(i) $\|A\|_2 = [\rho(A^t A)]^{1/2}$

(ii) $\rho(A) \leq \|A\|$, for any natural norm $\|\cdot\|$.

Proof

To prove part **(ii)**, suppose λ is an eigenvalue of A with eigenvector **x** and $\|\boldsymbol{x}\| = 1$.

Then $A\mathbf{x} = \lambda\mathbf{x}$ and $|\lambda| = |\lambda| \cdot \|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\| = \|A\|$

Thus

$$\rho(A) = max|\lambda| \leq \|A\|$$

Part (i) of **Theorem 7.15** implies that if A is symmetric, then $\|A\|_2 = \rho(A)$

An interesting and useful result is that for any matrix A and any $\varepsilon > 0$, there exists a natural norm $\|\cdot\|$ with the property that $\rho(A) < \|A\| < \rho(A) + \varepsilon$.

Consequently, $\rho(A)$ is the greatest lower bound for the natural norms on A.

Example 3 Determine the l_2 norm of

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix}.$$

Solution To apply **Theorem 7.15** we need to calculate $\rho(A^t A)$, so we first need the eigenvalues of $A^t A$.

$$A^t A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 2 & -1 \\ 2 & 6 & 4 \\ -1 & 4 & 5 \end{bmatrix}.$$

If

$$0 = \det\left(A^t A - \lambda I\right) = \det\begin{bmatrix} 3-\lambda & 2 & -1 \\ 2 & 6-\lambda & 4 \\ -1 & 4 & 5-\lambda \end{bmatrix}$$

$$= -\lambda^3 + 14\lambda^2 - 42\lambda = -\lambda\left(\lambda^2 - 14\lambda + 42\right),$$

then $\lambda = 0$ or $\lambda = 7 \pm \sqrt{7}$. By Theorem 7.15 we have

$$\|A\|_2 = \sqrt{\rho(A^t A)} = \sqrt{max\{0,\ \ 7-\sqrt{7},\ \ 7+\sqrt{7}\}} = \sqrt{7+\sqrt{7}} \approx 3.106$$

Convergent Matrices

In studying iterative matrix techniques, it is of particular importance to know when powers of a matrix become small (that is, when all the entries approach zero). Matrices of this type are called *convergent*.

Definition 7.16 We call an $n \times n$ matrix A convergent if

$$\lim_{k\to\infty}\left(A^k\right)_{ij} = 0, \text{ for each } i = 1, 2, \dots, n \text{ and } j = 1, 2, \dots, n.$$

Example 4 Show that

$$A = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

is a convergent matrix.

Solution Computing powers of A, we obtain:

$$A^2 = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \quad A^3 = \begin{bmatrix} \frac{1}{8} & 0 \\ \frac{3}{16} & \frac{1}{8} \end{bmatrix}, \quad A^4 = \begin{bmatrix} \frac{1}{16} & 0 \\ \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

and, in general,

$$A^k = \begin{bmatrix} \left(\frac{1}{2}\right)^k & 0 \\ \frac{k}{2^{k+1}} & \left(\frac{1}{2}\right)^k \end{bmatrix}.$$

So A is a convergent matrix because

$$\lim_{k\to\infty} \left(\frac{1}{2}\right)^k = 0 \quad \text{and} \quad \lim_{k\to\infty} \frac{k}{2^{k+1}} = 0.$$

Notice that the convergent matrix A in **Example 4** has $\rho(A) = \frac{1}{2}$, because $\frac{1}{2}$ is the only eigenvalue of A.

This illustrates an important connection that exists between the spectral radius of a matrix and the convergence of the matrix, as detailed in the following result.

Theorem 7.17 The following statements are equivalent.

(i) A is a convergent matrix

(ii) $\lim_{n\to\infty} \|A^n\| = 0$, for some natural norm.

(iii) $\lim_{n\to\infty} \|A^n\| = 0$, for all natural norms.

(iv) $\rho(A) < 1$

(v) $\lim_{n\to\infty} A^n \mathbf{x} = 0$, for every $\mathbf{x}$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 7-3

The Jacobi and Gauss-Siedel Iterative Techniques

Introduction

In this section we describe the *Jacobi and the Gauss-Seidel iterative methods*, classic methods that date to the late eighteenth century.

Iterative techniques are seldom used for solving linear systems of small dimension since the time required for sufficient accuracy exceeds that required for direct techniques such as Gaussian elimination.

For large systems with a *high percentage of 0* entries, however, these techniques are efficient in terms of both computer storage and computation.

Systems of this type arise frequently in circuit analysis and in the numerical solution of boundary-value problems and partial-differential equations.

An iterative technique to solve the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$ starts with an initial approximation $\mathbf{x}^{(0)}$ to the solution $\mathbf{x}$ and generates a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ that converges to $\mathbf{x}$.

Jacobi's Method

The Jacobi iterative method is obtained by solving the *ith* equation in $A\mathbf{x} = \mathbf{b}$ for x_i to obtain (provided $a_{ii} \neq 0$)

$$x_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}}\right) + \frac{b_i}{a_{ii}}, \qquad \text{for } i = 1, 2, \dots, n.$$

For each $k \geq 1$, generate the components $x_i^{(k)}$ of $\mathbf{x}^{(k)}$ from the components of $\mathbf{x}^{(k-1)}$ by

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1 \\ j\neq i}}^{n}\left(-a_{ij}x_j^{(k-1)}\right) + b_i\right], \quad \text{for } i = 1, 2, \dots, n. \qquad (7.5)$$

Example 1 The linear system $A\mathbf{x} = \mathbf{b}$ given by

$$\begin{aligned}
E_1 &: 10x_1 - x_2 + 2x_3 = 6,\\
E_2 &: -x_1 + 11x_2 - x_3 + 3x_4 = 25,\\
E_3 &: 2x_1 - x_2 + 10x_3 - x_4 = -11,\\
E_4 &: 3x_2 - x_3 + 8x_4 = 15.
\end{aligned}$$

has the unique solution $\mathbf{x} = (1,2,-1,1)^t$. Use Jacobi's iterative technique to find approximations $\mathbf{x}^{(k)}$ to $\mathbf{x}$ starting with $\mathbf{x}^{(0)} = (0,0,0,0)^t$ until

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty}{\left\|\mathbf{x}^{(k)}\right\|_\infty} < 10^{-3}$$

Solution

We first solve equation E_i for x_i, for each $i = 1, 2, 3, 4$, to obtain

$$
\begin{aligned}
x_1 &= \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5}, \\
x_2 &= \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11}, \\
x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10}, \\
x_4 &= -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8}.
\end{aligned}
$$

From the initial approximation $\mathbf{x}^{(0)} = (0, 0, 0, 0)^t$ we have $\mathbf{x}^{(1)}$ given by

Solution

$$x_1^{(1)} = \frac{1}{10}x_2^{(0)} - \frac{1}{5}x_3^{(0)} + \frac{3}{5} = 0.6000,$$

$$x_2^{(1)} = \frac{1}{11}x_1^{(0)} + \frac{1}{11}x_3^{(0)} - \frac{3}{11}x_4^{(0)} + \frac{25}{11} = 2.2727,$$

$$x_3^{(1)} = -\frac{1}{5}x_1^{(0)} + \frac{1}{10}x_2^{(0)} + \frac{1}{10}x_4^{(0)} - \frac{11}{10} = -1.1000,$$

$$x_4^{(1)} = -\frac{3}{8}x_2^{(0)} + \frac{1}{8}x_3^{(0)} + \frac{15}{8} = 1.8750.$$

Additional iterates, $\mathbf{x}^k = \left(x_1^k, x_2^k, x_3^k, x_4^k\right)^t$, are generated in a similar manner and are presented in **Table 7.1**.

Table 7.1

k	0	1	2	3	4	5	6	7	8	9	10
$x_1^{(k)}$	0.0000	0.6000	1.0473	0.9326	1.0152	0.9890	1.0032	0.9981	1.0006	0.9997	1.0001
$x_2^{(k)}$	0.0000	2.2727	1.7159	2.053	1.9537	2.0114	1.9922	2.0023	1.9987	2.0004	1.9998
$x_3^{(k)}$	0.0000	−1.1000	−0.8052	−1.0493	−0.9681	−1.0103	−0.9945	−1.0020	−0.9990	−1.0004	−0.9998
$x_4^{(k)}$	0.0000	1.8750	0.8852	1.1309	0.9739	1.0214	0.9944	1.0036	0.9989	1.0006	0.9998

We stopped after ten iterations because

$$\frac{\left\|\mathbf{x}^{(10)}-\mathbf{x}^{(9)}\right\|_\infty}{\left\|\mathbf{x}^{(10)}\right\|_\infty}=\frac{8.0\times 10^{-4}}{1.9998}<10^{-3}$$

In fact, $\left\|\mathbf{x}^{(10)}-\mathbf{x}\right\|_\infty=0.0002.$

In general, iterative techniques for solving linear systems involve a process that converts the system $A\mathbf{x} = \mathbf{b}$ into an equivalent system of the form $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ for some fixed matrix T and vector $\mathbf{c}$.

After the initial vector $\mathbf{x}^{(0)}$ is selected, the sequence of approximate solution vectors is generated by computing

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \boldsymbol{c}$$

for each $k = 1, 2, 3, \ldots$. This should be reminiscent of the fixed-point iteration.

The Jacobi method can be written in the form $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \boldsymbol{c}$ by splitting A into its *diagonal and off-diagonal parts*.

To see this, let D be the diagonal matrix whose diagonal entries are those of A, $-L$ be the strictly lower-triangular part of A, and $-U$ be the strictly upper-triangular part of A.

With this notation,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

is split into

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

$$= D - L - U.$$

The equation $A\mathbf{x} = \mathbf{b}$, or $(D - L - U)\mathbf{x} = \mathbf{b}$, is then transformed into

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b},$$

and, if D^{-1} exists, that is, if $a_{ii} \neq 0$ for each i, then

$$\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b}.$$

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1\\ j\neq i}}^{n}\left(-a_{ij}x_j^{(k-1)}\right)+b_i\right], \quad \text{for } i = 1, 2, \ldots, n. \quad (7.5)$$

This results in the matrix form of the Jacobi iterative technique:

$$\mathbf{x}^{(k)} = D^{-1}(L+U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}. \quad k=1,2,\ldots \qquad (7.6)$$

Introducing the notation $T_j = D^{-1}(L+U)$ and $c_j = D^{-1}\mathbf{b}$ gives the Jacobi technique the form

$$\mathbf{x}^{(k)} = T_j\mathbf{x}^{(k-1)} + c_j. \qquad (7.7)$$

In practice, Eq. (7.5) is used in computation and Eq. (7.7) for theoretical purposes.

Example 2

Express the Jacobi iteration method for the linear system $A\mathbf{x} = \mathbf{b}$ given by

$$
\begin{aligned}
E_1 &: 10x_1 - x_2 + 2x_3 = 6, \\
E_2 &: -x_1 + 11x_2 - x_3 + 3x_4 = 25, \\
E_3 &: 2x_1 - x_2 + 10x_3 - x_4 = -11, \\
E_4 &: 3x_2 - x_3 + 8x_4 = 15.
\end{aligned}
$$

in the form $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$.

Solution

We saw in Example 1 that the Jacobi method for this system has the form

$$
\begin{aligned}
x_1 &= \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5}, \\
x_2 &= \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11}, \\
x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10}, \\
x_4 &= -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8}.
\end{aligned}
$$

Hence we have

$$
T = \begin{bmatrix} 0 & \frac{1}{10} & -\frac{1}{5} & 0 \\ \frac{1}{11} & 0 & \frac{1}{11} & -\frac{3}{11} \\ -\frac{1}{5} & \frac{1}{10} & 0 & \frac{1}{10} \\ 0 & -\frac{3}{8} & \frac{1}{8} & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} \frac{3}{5} \\ \frac{25}{11} \\ -\frac{11}{10} \\ \frac{15}{8} \end{bmatrix}
$$

Jacobi Iterative

Algorithm 7.1 implements the Jacobi iterative technique.

Algorithm 7.1

To solve $A\mathbf{x} = \mathbf{b}$ given an initial approximation $\mathbf{x}^{(0)}$:

INPUT the number of equations and unknowns n; the entries a_{ij}, $1 \leq i, j \leq n$ of the matrix A;the entries b_i ,$1 \leq i \leq n$ of $\mathbf{b}$; the entries XO_i, $1 \leq i \leq n$ of $\mathbf{XO} = \mathbf{x}^{(0)}$; tolerance TOL; maximum number of iterations N.

OUTPUT the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

Algorithm 7.1

Step 1 Set $k = 1$.

Step 2 While ($k \leq N$) do Steps 3–6.

Step 3 For $i = 1, \ldots, n$

$$\text{set} \quad x_i = \frac{1}{a_{ii}}\left[-\sum_{\substack{j=1 \\ j\neq i}}^{n}\left(a_{ij}XO_j\right) + b_i\right]$$

Step 4 If $\|\mathbf{x} - \mathbf{XO}\| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$;

(The procedure was successful.)

STOP.

Algorithm 7.1

Step 5 Set $k = k + 1$.

Step 6 For $i = 1, \ldots, n$ set $XO_i = x_i$.

Step 7 OUTPUT *('Maximum number of iterations exceeded')*;
(The procedure was successful.)
STOP.

Step 3 of the algorithm requires that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$. If one of the a_{ii} entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no $a_{ii} = 0$.

To speed convergence, the equations should be arranged so that a_{ii} is as large as possible. This subject is discussed in more detail later in this chapter.

Another possible *stopping criterion* in Step 4 is to iterate until

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|}{\left\|\mathbf{x}^{(k)}\right\|}$$

is smaller than some prescribed tolerance.
For this purpose, any convenient norm can be used, the usual being the l_∞ norm.

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1\\ j\neq i}}^{n}\left(-a_{ij}x_j^{(k-1)}\right)+b_i\right], \quad \text{for } i = 1, 2, \dots, n. \quad (7.5)$$

The Gauss-Seidel Method

A possible improvement in Algorithm 7.1 can be seen by reconsidering Eq. (7.5)

The components of $\mathbf{x}^{(k-1)}$ are used to compute all the components $x_i^{(k)}$ of $\mathbf{x}^{(k)}$. But, for $i > 1$, the components $x_1^{(k)}, \dots, x_{i-1}^{(k)}$ of $\mathbf{x}^{(k)}$ have already been computed and are expected to be better approximations to the actual solutions $x_1, \dots, x_{i-1}$ than are $x_1^{(k-1)}, \dots, x_{i-1}^{(k-1)}$.

It seems reasonable, then, to compute $x_i^{(k)}$ using these most recently calculated values.

The Gauss-Seidel Method

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1\\ j\neq i}}^{n}\left(-a_{ij}x_j^{(k-1)}\right)+b_i\right], \quad \text{for } i = 1, 2, \ldots, n. \quad (7.5)$$

That is, to use

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}\left(a_{ij}x_j^{(k)}\right)-\sum_{j=i+1}^{n}\left(a_{ij}x_j^{(k-1)}\right)+b_i\right]. \quad (7.8)$$

for each $i = 1, 2, \ldots, n$, instead of Eq. (7.5).

This modification is called the *Gauss-Seidel iterative technique* and is illustrated in the following example.

Example 3

Use the Gauss-Seidel iterative technique to find approximate solutions to

$$\begin{aligned} 10x_1 - x_2 + 2x_3 \qquad &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 252, \\ x_1 - x_2 + 10x_3 - x_4 &= -113, \\ x_2 - x_3 + 8x_4 &= 15. \end{aligned}$$

starting with $\mathbf{x} = (0,0,0,0)^t$ and iterating until

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty}{\left\|\mathbf{x}^{(k)}\right\|_\infty} < 10^{-3}$$

Solution

The solution $\mathbf{x} = (1,2,-1,1)^t$ was approximated by Jacobi's method in Example 1. For the Gauss-Seidel method we write the system, for each $k = 1, 2, \ldots$ as

$$
\begin{aligned}
x_1^{(k)} &= \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5},\\
x_2^{(k)} &= \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11},\\
x_3^{(k)} &= -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10},\\
x_4^{(k)} &= -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8}.
\end{aligned}
$$

When $\mathbf{x}^{(0)} = (0,0,0,0)^t$, we have $\mathbf{x}^{(1)} = (0.6000, 2.3272, -0.9873, 0.8789)^t$

Subsequent iterations give the values in **Table 7.2**.

Table 7.2

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	−0.9873	−1.014	−1.0025	−1.0003	−1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

Because

$$\frac{\|\mathbf{x}^{(5)} - \mathbf{x}^{(4)}\|_\infty}{\|\mathbf{x}^{(5)}\|_\infty} = \frac{0.0008}{2.000} = 4 \times 10^{-4}$$

$\mathbf{x}^{(5)}$ is accepted as a reasonable approximation to the solution. Note that Jacobi's method in Example 1 required *twice* as many iterations for the same accuracy.

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}\left(a_{ij}x_j^{(k)}\right) - \sum_{j=i+1}^{n}\left(a_{ij}x_j^{(k-1)}\right) + b_i\right]. \qquad (7.8)$$

To write the Gauss-Seidel method in matrix form, multiply both sides of Eq. (7.8) by a_{ii} and collect all *kth* iterate terms, to give

$$a_{i1}x_1^{(k)} + a_{i2}x_2^{(k)} + \cdots + a_{ii}x_i^{(k)} = -a_{i,i+1}x_{i+1}^{(k-1)} - \cdots - a_{in}x_n^{(k-1)} + b_i$$

for each $i = 1, 2, \ldots, n$. Writing all n equations gives

$$\begin{aligned}
a_{11}x_1^{(k)} &= -a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots - a_{1n}x_n^{(k-1)} + b_1, \\
a_{21}x_1^{(k)} + a_{22}x_2^{(k)} &= -a_{23}x_3^{(k-1)} - \cdots - a_{2n}x_n^{(k-1)} + b_2, \\
&\vdots \\
a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn}x_n^{(k)} &= b_n.
\end{aligned}$$

with the definitions of D, L, and U given previously, we have the Gauss-Seidel method represented by

$$(D-L)\mathbf{x}^{(k)} = U\mathbf{x}^{(k-1)} + \mathbf{b}$$

and

$$\mathbf{x}^{(k)} = (D-L)^{-1}U\mathbf{x}^{(k-1)} + (D-L)^{-1}\mathbf{b}. \quad \text{for each } k = 1,2,\ldots \qquad (7.9)$$

Letting $T_g = (D-L)^{-1}U$ and $\mathbf{c}_j = (D-L)^{-1}\mathbf{b}$, gives the Gauss-Seidel technique the form

$$\mathbf{x}^{(k)} = T_g\mathbf{x}^{(k-1)} + \mathbf{c}_j. \qquad (7.10)$$

For the lower-triangular matrix $D - L$ to be nonsingular, it is necessary and sufficient that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$.

Gauss-Seidel Iterative

Algorithm 7.2 implements the Gauss-Seidel method.

Algorithm 7.2

To solve $A\mathbf{x} = \mathbf{b}$ given an initial approximation $\mathbf{x}^{(0)}$:

INPUT the number of equations and unknowns n; the entries a_{ij}, $1 \leq i, j \leq n$ of the matrix A;the entries b_i ,$1 \leq i \leq n$ of $\mathbf{b}$; the entries XO_i, $1 \leq i \leq n$ of $\mathbf{XO} = \mathbf{x}^{(0)}$; tolerance TOL; maximum number of iterations N.

OUTPUT the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

Algorithm 7.2

Step 1 Set $k = 1$.

Step 2 While ($k \leq N$) do Steps 3–6.

Step 3 For $i = 1, \ldots, n$

$$\text{set} \quad x_i = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}\, x_j - \sum_{j=i+1}^{n} a_{ij}\, XO_j + b_i\right]$$

Step 4 If $\|\mathbf{x} - \mathbf{XO}\| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$;

(The procedure was successful.)

STOP.

Algorithm 7.2

Step 5 Set $k = k + 1$.

Step 6 For $i = 1, \ldots, n$ set $XO_i = x_i$.

Step 7 OUTPUT *('Maximum number of iterations exceeded')*;
(The procedure was successful.)
STOP.

The results of Examples 1 and 2 appear to imply that the Gauss-Seidel method is superior to the Jacobi method

This is almost always true, but there are linear systems for which the Jacobi method converges and the Gauss-Seidel method does not.

General Iteration Methods

To study the convergence of general iteration techniques, we need to analyze the formula

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}. \qquad \text{for each } k = 1, 2, \ldots,$$

where $\mathbf{x}^{(0)}$ is arbitrary. The next lemma and Theorem 7.17 provide the key for this study.

General Iteration Methods

Lemma 7.18 If the spectral radius satisfies $\rho(T) < 1$, then $(I-T)^{-1}$ exists, and

$$(I-T)^{-1} = I + T + T^2 + \ldots = \sum_{j=0}^{\infty} T^j$$

Theorem 7.19 For any $\mathbf{x}^{(0)} \in R^n$, the sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ defined by

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}. \qquad \text{for each } k \geq 1,$$

converges to the unique solution of $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ if and only if $\rho(T) < 1$.

Corollary 7.20

If $\|T\| < 1$ for any natural matrix norm and $\mathbf{c}$ is a given vector, then the sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ defined by $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$ converges, for any $\mathbf{x}^{(0)} \in$ Rn, to a vector $\mathbf{x} \in$ Rn with $\mathbf{x} = T\mathbf{x} + \mathbf{c}$, and the following error bounds hold:

(i) $\left\|\mathbf{x} - \mathbf{x}^{(k)}\right\| \leq \|T\|^k \left\|\mathbf{x}^{(0)} - \mathbf{x}\right\|$;

(ii) $\left\|\mathbf{x} - \mathbf{x}^{(k)}\right\| \leq \frac{\|T\|^k}{1-\|T\|} \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\|$.

We have seen that the *Jacobi* and *Gauss-Seidel* iterative techniques can be written

$$\mathbf{x}^{(k)} = T_j\mathbf{x}^{(k-1)} + \mathbf{c}_j \qquad \text{and} \qquad \mathbf{x}^{(k)} = T_g\mathbf{x}^{(k-1)} + \mathbf{c}_g$$

using the matrices

$$T_j = D^{-1}(L+U) \qquad and \qquad T_g = (D-L)^{-1}U$$

If $\rho(T_j)$ or $\rho(T_g)$ is less than 1, then the corresponding sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ will converge to the solution x of $A\mathbf{x} = \mathbf{b}$. For example, the Jacobi scheme has

$$\mathbf{x}^{(k)} = D^{-1}(L+U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}$$

and, if $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ converges to $\mathbf{x}$, then

$$\mathbf{x} = D^{-1}(L+U)\mathbf{x} + D^{-1}\mathbf{b}$$

This implies that

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b} \qquad \text{and} \qquad (D - L - U)\mathbf{x} = \mathbf{b}$$

Since $D - L - U = A$, the solution $\mathbf{x}$ satisfies $A\mathbf{x} = \mathbf{b}$.

We can now give easily verified sufficiency conditions for convergence of the Jacobi and Gauss-Seidel methods.

Theorem 7.21

If A is strictly diagonally dominant, then for any choice of $\mathbf{x}^{(0)}$, both the Jacobi and Gauss-Seidel methods give sequences $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ that converge to the unique solution of $A\mathbf{x} = \mathbf{b}$.

Theorem 7.15 If A is an $n \times n$ matrix, then
(i) $\|A\|_2 = [\rho(A^t A)]^{1/2}$ **(ii)** $\rho(A) \leq \|A\|$, for any natural norm $\|\cdot\|$.

The relationship of the rapidity of convergence to the spectral radius of the iteration matrix T can be seen from **Corollary 7.20.**
The inequalities hold for any natural matrix norm, so it follows from the statement after **Theorem 7.15** that

$$\|\mathbf{x}^{(k)} - \mathbf{x}\| \approx \rho(T)^k \|\mathbf{x}^{(0)} - \mathbf{x}\| \qquad (7.12)$$

Thus we would like to select the iterative technique with minimal $\rho(T) < 1$ for a particular system $A\mathbf{x} = \mathbf{b}$.

Theorem 7.22

If $a_{ij} \leq 0$, for each i $\neq$ j and $a_{ii} > 0$, for each $i = 1, 2, \ldots, n$, then one and only one of the following statements holds:

(i) $0 \leq \rho(T_g) < \rho(T_j) < 1$; **(ii)** $1 < \rho(T_j) < \rho(T_g)$;

(iii) $\rho(T_j) = \rho(T_g) = 0$; **(iv)** $\rho(T_j) = \rho(T_g) = 1$.

For the special case described in **Theorem 7.22**, we see from part (i) that when one method gives convergence, then both give convergence, and the Gauss-Seidel method converges faster than the Jacobi method. Part (ii) indicates that when one method diverges then both diverge, and the divergence is more pronounced for the Gauss-Seidel method.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 7-4

Relaxation Techniques for Solving Linear Systems

Introduction

We saw in **Section 7.3** that the rate of convergence of an iterative technique depends on the spectral radius of the matrix associated with the method.

One way to select a procedure to accelerate convergence is to choose a method whose associated matrix has minimal spectral radius.

Before describing a procedure for selecting such a method, we need to introduce a new means of measuring the amount by which an approximation to the solution to a linear system differs from the true solution to the system.

The method makes use of the vector described in the following definition.

Definition 7.23

Suppose $\tilde{\mathbf{x}} \in R^n$ is an approximation to the solution of the linear system defined by $A\mathbf{x} = \mathbf{b}$. The residual vector for $\tilde{\mathbf{x}}$ with respect to this system is $\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{x}}$.

In procedures such as the Jacobi or Gauss-Seidel methods, a residual vector is associated with each calculation of an approximate component to the solution vector.

The true objective is to generate a sequence of approximations that will cause the residual vectors to *converge rapidly to zero*.Suppose we let

$$\mathbf{r}_i^{(k)} = \left(\mathrm{r}_{1i}^{(k)}, \mathrm{r}_{2i}^{(k)}, \cdots, \mathrm{r}_{ni}^{(k)}\right)^t$$

denote the residual vector for the *Gauss-Seidel method* corresponding to the approximate solution vector $\mathbf{x}_i^{(k)}$ defined by

$$\mathbf{x}_i^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \cdots, x_{i-1}^{(k)}, x_i^{(k-1)}, \cdots, x_n^{(k-1)}\right)^t.$$

The *m*th component of $\mathbf{r}_i^{(k)}$ is

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj} x_j^{(k)} - \sum_{j=i}^{n} a_{mj} x_j^{(k-1)}, \qquad (7.13)$$

or, equivalently,

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj} x_j^{(k)} - \sum_{j=i+1}^{n} a_{mj} x_j^{(k-1)} - a_{mi} x_i^{(k-1)},$$

for each $m = 1, 2, \ldots, n$.

In particular, the *i*th component of $\mathbf{r}_i^{(k)}$ is

$$\mathrm{r}_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} - a_{ii} x_i^{(k-1)},$$

so

$$a_{ii} x_i^{(k-1)} + \mathrm{r}_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)}. \qquad (7.14)$$

Recall, however, that in the Gauss-Seidel method, $x_i^{(k)}$ is chosen to be

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right], \qquad (7.15)$$

so Eq. (7.14) can be rewritten as

$$a_{ii} x_i^{(k-1)} + \mathrm{r}_{ii}^{(k)} = a_{ii} x_i^{(k)}.$$

Consequently, the Gauss-Seidel method can be characterized as choosing $x_i^{(k)}$ to satisfy

$$x_i^{(k)} = x_i^{(k-1)} + \frac{\mathrm{r}_{ii}^{(k)}}{a_{ii}}. \qquad (7.16)$$

We can derive another connection between the residual vectors and the Gauss-Seidel technique.

Consider the residual vector $\mathbf{r}_{i+1}^{(k)}$, associated with the vector $\mathbf{x}_{i+1}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \cdots, x_i^{(k)}, x_{i+1}^{(k-1)}, \cdots, x_n^{(k-1)}\right)^t$. By Eq. (7.13) the *i*th component of $\mathbf{r}_{i+1}^{(k)}$ is

$$\begin{aligned}\mathrm{r}_{i,i+1}^{(k)} &= b_i - \sum_{j=1}^{i} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} \\ &= b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k-1)}.\end{aligned}$$

By the manner in which $x_i^{(k)}$ is defined in Eq. (7.15) we see that $\mathrm{r}_{i,i+1}^{(k)} = 0$. In a sense, then, the Gauss-Seidel technique is characterized by choosing each $x_{i+1}^{(k)}$ in such a way that the ith component of $\mathbf{r}_{i+1}^{(k)}$ is zero.

Choosing $x_{i+1}^{(k)}$ so that one coordinate of the residual vector is zero, however, is not necessarily the most efficient way to reduce the norm of the vector $\mathbf{r}_{i+1}^{(k)}$. If we modify the Gauss-Seidel procedure, as given by Eq. (7.16), to

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{\mathrm{r}_{ii}^{(k)}}{a_{ii}}. \qquad (7.17)$$

then for certain choices of positive ω we can reduce the norm of the residual vector and obtain significantly faster convergence.

Methods involving Eq. (7.17) are called ***relaxation methods***. For choices of ω with $0 < \omega < 1$, the procedures are called ***under-relaxation methods***.

We will be interested in choices of ω with $1 < \omega$, and these are called ***over-relaxation methods***.

They are used to accelerate the convergence for systems that are convergent by the Gauss-Seidel technique.The methods are abbreviated ***SOR***, for ***Successive Over-Relaxation***, and are particularly useful for solving the linear systems that occur in the numerical solution of certain partial-differential equations.

Before illustrating the advantages of the *SOR method*, we note that by using Eq. (7.14), we can reformulate Eq. (7.17) for calculation purposes as

$$a_{ii}x_i^{(k-1)} + \mathrm{r}_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}. \qquad (7.14)$$

$$x_i^{(k)} = (1-\omega)x_i^{(k-1)} + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right].$$

To determine the matrix form of the *SOR method*, we rewrite this as

$$a_{ii}x_i^{(k)} + \omega\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} = (1-\omega)a_{ii}x_i^{(k)} - \omega\sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} + \omega b_i\ ,$$

so that in vector form, we have

$$(D-\omega L)\mathbf{x}^{(k)} = [(1-\omega)D + \omega U]\mathbf{x}^{(k-1)} + \omega\mathbf{b}.$$

That is,

$$\mathbf{x}^{(k)} = (D-\omega L)^{-1}[(1-\omega)D + \omega U]\mathbf{x}^{(k-1)} + \omega(D-\omega L)^{-1}\mathbf{b}. \qquad (7.18)$$

Letting $T_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega U]$ and $\mathbf{c}_\omega = \omega(D - \omega L)^{-1}\mathbf{b}$, gives the SOR technique the form

$$\mathbf{x}^{(k)} = T_\omega \mathbf{x}^{(k-1)} + \mathbf{c}_\omega \ . \qquad (7.19)$$

Example 1 The linear system $A\mathbf{x} = \mathbf{b}$ given by

$$\begin{aligned} 4x_1 + 3x_2 \qquad\quad &= 24, \\ 3x_1 + 4x_2 - \ \ x_3 &= 30, \\ -x_2 + 4x_3 &= -24, \end{aligned}$$

has the solution $(3,4,-5)^t$. Compare the iterations from the *Gauss-Seidel method* and the *SOR method* with $\omega = 1.25$ using $\mathbf{x}^{(0)} = (1,1,1)^t$ for both methods.

Solution

For each $k = 1, 2, \ldots$, the equations for the Gauss-Seidel method are

$$x_1^{(k)} = -0.75x_2^{(k-1)} + 6,$$
$$x_2^{(k)} = -0.75x_1^{(k)} + 0.25x_3^{(k-1)} + 7.5,$$
$$x_3^{(k)} = 0.25x_2^{(k)} - 6,$$

and the equations for the SOR method with $\omega = 1.25$ are

$$x_1^{(k)} = -0.25x_1^{(k-1)} - 0.9375x_2^{(k-1)} + 7.5,$$
$$x_2^{(k)} = -0.9375x_1^{(k)} - 0.25x_2^{(k-1)} + 0.3125x_3^{(k-1)} + 9.375,$$
$$x_3^{(k)} = 0.3125x_2^{(k)} - 0.25x_3^{(k-1)} - 7.5.$$

Solution

The first seven iterates for each method are listed in **Tables 7.3** and **7.4**. For the iterates to be accurate to seven decimal places, the Gauss-Seidel method requires 34 iterations, as opposed to 14 iterations for the SOR method with $\omega = 1.25$.

Table 7.3

k	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1	5.250000	3.1406250	3.0878906	3.0549316	3.0343323	3.0214577	3.0134110
$x_2^{(k)}$	1	3.812500	3.8828125	3.9267578	3.9542236	3.9713898	3.9821186	3.9888241
$x_3^{(k)}$	1	−5.046875	−5.0292969	−5.0183105	−5.0114441	−5.0071526	−5.0044703	−5.0027940

Table 7.4

k	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1	6.312500	2.6223145	3.1333027	2.9570512	3.0037211	2.9963276	3.0000498
$x_2^{(k)}$	1	3.5195313	3.9585266	4.0102646	4.0074838	4.0029250	4.0009262	4.0002586
$x_3^{(k)}$	1	−6.6501465	−4.6004238	−5.0966863	−4.9734897	−5.0057135	−4.9982822	−5.0003486

An obvious question to ask is how the appropriate value of ω is chosen when the SOR method is used. Although no complete answer to this question is known for the general $n \times n$ linear system, the following results can be used in certain important situations.

Theorem 7.24 (Kahan)

If $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$, then $\rho(T_\omega) \geq |\omega - 1|$. This implies that the SOR method can converge only if $0 < \omega < 2$.

Theorem 7.25 (Ostrowski-Reich)

If A is a positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of initial approximate vector $\mathbf{x}^{(0)}$.

Theorem 7.26

If A is positive definite and tridiagonal, then $\rho(T_g) = [\rho(T_j)]^2 < 1$, and the optimal choice of ω for the SOR method is

$$\omega = \frac{2}{1+\sqrt{1-[\rho(T_j)]^2}}.$$

With this choice of ω, we have $\rho(T_\omega) = \omega - 1$.

Example 2

Find the optimal choice of ω for the SOR method for the matrix

$$A = \begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}.$$

Solution

This matrix is clearly tridiagonal, we can also know that it is positive definite.

$$T_j = D^{-1}(L+U) = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} 0 & -3 & 0 \\ -3 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -0.75 & 0 \\ -0.75 & 0 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix},$$

Solution

we have

$$T_j - \lambda I = \begin{bmatrix} -\lambda & -0.75 & 0 \\ -0.75 & -\lambda & 0.25 \\ 0 & 0.25 & -\lambda \end{bmatrix},$$

so

$$det\left(T_j - \lambda I\right) = -\lambda(\lambda^2 - 0.625)$$

Thus

$$\rho(T_j) = \sqrt{0.625}$$

and

$$\omega = \frac{2}{1+\sqrt{1-\left[\rho(T_j)\right]^2}} = \frac{2}{1+\sqrt{1-0.625}} \approx 1.24\ .$$

This explains the rapid convergence obtained in Example 1 when using $\omega = 1.25$.

SOR

Algorithm 7.3

To solve $A\mathbf{x} = \mathbf{b}$ given the parameter ω and an initial approximation $\mathbf{x}^{(0)}$:

INPUT the number of equations and unknowns n; the entries a_{ij}, $1 \leq i, j \leq n$ of the matrix A;the entries b_i ,$1 \leq i \leq n$ of $\mathbf{b}$; the entries XO_i, $1 \leq i \leq n$ of $\mathbf{XO} = \mathbf{x}^{(0)}$; the parameter ω; tolerance TOL; maximum number of iterations N.

OUTPUT the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

Algorithm 7.3

Step 1 Set $k = 1$.

Step 2 While ($k \leq N$) do Steps 3–6.

Step 3 For $i = 1, \ldots, n$

$$\text{set} \quad x_i = (1-\omega)XO_i + \frac{1}{a_{ii}}\left[\omega\left(-\sum_{j=1}^{i-1} a_{ij}\, x_j - \sum_{j=i+1}^{n} a_{ij}\, XO_j + b_i\right)\right]$$

Step 4 If $\|\mathbf{x} - \mathbf{XO}\| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$;

(The procedure was successful.)

STOP.

Algorithm 7.3

Step 5 Set $k = k + 1$.

Step 6 For $i = 1, \ldots, n$ set $XO_i = x_i$.

Step 7 OUTPUT *('Maximum number of iterations exceeded')*;
(The procedure was successful.)
STOP.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 7-5

Error Bounds and Iterative Refinement

Introduction

It seems intuitively reasonable that if $\tilde{\mathbf{x}}$ is an approximation to the solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ and the residual vector $\boldsymbol{r} = \boldsymbol{b} - A\tilde{\boldsymbol{x}}$ has the property that $\|\mathbf{r}\|$ is small, then $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ would be small as well.

This is often the case, but certain systems, which occur frequently in practice, fail to have this property.

Example 1 The linear system $A\mathbf{x} = \mathbf{b}$ given by

$$\begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3.0001 \end{bmatrix}.$$

has the unique solution $\mathbf{x} = (1,1)^t$. Determine the residual vector for the poor approximation $\tilde{\mathbf{x}} = (3, -0.0001)^t$.

Solution We have

$$\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{x}} = \begin{bmatrix} 3 \\ 3.0001 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -0.0001 \end{bmatrix} = \begin{bmatrix} 0.0002 \\ 0 \end{bmatrix},$$

so $\|\mathbf{r}\|_\infty = 0.0002$. Although the norm of the residual vector is small, the approximation $\tilde{\mathbf{x}} = (3, -0.0001)^t$ is obviously quite poor; in fact, $\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty = 2$.

The difficulty in **Example 1** is explained quite simply by noting that the solution to the system represents the intersection of the lines

$$l_1: \quad x_1 + 2x_2 = 3 \qquad \text{and} \qquad l_2: \quad 1.0001x_1 + 2x_2 = 3.0001.$$

The point (3, −0.0001) lies on l_2, and the lines are nearly parallel.

This implies that (3, −0.0001) also lies close to l_1, even though it differs significantly from the solution of the system, given by the intersection point (1, 1).(See **Figure 7.7**.)

Table 7.7

Example 1 was clearly constructed to show the difficulties that can—and, in fact, do—arise. Had the lines not been nearly coincident, we would expect a *small residual vector* to imply an accurate approximation.

Theorem 7.27

Suppose that $\tilde{\mathbf{x}}$ is an approximation to the solution of $A\mathbf{x} = \mathbf{b}$, A is a nonsingular matrix, and $\mathbf{r}$ is the residual vector for $\tilde{\mathbf{x}}$. Then for any natural norm,

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| \leq \|\mathbf{r}\| \cdot \|A^{-1}\|$$

and if $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{b} \neq \mathbf{0}$,

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} . \quad (7.20)$$

Proof Since $\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{x}} = A\mathbf{x} - A\tilde{\mathbf{x}}$ and A is nonsingular, we have $\mathbf{x} - \tilde{\mathbf{x}} = A^{-1}\mathbf{r}$.

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| = \|A^{-1}\mathbf{r}\| \leq \|A^{-1}\| \cdot \|\mathbf{r}\|$$

Moreover, since $\mathbf{b} = A\mathbf{x}$, we have $\|\mathbf{b}\| \leq \|A\| \cdot \|\mathbf{x}\|$. So $1/\|\mathbf{x}\| \leq \|A\|/\|\mathbf{b}\|$ and

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \frac{\|A\| \cdot \|A^{-1}\|}{\|\mathbf{b}\|} \|\mathbf{r}\|.$$

Condition Numbers

The inequalities in **Theorem 7.27** imply that $\|A^{-1}\|$ and $\|A\| \cdot \|A^{-1}\|$ provide an indication of the connection between the residual vector and the accuracy of the approximation.

In general, the relative error $\|\mathbf{x} - \tilde{\mathbf{x}}\|/\|\mathbf{x}\|$ is of most interest, and, by Inequality (7.20), this error is bounded by the product of $\|A\| \cdot \|A^{-1}\|$ with the relative residual for this approximation, $\|\mathbf{r}\|/\|\mathbf{b}\|$.

Any convenient norm can be used for this approximation; the only requirement is that it be used consistently throughout.

Definition 7.28

The *condition number* of the nonsingular matrix A relative to a norm $\|\cdot\|$ is

$$K(A) = \|A\| \cdot \|A^{-1}\|.$$

With this notation, the inequalities in Theorem 7.27 become

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| \leq K(A) \frac{\|\mathbf{r}\|}{\|A\|}$$

and

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq K(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

For any nonsingular matrix A and natural norm $\|\cdot\|$,

$$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = K(A).$$

A matrix A is *well-conditioned* if $K(A)$ is close to 1, and is *ill-conditioned* when $K(A)$ is significantly greater than 1.

Conditioning in this context refers to the relative security that a small residual vector implies a correspondingly accurate approximate solution.

Example 2 Determine the condition number for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix}.$$

Solution

We saw in Example 1 that the very poor approximation $(3, -0.0001)^t$ to the exact solution $(1,1)^t$ had a residual vector with small norm, so we should expect the condition number of A to be large.

Solution

We have $\|A\|_\infty = \max\{|1| + |2|, |1.001| + |2|\} = 3.0001$, which would not be considered large. However,

$$A^{-1} = \begin{bmatrix} -10000 & 10000 \\ 5000.5 & -5000 \end{bmatrix}, \qquad \text{so} \qquad \left\|A^{-1}\right\|_\infty = 20000,$$

and for the infinity norm, $K(A) = (20000)(3.0001) = 60002$. The size of the condition number for this example should certainly keep us from making hasty accuracy decisions based on the residual of an approximation.

Normally, if $\|\delta A\|$ and $\|\delta \mathbf{b}\|$ are small (on the order of 10^{-t}), the *t-digit* arithmetic should yield a solution $\tilde{\mathbf{x}}$ for which $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ is correspondingly small.

Realistically, the entries a_{ij} and b_j will be altered or perturbed by an amount δa_{ij} and δb_j, causing the linear system

$$(A + \delta A)\mathbf{x} = \mathbf{b} + \delta \mathbf{b}$$

to be solved in place of $A\mathbf{x} = \mathbf{b}$.

Throughout this section it has been assumed that in the linear system $A\mathbf{x} = \mathbf{b}$, A and $\mathbf{b}$ can be represented exactly.

However, in the case of *ill-conditioned* systems, we have seen that even if A and $\mathbf{b}$ are represented exactly, rounding errors can cause $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ to be large.

The following theorem relates the perturbations of linear systems to the condition number of a matrix.

Theorem 7.29 Suppose A is nonsingular and

$$\|\delta A\| < \frac{1}{\|A^{-1}\|}.$$

The solution $\tilde{\mathbf{x}}$ to $(A + \delta A)\tilde{\mathbf{x}} = \mathbf{b} + \delta\mathbf{b}$ approximates the solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ with the error estimate

$$\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \frac{K(A)\|A\|}{\|A\|-K(A)\|\delta A\|}\left(\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta A\|}{\|A\|}\right). \tag{7.25}$$

The estimate in inequality (7.25) states that if the matrix A is well-conditioned (that is, $K(A)$ is not too large), then small changes in A and $\mathbf{b}$ produce correspondingly small changes in the solution $\mathbf{x}$.

If, on the other hand, A is ill-conditioned, then small changes in A and $\mathbf{b}$ may produce large changes in $\mathbf{x}$.

The theorem is independent of the particular numerical procedure used to solve $A\mathbf{x} = \mathbf{b}$.

It can be shown, by means of a *backward error analysis*, that if Gaussian elimination with pivoting is used to solve $A\mathbf{x} = \mathbf{b}$ in t-digit arithmetic, the numerical solution $\tilde{\mathbf{x}}$ is the actual solution of a linear system:

$$(A + \delta A)\tilde{\mathbf{x}} = \mathbf{b}, \qquad \text{where } \|\delta A\|_\infty \leq f(n)10^{1-t} \max_{i,j,k} \left| a_{ij}^{(k)} \right|.$$

for some function $f(n)$. Wilkinson found that in practice $f(n) \approx n$ and, at worst, $f(n) \leq 1.01(n^3 + 3n^2)$.

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 8-1

Discrete Least Squares Approximation Theory

- Figure 8.1 shows a graph of the values in Table 8.1. It appears that the actual relationship between x and y is linear.

Table 8.1

x_i	y_i	x_i	y_i
1	1.3	6	8.8
2	3.5	7	10.1
3	4.2	8	12.5
4	5.0	9	13.0
5	7.0	10	15.6

Figure 8.1

- The plot obtained (with the data points added) is shown in Figure 8.2. This polynomial is clearly a poor predictor of information between a number of the data points.

Figure 8.1

Figure 8.2

- Let $a_1 x_i + a_0$ denote the ith value on the approximating line and y_i be the ith given y-value. The problem of finding the equation of the best linear approximation in the absolute sense requires that values of a_0 and a_1 be found to minimize

$$E_\infty(a_0, a_1) = \max_{1 \le i \le 10} \{|y_i - (a_1 x_i + a_0)|\}.$$

This is commonly called a **minimax problem**.

- Another approach involves finding values of a_0 and a_1 to minimize

$$E_1(a_0, a_1) = \sum_{i=1}^{10} |y_i - (a_1 x_i + a_0)|.$$

This quantity is called the **absolute deviation**.

To minimize a function of a_0 and a_1, we set its partial derivatives to zero:

$$0 = \frac{\partial}{\partial a_0}\sum_{i=1}^{10}|y_i - (a_1 x_i + a_0)| \text{ and } 0 = \frac{\partial}{\partial a_1}\sum_{i=1}^{10}|y_i - (a_1 x_i + a_0)|.$$

However, the absolute-value function is not differentiable at zero, and we might not be able to find solutions to this pair of equations.

- The **least squares** approach to this problem involves determining the best approximating line. Hence, constants a_0 and a_1 must be found that minimize the least squares error:

$$E_2(a_0, a_1) = \sum_{i=1}^{10}[y_i - (a_1 x_i + a_0)]^2$$

Linear Independent Functions

We consider an new approach to obtain least squares approximations, once $P_n(x)$ is known, it is easy to determine $P_{n+1}(x)$.

Definition 8.1 The set of functions $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is said to be **linearly independent** on $[a, b]$ if, whenever

$$c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0, \text{ for all } x \in [a, b],$$

we have $c_0 = c_1 = \cdots = c_n = 0$.

Otherwise the set of functions is said to be **linearly dependent**.

Theorem 8.2 Suppose that, for each $j = 0, 1, \ldots, n$, $\phi_j(x)$is a polynomial of degree j. Then $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is linearly independent on any interval $[a, b]$.

Example 2 Let $\phi_0(x) = 2$, $\phi_1(x) = x - 3$,and $\phi_2(x) = x^2 + 2x + 7$, and $Q(x) = a_0 + a_1 x + a_2 x^2$. Show that there exist constants c_0,c_1, and c_2 such that $Q(x) = c_0\phi_0(x) + c_1\phi_1(x) + c_2\phi_2(x)$.

Solution: By Theorem 8.2, $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is linearly independent on any interval $[a, b]$. First note that

$$1 = \frac{1}{2}\phi_0(x),\ x = \phi_1(x) + 3 = \phi_1(x) + \frac{3}{2}\phi_0(x),$$

and

$$x^2 = \phi_2(x) - 2x - 7 = \phi_2(x) - 2\left[\phi_1(x) + \frac{3}{2}\phi_0(x)\right] - 7\left[\frac{1}{2}\phi_0(x)\right]$$
$$= \phi_2(x) - 2\phi_1(x) - \frac{13}{2}\phi_0(x).$$

Hence

$$Q(x) = a_0\left[\frac{1}{2}\phi_0(x)\right] + a_1\left[\phi_1(x) + \frac{3}{2}\phi_0(x)\right] + a_2\left[\phi_2(x) - 2\phi_1(x) - \frac{13}{2}\phi_0(x)\right]$$
$$= \left(\frac{1}{2}a_0 + \frac{3}{2}a_1 - \frac{13}{2}a_2\right)\phi_0(x) + [a_2 - a_1]\phi_1(x) + a_2\phi_2(x).$$

The situation illustrated in Example 2 holds in a much more general setting.

Let $\prod_n$ note the **set of all polynomials of degree at most n**.

Theorem 8.3 Suppose that $\{\phi_0(x), \phi_1(x), \cdots, \phi_n(x)\}$ is a collection of linearly independent polynomials in $\prod_n$. Then any polynomial in $\prod_n$ can be written uniquely as a linear combination of $\phi_0(x), \phi_1(x), \cdots, \phi_n(x)$.

The general problem of fitting the best least squares line to a collection of data $\{(x_i, y_i)\}_{i=1}^{m}$ involves minimizing the total error

$$E \equiv E_2(a_0, a_1) = \sum_{i=1}^{10}[y_i - (a_1 x_i + a_0)]^2,$$

with respect to the parameters a_0 *and* a_1. For a minimum to occur, we need both

$$\frac{\partial E}{\partial a_0} = 0 \text{ and } \frac{\partial E}{\partial a_1} = 0.$$

By simplification, we can obtain the **normal equations**:

$$a_0 \cdot m + a_1 \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} y_i \text{ and } a_0 \cdot \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2 = \sum_{i=1}^{m} x_i y_i.$$

The solution to this system of equations is

$$a_0 = \frac{\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i - \sum_{i=1}^{m} x_i y_i \sum_{i=1}^{m} x_i}{m\left(\sum_{i=1}^{m} x_i^2\right) - \left(\sum_{i=1}^{m} x_i\right)^2} \qquad (8.1)$$

and

$$a_1 = \frac{m \sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\left(\sum_{i=1}^{m} x_i^2\right) - \left(\sum_{i=1}^{m} x_i\right)^2}. \qquad (8.2)$$

Example 1 Find the least squares line approximating the data in Table 8.1.

Solution:We first extend the table to include x_i^2 and $x_i y_i$ and sum the columns. This is shown in Table 8.2.

Table 8.2

x_i	y_i	x_i^2	$x_i y_i$	$P(x_i) = 1.538x_i - 0.360$
1	1.3	1	1.3	1.18
2	3.5	4	7.0	2.72
3	4.2	9	12.6	4.25
4	5.0	16	20.0	5.79
5	7.0	25	35.0	7.33
6	8.8	36	52.8	8.87
7	10.1	49	70.7	10.41
8	12.5	64	100.0	11.94
9	13.0	81	117.0	13.48
10	15.6	100	156.0	15.02
55	81.0	385	572.4	$E = \sum_{i=1}^{10}(y_i - P(x_i))^2 \approx 2.34$

The normal equations (8.1) and (8.2) imply that

$$a_0 = \frac{385(81)-55(572.4)}{10(385)-(55)^2} = -0.360 \text{ and } a_1 = \frac{10(572.4)-55(81)}{10(385)-(55)^2} = 1.538,$$

so $P(x) = 1.538x - 0.360$.

The graph of this line and the data points are shown in Figure 8.3.

Figure 8.3

Polynomial Least Squares

- The general problem of approximating a set of data, $\{(x_i, y_i) | i = 1,2,\ldots,m\}$, with an algebraic polynomial

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_z x + a_0, \text{ of gegree } n < m - 1.$$

- We choose the constants $a_0, a_1, \ldots, a_n$ to minimize the least squares error $E = E_2(a_0, a_1, \ldots, a_n)$, where

$$\begin{aligned} E &= \sum_{i=1}^{m} \left(y_i - P_n(x_i)\right)^2 \\ &= \sum_{i=1}^{m} y_i^2 - 2\sum_{i=1}^{m} P_n(x_i) y_i + \sum_{i=1}^{m} \left(P_n(x_i)\right)^2 \\ &= \sum_{i=1}^{m} y_i^2 - 2\sum_{j=0}^{n} a_j \left(\sum_{i=1}^{m} y_i x_i^j\right) + \sum_{j=0}^{n} \sum_{k=0}^{n} a_j\, a_k a_j \left(\sum_{i=1}^{m} x_i^{j+k}\right). \end{aligned}$$

Due to $\frac{\partial E}{\partial a_j} = 0$, for each j, we must have

$$0 = \frac{\partial E}{\partial a_j} = -2\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k}.$$

This gives $n+1$ **normal equations** in the $n+1$ unkowns a_j, these are

$$\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k} = \sum_{i=1}^{m} y_i x_i^j, \text{ for each } j = 0,1,\ldots n.$$

We can obtain

$$a_0 \sum_{i=1}^{m} x_i^0 + a_1 \sum_{i=1}^{m} x_i^1 + \ldots + a_n \sum_{i=1}^{m} x_i^n = \sum_{i=1}^{m} y_i x_i^0,$$
$$a_0 \sum_{i=1}^{m} x_i^1 + a_1 \sum_{i=1}^{m} x_i^2 + \ldots + a_n \sum_{i=1}^{m} x_i^{n+1} = \sum_{i=1}^{m} y_i x_i^1,$$
$$\vdots$$
$$a_0 \sum_{i=1}^{m} x_i^n + a_1 \sum_{i=1}^{m} x_i^{n+1} + \ldots + a_n \sum_{i=1}^{m} x_i^{2n} = \sum_{i=1}^{m} y_i x_i^m.$$

The normal equations have a unique solution provided that the x_i are distinct.

Example 2 Fit the data in Table 8.3 with the discrete least squares polynomial of degree at most 2.

Solution:For this problem, $n = 2$, $m = 5$, and the three normal equations are

Table 8.3

i	x_i	y_i
1	0	1.0000
2	0.25	1.2840
3	0.50	1.6487
4	0.75	2.1170
5	1.00	2.7183

$$5a_0 + 2.5a_1 + 1.875a_2 = 8.7680,$$
$$2.5a_0 + 1.875a_1 + 1.5625a_2 = 5.4514,$$
$$1.785a_0 + 1.5625a_1 + 1.3828a_2 = 4.4015,$$

The least squares polynomial of degree 2 fitting the data in Table 8.3 is

$$P_2(x) = 1.0051 + 0.86468x + 0.84316x^2$$

Figure 8.4

At the given values of x_i we have the approximations shown in Table 8.4.

Table 8.4

i	1	2	3	4	5
x_i	0	0.25	0.50	0.75	1.00
y_i	1.0000	1.2840	1.6487	2.1170	2.7183
$P(x_i)$	1.0051	1.2740	1.6482	2.1279	2.7129
$y_i - P(x_i)$	−0.0051	0.0100	0.0004	−0.0109	0.0054

The total error,

$$E = \sum_{i=1}^{5} \left(y_i - P(x_i)\right)^2 = 2.74 \times 10^{-4}$$

is the least that can be obtained by using a polynomial of degree at most 2.

At times it is appropriate to assume that the data are exponentially related and the function will be

$$y = be^{ax} \qquad (8.4)$$

or

$$y = bx^{a}. \qquad (8.5)$$

The difficulty is to minimizing

$$E = \sum_{i=1}^{m}(y_i - be^{ax_i})^2, \text{ in the case of Eq.(8.4),}$$

$$E = \sum_{i=1}^{m}(y_i - b{x_i}^{a})^2, \text{ in the case of Eq.(8.5),}$$

The corresponding normal equations:

$$0 = \frac{\partial E}{\partial b} = 2\sum_{i=1}^{m}(y_i - be^{ax_i})\,(-e^{ax_i})$$

and

$$0 = \frac{\partial E}{\partial a} = 2\sum_{i=1}^{m}(y_i - be^{ax_i})\,(-bx_i e^{ax_i}),\ \text{in the case of Eq.(8.4);}$$

or

$$0 = \frac{\partial E}{\partial b} = 2\sum_{i=1}^{m}(y_i - bx_i{}^{a})\,(-x_i{}^{a})$$

and

$$0 = \frac{\partial E}{\partial a} = 2\sum_{i=1}^{m}(y_i - bx_i{}^{a})\left(-b(\ln x_i)x_i{}^{a}\right),\ \text{in the case of Eq.(8.5).}$$

No exact solution to either of them in a and b can generally be found.

We usually consider the logarithm of the approximating equation when the data are suspected to be exponentially related:

$$\ln y = \ln b + ax, \text{ in the case of Eq.(8.4),}$$

and

$$\ln y = \ln b + a \ln x, \text{ in the case of Eq.(8.5).}$$

In either case, a linear problem now appears, and solutions for $\ln b$ and a can be obtained by appropriately modifying the normal equations (8.1) and (8.2).

However, the approximation obtained in this manner is not the least squares approximation for the original problem, and this approximation can in some differ significantly from the least squares approximation to the original problem.

Illustration Consider the collection of data in the first three columns of Table 8.5.

Table 8.5

i	x_i	y_i	$\ln y_i$	x_i^2	$x_i \ln y_i$
1	1.00	5.10	1.629	1.0000	1.629
2	1.25	5.79	1.756	1.5625	2.195
3	1.50	6.53	1.876	2.2500	2.814
4	1.75	7.45	2.008	3.0625	3.514
5	2.00	8.46	2.135	4.0000	4.270
	7.50		9.404	11.875	14.422

It is reasonable to assume an approximation of the form

$$y = be^{ax}, \quad \text{which implies that } \ln y = \ln \mathrm{b} + ax.$$

Using the normal equations (8.1) and (8.2),

$$a = \frac{(5)(14.422)-(7.5)(9.404)}{(5)(11.875)-(7.5)^2} = 0.5056 \text{ and } \ln b = \frac{(11.875)(9.404)-(14.422)(7.5)}{(5)(11.875)-(7.5)^2} = 1.122.$$

With $\ln b = 1.122$ we have $b = e^{1.122} = 3.071$, and the approximation assumes the form

$$y = 3.071e^{0.5056x}.$$

Table 8.6

i	x_i	y_i	$3.071e^{0.5056x_i}$	$\lvert y_i - 3.071e^{0.5056x_i}\rvert$
1	1.00	5.10	5.09	0.01
2	1.25	5.79	5.78	0.01
3	1.50	6.53	6.56	0.03
4	1.75	7.45	7.44	0.01
5	2.00	8.46	8.44	0.02

Figure 8.5

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 8-2

Orthogonal Polynomials and Least Squares Approximation

- Consider the approximation of functions.

Suppose $f \in C[a,b]$ and that a polynomial $P_n(x)$ of degree at most n is required that will minimize the error

$$\int_a^b [f(x) - P_n(x)]^2 dx.$$

To determine a least squares approximating polynomial; that is, a polynomial to minimize this expression, let

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_z x + a_0 = \sum_{k=0}^{n} a_k x^k,$$

and we define, as shown in Figure 8.6,

$$E \equiv E_2(a_0, a_1, \cdots, a_n) = \int_a^b [f(x) - \sum_{k=0}^{n} a_k x^k]^2 dx.$$

Figure 8.6

The problem is to find real coefficients $a_0, a_1, \cdots, a_n$ that will minimize E. A necessary condition for the numbers $a_0, a_1, \cdots, a_n$ to minimize E is that

$$\frac{\partial E}{\partial a_j} = 0, \text{ for each } j = 1,2,\cdots,n.$$

Since

$$E = \int_a^b [f(x)]^2 dx - 2\sum_{k=0}^n a_k \int_a^b x^k f(x) dx + \int_a^b [\sum_{k=0}^n a_k x^k]^2 dx,$$

we have

$$\frac{\partial E}{\partial a_j} = -2\int_a^b x^j f(x) dx + 2\sum_{k=0}^n a_k \int_a^b x^{j+k} dx.$$

Hence, to find $P_n(x)$, the $n+1$ linear **normal euqations**

$$\sum_{k=0}^{n} a_k \int_a^b x^{j+k} dx = \int_a^b x^j f(x) dx, \text{ for each } j = 1,2,\cdots,n, \qquad (8.6)$$

must be solved for the $(n+1)$ unknowns a_j.

Example 1 Find the least squares approximating polynomial of degree 2 for the function $f(x) = \sin \pi x$ on the interval $[0,1]$.

Solution: The normal equations for $P_2(x) = a_2x^2 + a_1x + a_0$ are

$$a_0 \int_0^1 1dx + a_1 \int_0^1 x\, dx + a_2 \int_0^1 x^2\, dx = \int_0^1 \sin \pi x\, dx,$$

$$a_0 \int_0^1 xdx + a_1 \int_0^1 x^2\, dx + a_2 \int_0^1 x^3\, dx = \int_0^1 x\sin \pi x\, dx,$$

$$a_0 \int_0^1 x^2dx + a_1 \int_0^1 x^3\, dx + a_2 \int_0^1 x^4\, dx = \int_0^1 x^2\sin \pi x\, dx.$$

Performing the integration yields

$$a_0 + \frac{1}{2}a_1 + \frac{1}{3}a_2 = \frac{2}{\pi},\ \frac{1}{2}a_0 + \frac{1}{3}a_1 + \frac{1}{4}a_2 = \frac{1}{\pi},\ \frac{1}{3}a_0 + \frac{1}{4}a_1 + \frac{1}{5}a_2 = \frac{\pi^2-4}{\pi^3}$$

Then we obtain

$$a_0 = \frac{12\pi^2-120}{\pi^3} \approx -0.050465 \text{ and } a_1 = \frac{720-60\pi^2}{\pi^3} \approx 4.12251.$$

The least squares polynomial approximation of degree 2 for $f(x) = \sin \pi x$ on $[0,1]$ is

$$P_2(x) = -4.12251x^2 + 4.12251x - 0.050465.$$

Ex1 illustrates a difficulty in obtaining a least squares polynomial approximation. An $(n+1)\times(n+1)$ linear system for the unknowns $a_0, a_1, \cdots, a_n$ must be solved, and the coefficients in the linear system are of the form

$$\int_a^b x^{j+k}\,dx = \frac{b^{j+k-1} - a^{j+k+1}}{j+k+1}$$

a linear system that does not have an easily computed numerical solution. The matrix in the linear system is known as a Hilbert matrix, which is a classic example for demonstrating round-off error difficulties.

Another disadvantage: The computations that were performed in obtaining the best nth-degree polynomial, $P_n(x)$, does not lessen the amount of work required to obtain $P_{n+1}(x)$, the polynomial of next higher degree.

Linear Independent Functions

We consider an new approach to obtain least squares approximations, once $P_n(x)$ is known, it is easy to determine $P_{n+1}(x)$.

Definition 8.1 The set of functions $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is said to be **linearly independent** on $[a, b]$ if, whenever

$$c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0, \text{ for all } x \in [a, b],$$

we have $c_0 = c_1 = \cdots = c_n = 0$.

Otherwise the set of functions is said to be **linearly dependent**.

Theorem 8.2 Suppose that, for each $j = 0, 1, \ldots, n$, $\phi_j(x)$is a polynomial of degree j. Then $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is linearly independent on any interval $[a, b]$.

Example 2 Let $\phi_0(x) = 2$, $\phi_1(x) = x - 3$,and $\phi_2(x) = x^2 + 2x + 7$, and $Q(x) = a_0 + a_1 x + a_2 x^2$. Show that there exist constants c_0,c_1, and c_2 such that $Q(x) = c_0\phi_0(x) + c_1\phi_1(x) + c_2\phi_2(x)$.

Solution: By Theorem 8.2, $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is linearly independent on any interval $[a, b]$. First note that

$$1 = \frac{1}{2}\phi_0(x),\ x = \phi_1(x) + 3 = \phi_1(x) + \frac{3}{2}\phi_0(x),$$

and

$$x^2 = \phi_2(x) - 2x - 7 = \phi_2(x) - 2\left[\phi_1(x) + \frac{3}{2}\phi_0(x)\right] - 7\left[\frac{1}{2}\phi_0(x)\right]$$
$$= \phi_2(x) - 2\phi_1(x) - \frac{13}{2}\phi_0(x).$$

Hence

$$Q(x) = a_0\left[\frac{1}{2}\phi_0(x)\right] + a_1\left[\phi_1(x) + \frac{3}{2}\phi_0(x)\right] + a_2\left[\phi_2(x) - 2\phi_1(x) - \frac{13}{2}\phi_0(x)\right]$$
$$= \left(\frac{1}{2}a_0 + \frac{3}{2}a_1 - \frac{13}{2}a_2\right)\phi_0(x) + [a_2 - a_1]\phi_1(x) + a_2\phi_2(x).$$

The situation illustrated in Example 2 holds in a much more general setting.

Let $\prod_n$ note the **set of all polynomials of degree at most n**.

Theorem 8.3 Suppose that $\{\phi_0(x), \phi_1(x), \cdots, \phi_n(x)\}$ is a collection of linearly independent polynomials in $\prod_n$. Then any polynomial in $\prod_n$ can be written uniquely as a linear combination of $\phi_0(x), \phi_1(x), \cdots, \phi_n(x)$.

Orthogonal Functions

Definition 8.4 An integrable function w is called a weight function on the interval I if $w(x) \geq 0$, for all x in I, but $w(x) \not\equiv 0$ on any subinterval of I.

- The purpose of a weight function is to assign varying degrees of importance to approximations on certain portions of the interval. For example, the weight function

$$w(x) = \frac{1}{\sqrt{1-x^2}}$$

places less emphasis near the center of the interval $(-1,1)$ and more emphasis when $|x|$ is near 1.

- Suppose $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is a set of linearly independent functions on $[a, b]$ and w is a weight function for $[a, b]$. Given $f \in C[a, b]$, we seek a linear combination

$$P(x) = \sum_{k=0}^{n} a_k \phi_k(x)$$

to minimize the error

$$E = E(a_0, \cdots, a_n) = \int_a^b w(x)\,[f(x) - \sum_{k=0}^{n} a_k \phi_k(x)]^2 dx.$$

This problem reduces to the situation considered at the beginning of this section in the special case when $w(x) \equiv 1$ and $\phi_k(x) = x^k$, for each $k = 0,1, \cdots, n$.

- The normal equations associated with this problem are derived from the fact that for each $j = 0,1,\cdots,n$,

$$0 = \frac{\partial E}{\partial a_j} = 2\int_a^b w(x)\,[f(x) - \sum_{k=0}^{n} a_k \phi_k(x)]^2 dx.$$

The system of normal equations can be written

$$\int_a^b w(x)\,f(x)\phi_j(x)dx = \sum_{k=0}^{n} a_k \int_a^b w(x)\,\phi_k(x)\phi_j(x)dx \text{ ,for } j = 0,1,\cdots,n.$$

If the functions $\phi_0, \phi_1, \cdots, \phi_n$ can be chosen so that

$$\int_a^b w(x)\,\phi_k(x)\phi_j(x)dx = \begin{cases} 0, & j \neq k, \\ \alpha_j > 0 & j = k, \end{cases} \qquad (8.7)$$

then the normal equations will reduce to

$$\int_a^b w(x)\,f(x)\phi_j(x)dx = a_j \int_a^b w(x)\left[\phi_j(x)\right]^2 dx = a_j\alpha_j \text{ ,for each } j = 0,1,\cdots,n.$$

These are easily solved to give

$$a_j = \frac{1}{\alpha_j}\int_a^b w(x)\, f(x)\phi_j(x)dx.$$

Hence the least squares approximation problem is greatly simplified when the functions $\phi_0, \phi_1, \cdots, \phi_n$ are chosen to satisfy the *orthogonality* condition in Eq. (8.7).

Definition 8.5 $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is said to be an **orthogonal set of functions** for the interval $[a, b]$ with respect to the weight function w if

$$\int_a^b w(x)\,\phi_k(x)\phi_j(x)dx = \begin{cases} 0, & j \neq k, \\ \alpha_j > 0 & j = k, \end{cases}$$

If, in addition, $\alpha_j = 1$ for each $j = 0,1,\cdots,n$, the set is said to be **orthogonal**.

Theorem 8.6 If $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is an orthogonal set of functions on an interval $[a, b]$ with respect to the weight function w, then

$$P(x) = \sum_{j=0}^{n} a_j \phi_j\,(x)$$

where, for each $j = 0,1,\cdots,n$,

$$a_j = \frac{\int_a^b w(x) f(x)\phi_j(x)dx}{\int_a^b w(x)\left[\phi_j(x)\right]^2 dx} = \frac{1}{\alpha_j}\int_a^b w(x)\,f(x)\phi_j(x)dx.$$

Gram-Schmidt process

Definition 8.7 The set of polynomial functions $\{\phi_0, \phi_1, \cdots, \phi_n\}$ defined in the following way is orthogonal on $[a, b]$ with respect to the weight function w

$$\phi_0(x) \equiv 1, \phi_1(x) = x - B_1, \text{ for each } x \text{ in } [a, b],$$

where

$$B_1 = \frac{\int_a^b xw(x)[\phi_0(x)]^2 dx}{\int_a^b w(x)[\phi_0(x)]^2 dx},$$

and when $k \geq 2$,

$$\phi_k(x) = (x - B_k)\phi_{k-1}(x) - C_k\phi_{k-2}(x), \text{ for each } x \text{ in } [a, b],$$

where

$$B_k = \frac{\int_a^b xw(x)[\phi_{k-1}(x)]^2 dx}{\int_a^b w(x)[\phi_{k-1}(x)]^2 dx} \text{ and } C_k = \frac{\int_a^b xw(x)\phi_{k-1}(x)\phi_{k-2}(x) dx}{\int_a^b w(x)[\phi_{k-2}(x)]^2 dx}.$$

Corollary 8.8 For any $n > 0$, the set of polynomial functions $\{\phi_0, \phi_1, \cdots, \phi_n\}$ given in Theorem 8.7 is linearly independent on $[a, b]$ and

$$\int_a^b w(x)\,\phi_n(x)Q_k(x)dx = 0,$$

for any polynomial $Q_k(x)$ of degree $k < n$.

Proof: For each $k = 0,1,\cdots,n$, $\phi_k(x)$ is a polynomial of degree k. So Theorem 8.2 implies that $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is a linearly independent set.

Let $Q_k(x)$ be a polynomial of degree k. By Theorem 8.3 there exist numbers $c_0, c_1, \cdots, c_k$ such that

$$Q_k(x) = \sum_{j=0}^{k} c_j \phi_j(x).$$

Because ϕ_n is orthogonal to ϕ_j for each $j = 1,2,\cdots,k$ we have

$$\int_a^b w(x)\,\phi_n(x)Q_k(x)dx = \sum_{j=0}^{k} c_j \int_a^b w(x)\,\phi_j(x)\phi_n(x)dx = \sum_{j=0}^{k} c_j \cdot 0 = 0.$$

Illustration The set of **Legendre polynomials**, $\{P_n(x)\}$, is orthogonal on $[-1,1]$ with respect to the weight function $w(x) \equiv 1$.

Using the Gram-Schmidt process with $P_0(x) \equiv 1$ gives

$$B_1 = \frac{\int_{-1}^{1} x dx}{\int_{-1}^{1} dx} = 0 \text{ and } P_1(x) = (x - B_1)P_0(x) = x.$$

Also,

$$B_2 = \frac{\int_{-1}^{1} x^3 dx}{\int_{-1}^{1} x^2 dx} = 0 \text{ and } C_2 = \frac{\int_{-1}^{1} x^2 dx}{\int_{-1}^{1} 1 dx} = \frac{1}{3},$$

so

$$P_2 = (x - B_2)P_1(x) - C_2 P_0(x) = (x - 0)x - \frac{1}{3} \cdot 1 = x^2 - \frac{1}{3}.$$

The higher-degree Legendre polynomials shown in Figure 8.9 are derived in the same manner.

Figure 8.9

UNIVERSITY OF LEICESTER

DLI

大连理工大学 DALIAN UNIVERSITY OF TECHNOLOGY

Lecture 8-3

Chebyshev Polynomials and Economization of Power Series

The Chebyshev polynomials $\{T_n(x)\}$ are orthogonal on $(-1,1)$ with respect to the weight function $w(x) = (1-x^2)^{-1/2}$.

- For $x \in [-1,1]$,define

$$T_n(x) = \cos[n\arccos x\,] \text{ ,for each } n \geq 0.$$

- We will now show that for each n, $T_n(x)$ is a polynomial in x.

First note that

$$T_0(x) = \cos 0 = 1 \text{ and } T_1(x) = \cos[n\arccos x\,] = x.$$

For $n \geq 1$, we introduce $\theta = \arccos x$, then

$$T_n\big(\theta(x)\big) \equiv T_n(\theta) = \cos n\theta \text{ ,where } \theta \in [0,\pi].$$

A recurrence relation is derived by noting that

$$T_{n+1}(\theta) = \cos(n+1)\theta = \cos\theta\cos(n\theta) - \sin\theta\sin(n\theta)$$
$$T_{n-1}(\theta) = \cos(n-1)\theta = \cos\theta\cos(n\theta) + \sin\theta\sin(n\theta)$$

Adding these equations gives

$$T_{n+1}(\theta) = 2\cos\theta\cos(n\theta) - T_{n-1}(\theta).$$

Returning to the variable $x = \cos\theta$,we have, for $n \geq 1$,

$$T_{n+1}(x) = 2x\cos(n\mathrm{arc}cosx) - T_{n-1}(x),$$

that is,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \qquad (8.9)$$

Because $T_0(x) = 1$ and $T_1(x) = x$, the recurrence relation implies that the next three Chebyshev polynomials are

$$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1,$$
$$T_3(x) = 2xT_2(x) - T_1(x) = 4x^3 - 3x,$$
$$T_4(x) = 2xT_3(x) - T_2(x) = 8x^4 - 8x^2 + 1.$$

The recurrence relation also implies that when $n \geq 1$, $T_n(x)$ is a polynomial of degree n with leading coefficient 2^{n-1}.

The graphs of T_1,T_2,T_3, and T_4 are shown in Figure 8.10.

Figure 8.10

To show the orthogonality of the Chebyshev polynomials with respect to the weight function $w(x) = (1 - x^2)^{-1/2}$, consider

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \int_{-1}^{1} \frac{\cos(n\arccos x)\cos(m\arccos x)}{\sqrt{1-x^2}} dx.$$

Reintroducing the substitution $\theta = \arccos x$ gives

$$d\theta = -\frac{1}{\sqrt{1-x^2}} dx$$

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = -\int_{\pi}^{0} \cos(n\theta)\cos(m\theta)\, d\theta = \int_{0}^{\pi} \cos(n\theta)\cos(m\theta)\, d\theta.$$

Suppose $n \neq m$. Since

$$\cos(n\theta)\cos(m\theta) = \frac{1}{2}[\cos(n+m)\theta + \cos(n-m)\theta],$$

we have

$$\begin{aligned}\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\,dx &= \frac{1}{2}\int_0^{\pi} \cos\big((n+m)\theta\big)d\theta + \frac{1}{2}\int_0^{\pi} \cos\big((n-m)\theta\big)d\theta \\ &= \left[\frac{1}{2(n+m)}\sin\big((n+m)\theta\big) + \frac{1}{2(n-m)}\sin\big((n-m)\theta\big)\right]_0^{\pi} \\ &= 0\end{aligned}$$

Then

$$\int_{-1}^{1} \frac{[T_n(x)]^2}{\sqrt{1-x^2}}\,dx = \frac{\pi}{2}, \text{ for each } n \geq 1. \qquad (8.10)$$

Theorem 8.9 The Chebyshev polynomial $T_n(x)$ of degree $n \geq 1$ has n simple zeros in $[-1,1]$ at

$$\bar{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right), \text{ for each } k = 1,2,\cdots,n.$$

Moreover, $T_n(x)$ assumes its absolute extrema at

$$\bar{x}_k^{'} = \cos\left(\frac{k\pi}{n}\right) \text{ with } T_n\left(\bar{x}_k^{'}\right) = (-1)^k, \text{ for each } k = 0,1,2,\cdots,n.$$

Proof: Let

$$\bar{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right), \text{ for each } k = 1,2,\cdots,n.$$

Then

$$T_n(\bar{x}_k) = \cos(n\arccos\bar{x}_k) = \cos\left(n\arccos\left(\cos\left(\frac{2k-1}{2n}\pi\right)\right)\right) = \cos\left(\frac{2k-1}{2}\pi\right) = 0.$$

The $\bar{x}_k$ are distinct and $T_n(x)$ is a polynomial of degree n, so all the zeros of $T_n(x)$ must have this form.

Note that

$$T_n^{'}(x) = \frac{d}{dx}[\cos(n\arccos x)] = \frac{n\sin(n\arccos x)}{\sqrt{1-x^2}},$$

and that, when $k = 1,2,\cdots,n-1$,

$$T_n^{'}(\bar{x}_k^{'}) = \frac{n\sin\left(n\arccos\left(\cos\left(\frac{k\pi}{n}\right)\right)\right)}{\sqrt{1-\left[\cos\left(\frac{k\pi}{n}\right)\right]^2}} = \frac{n\sin(k\pi)}{\sin\left(\frac{k\pi}{n}\right)} = 0.$$

Since $T_n(x)$ is a polynomial of degree n, its derivative $T_n^{'}(x)$ is a polynomial of degree $(n-1)$, and all the zeros of $T_n^{'}(x)$ occur at these $n-1$ distinct points.

The only other possibilities for extrema of $T_n(x)$ occur at the endpoints of the interval $[-1,1]$; that is, at $\bar{x}_0^{'} = 1$ and at $\bar{x}_n^{'} = -1$.
For any $k = 0,1,2,\cdots,n$ we have

$$T_n^{'}(\bar{x}_k^{'}) = \cos\left(n\arccos\left(\cos\left(\frac{k\pi}{n}\right)\right)\right) = \cos k\pi = (-1)^k.$$

So a maximum occurs at each even value of k and a minimum at each odd value.

The monic (polynomials with leading coefficient 1) Chebyshev polynomials $\tilde{T}_n(x)$ are derived from the Chebyshev polynomials $T_n(x)$ by dividing by the leading coefficient 2^{n-1}.

Hence

$$\tilde{T}_0(x) = 1 \text{ and } \tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x), \text{ for each } n \geq 1. \qquad (8.11)$$

The recurrence relationship satisfied by the Chebyshev polynomials implies that

$$\tilde{T}_2(x) = x\tilde{T}_1(x) - \frac{1}{2}\tilde{T}_0(x) \text{ and} \qquad (8.12)$$

$$\tilde{T}_{n+1}(x) = x\tilde{T}_n(x) - \frac{1}{4}\tilde{T}_{n-1}(x), \text{ for each } n \geq 1.$$

The graphs of $\tilde{T}_1$, $\tilde{T}_2$, $\tilde{T}_3$, $\tilde{T}_4$, and $\tilde{T}_5$ are shown in Figure 8.11.

Figure 8.11

Because $\tilde{T}_n(x)$ is just a multiple of $T_n(x)$, Theorem 8.9 implies that the zeros of $\tilde{T}_n(x)$ also occur at

$$\bar{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right), \text{ for each } k = 1,2,\cdots,n.$$

and the extreme values of $\tilde{T}_n(x)$, for $n \geq 1$, occur at

$$\bar{x}_k^{'} = \cos\left(\frac{k\pi}{n}\right), \text{ with } T_n^{'}\left(\bar{x}_k^{'}\right) = \frac{(-1)^k}{2^{n-1}}, \text{ for each } k = 0,1,2,\cdots,n. \qquad (8.13)$$

Let $\prod_n$ denote **the set of all monic polynomials of degree *n***. The relation expressed in Eq. (8.13) leads to an important minimization property that distinguishes $\tilde{T}_n(x)$ from the other members of $\prod_n$.

Theorem 8.10 The polynomials of the form $\tilde{T}_n(x)$, when degree $n \geq 1$, have the property that

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} \left|\tilde{T}_n(x)\right| \leq \max_{x\in[-1,1]} |P_n(x)|, \text{ for all } P_n(x) \in \tilde{\prod}_n.$$

Moreover, equality occurs only if $P_n = \tilde{T}_n$.

Proof: Suppose that $P_n(x) \in \tilde{\prod}_n$ and that

$$\max_{x\in[-1,1]} |P_n(x)| \leq \frac{1}{2^{n-1}} = \max_{x\in[-1,1]} \left|\tilde{T}_n(x)\right|.$$

Let $Q = \tilde{T}_n - P_n$. Then $\tilde{T}_n(x)$ and $P_n(x)$ are both monic polynomials of degree n, so $Q(x)$ is a polynomial of degree at most $n-1$. Moreover, at the $n+1$ extreme points $\bar{x}_k^{'}$ of $\tilde{T}_n(x)$, we have

$$Q\left(\bar{x}_k^{'}\right) = \tilde{T}_n\left(\bar{x}_k^{'}\right) - P_n\left(\bar{x}_k^{'}\right) = \frac{(-1)^k}{2^{n-1}} - P_n\left(\bar{x}_k^{'}\right).$$

However

$$\left|P_n\left(\bar{x}_k^{'}\right)\right| \leq \frac{1}{2^{n-1}}, \text{ for each } k = 0,1,\cdots,n,$$

so we have

$Q\left(\bar{x}_k^{'}\right) \leq 0$, when k is odd and $Q\left(\bar{x}_k^{'}\right) \geq 0$, when k is even.

Since Q is continuous, the Intermediate Value Theorem implies that for each $j = 0,1,\cdots,n-1$ the polynomial $Q(x)$ has at least one zero between $\bar{x}_j^{'}$ and $\bar{x}_{j+1}^{'}$.

Thus, Q has at least n zeros in the interval $[-1,1]$. But the degree of $Q(x)$ is less than n, so $Q \equiv 0$. This implies that $P_n \equiv \tilde{T}_n$.

The Chebyshev polynomials are used to minimize approximation error.

- an optimal placing of interpolating points to minimize the error in Lagrange interpolation;
- a means of reducing the degree of an approximating polynomial with minimal loss of accuracy.

Minimizing Lagrange Interpolation Error

Theorem 8.10 can be used to answer the question of where to place interpolating nodes to minimize the error in Lagrange interpolation. Theorem 3.3 applied to the interval $[-1,1]$ states that, if $x_0, \cdots, x_n$ are distinct numbers in the interval $[-1,1]$ and if $f \in C^{n+1}[-1,1]$,then, for each $x \in [-1,1]$,a number $\xi(x)$ exists in $(-1,1)$ with

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n),$$

where $P(x)$ is the Lagrange interpolating polynomial.

Generally, there is no control over $\xi(x)$, so to minimize the error by shrewd placement of the nodes $x_0, \cdots, x_n$, we choose $x_0, \cdots, x_n$ to minimize the quantity

$$|(x - x_0)(x - x_1) \cdots (x - x_n)|$$

throughout the interval $[-1,1]$.

Since $(x - x_0)(x - x_1) \cdots (x - x_n)$ is a monic polynomial of degree $n + 1$, we have just seen that the minimum is obtained when

$$(x - x_0)(x - x_1) \cdots (x - x_n) = \tilde{T}_{n+1}(x).$$

The maximum value of $|(x - x_0)(x - x_1) \cdots (x - x_n)|$ is smallest when x_k is chosen for each $k = 0,1, \cdots, n$ to be the $k + 1$st zero of $\tilde{T}_{n+1}$. Hence we choose x_k to be

$$\bar{x}_{k+1} = \cos\left(\frac{2k+1}{2(n+1)}\pi\right).$$

Because $\max_{x\in[-1,1]}\left|\tilde{T}_{n+1}(x)\right| = 2^{-n}$, this also implies that

$$\frac{1}{2^n} = \max_{x\in[-1,1]}\left|(x-\bar{x}_1)\cdots(x-\bar{x}_{n+1})\right| \le \left|(x-x_0)\cdots(x-x_n)\right|,$$

for any choice of $x_0, x_1, \cdots, x_n$ in the interval $[-1,1]$. The next corollary follows from these observations.

Corollary 8.11 Suppose that $P(x)$ is the interpolating polynomial of degree at most n with nodes at the zeros of $T_{n+1}(x)$. Then

$$\max_{x\in[-1,1]}\left|f(x)-P(x)\right| \le \frac{1}{2^n(n+1)!}\max_{x\in[-1,1]}\left|f^{(n+1)}(x)\right|, \text{ for each } f \in C^{n+1}[-1,1]$$

Minimizing Approximation Error on Arbitrary Intervals

The technique for choosing points to minimize the interpolating error is extended to a general closed interval [a, b] by using the change of variables

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

to transform the numbers $\bar{x}_k$ in the interval $[-1,1]$ into the corresponding number $\bar{x}_k$ in the interval $[a, b]$, as shown in the next example.

Example 1 Let $f(x) = xe^x$ on [0,1.5]. Compare the values given by the Lagrange polynomial with four equally-spaced nodes with those given by the Lagrange polynomial with nodes given by zeros of the fourth Chebyshev polynomial.

Solution: The equally-spaced nodes $x_0 = 0$, $x_1 = 0.5$, $x_2 = 1$, and $x_3 = 1.5$ give

$$L_0(x) = -1.3333x^3 + 4.0000x^2 - 3.6667x + 1,$$
$$L_1(x) = 4.0000x^3 - 10.000x^2 + 6.0000x,$$
$$L_2(x) = -4.0000x^3 + 8.0000x^2 - 3.0000x,$$
$$L_3(x) = 1.3333x^3 - 2.000x^2 + 0.6667x,$$

which produces the polynomial

$$P_3(x) = L_0(x)(0) + L_1(x)(0.5e^{0.5}) + L_2(x)(1.5e^{1.5})$$
$$= 1.3875x^3 + 0.057570x^2 + 1.2730x.$$

For the second interpolating polynomial, we shift the zeros $\bar{x}_k = \cos\left(\frac{2k+1}{8}\pi\right)$, for $k = 0,1,2,3$, of $\tilde{T}_4$ from $[-1,1]$ to $[-0,1.5]$, using the linear transformation

$$\tilde{x}_k = \frac{1}{2}[(1.5 - 0)\bar{x}_k + (1.5 + 0)] = 0.75 + 0.75\bar{x}_k.$$

Because

$$\bar{x}_0 = \cos\frac{\pi}{8} = 0.92388,\ \bar{x}_1 = \cos\frac{3\pi}{8} = 0.38268,$$
$$\bar{x}_2 = \cos\frac{5\pi}{8} = -0.38268,\ \bar{x}_3 = \cos\frac{7\pi}{8} = -0.92388,$$

we have

$$\tilde{x}_0 = 1.44291,\ \tilde{x}_1 = 1.0370,\ \tilde{x}_2 = 0.46299,\ \tilde{x}_3 = 0.05709.$$

The Lagrange coefficient polynomials for this set of nodes are

$$\tilde{L}_0(x) = 1.8142x^3 - 2.8249x^2 + 1.0264x - 0.049728,$$
$$\tilde{L}_1(x) = -4.3799x^3 + 8.5977x^2 - 3.4026x + 0.16705,$$
$$\tilde{L}_2(x) = 4.3799x^3 - 11.112x^2 + 7.17738x - 0.37415,$$
$$\tilde{L}_3(x) = -1.8142x^3 + 5.3390x^2 - 4.7976x + 1.2568.$$

The functional values required for these polynomials are given in the last two columns of Table 8.7. The interpolation polynomial of degree at most 3 is

$$\tilde{P}_3(x) = 1.3811x^3 + 0.044652x^2 + 1.3031x - 0.014352.$$

Table 8.7

x	$f(x) = xe^x$	$\tilde{x}$	$f(\tilde{x}) = xe^x$
$x_0 = 0.0$	0.00000	$\tilde{x}_0 = 1.44291$	6.10783
$x_1 = 0.5$	0.824361	$\tilde{x}_1 = 1.03701$	2.92517
$x_2 = 1.0$	2.71828	$\tilde{x}_2 = 0.46299$	0.73560
$x_3 = 1.5$	6.72253	$\tilde{x}_3 = 0.05709$	0.060444

For comparison, Table 8.8 lists various values of x, together with the values of $f(x)$, $P_3(x)$, and $\tilde{P}_3(x)$.

Although the error using $P_3(x)$ is less than using $\tilde{P}_3(x)$ near the middle of the table, the maximum error involved with using $\tilde{P}_3(x)$, 0.0180, is considerably less than when using $P_3(x)$, which gives the error 0.0290.(See Figure 8.12.)

Table 8.8

x	$f(x) = xe^x$	$P_3(x)$	$\lvert xe^x - P_3(x)\rvert$	$\tilde{P}_3(x)$	$\lvert xe^x - \tilde{P}_3(x)\rvert$
0.15	0.1743	0.1969	0.0226	0.1868	0.0125
0.25	0.3210	0.3435	0.0225	0.3358	0.0148
0.35	0.4967	0.5121	0.0154	0.5064	0.0097
0.65	1.245	1.233	0.012	1.231	0.014
0.75	1.588	1.572	0.016	1.571	0.017
0.85	1.989	1.976	0.013	1.974	0.015
1.15	3.632	3.650	0.018	3.644	0.012
1.25	4.363	4.391	0.028	4.382	0.019
1.35	5.208	5.237	0.029	5.224	0.016

It can be seen from Figure 8.12 that, the maximum error involved with using $\tilde{P}_3(x)$ is very small.

Figure 8.12

Reducing the Degree of Approximating Polynomials

- Chebyshev polynomials can also be used to reduce the degree of an approximating polynomial with a minimal loss of accuracy.

Consider approximating an arbitrary nth-degree polynomial

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

on $[-1,1]$ with a polynomial of degree at most $n-1$. The object is to choose $P_{n-1}(x)$ in $\prod_{n-1}$ so that

$$\max_{x \in [-1,1]} |P_n(x) - P_{n-1}(x)|$$

is as small as possible.

Note that $|P_n(x) - P_{n-1}(x)|/a_n$ is a monic polynomial of degree n, so applying Theorem 8.10 gives

$$\max_{x\in[-1,1]} \left|\frac{1}{a_n}\left(P_n(x) - P_{n-1}(x)\right)\right| \geq \frac{1}{2^{n-1}}.$$

Equality occurs precisely when

$$\frac{1}{a_n}\left(P_n(x) - P_{n-1}(x)\right) = \tilde{T}_n(x).$$

This means that we should choose

$$P_{n-1}(x) = P_n(x) - a_n\tilde{T}_n(x),$$

and with this choice we have the minimum value of

$$\max_{x\in[-1,1]} |P_n(x) - P_{n-1}(x)| = |a_n| \max_{x\in[-1,1]} \left|\frac{1}{a_n}\left(P_n(x) - P_{n-1}(x)\right)\right| = \frac{|a_n|}{2^{n-1}}.$$

Illustration The function $f(x) = e^x$ is approximated on the interval $[-1,1]$ by the fourth Maclaurin polynomial

$$P_4(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24},$$

which has truncation error

$$|R_4(x)| = \frac{|f^{(5)}(\xi(x))||x^5|}{120} \leq \frac{e}{120} \approx 0.023, \text{ for } -1 \leq x \leq 1.$$

Suppose that an error of 0.05 is tolerable and that we would like to reduce the degree of the approximating polynomial while staying within this bound.

The polynomial of degree 3 or less that best uniformly approximates $P_4(x)$ on $[-1,1]$ is

$$P_3(x) = P_4(x) - a_4\tilde{T}_4(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} - \frac{1}{24}\left(x^4 - x^2 + \frac{1}{8}\right)$$
$$= \frac{191}{192} + x + \frac{1}{24}x^2 + \frac{1}{6}x^3.$$

With this choice, we have

$$|P_4(x) - P_3(x)| = \left|a_4\tilde{T}_4(x)\right| \leq \frac{1}{24} \cdot \frac{1}{2^3} = \frac{1}{192} \leq 0.0053.$$

Adding this error bound to the bound for the Maclaurin truncation error gives

$$0.023 + 0.0053 = 0.0283,$$

which is within the permissible error of 0.05.

The polynomial of degree 2 or less that best uniformly approximates $P_3(x)$ on $[-1,1]$ is

$$P_2(x) = P_3(x) - \frac{1}{6}\tilde{T}_3(x) = \frac{191}{192} + x + \frac{13}{24}x^2 + \frac{1}{6}x^3 - \frac{1}{6}\left(x^3 - \frac{3}{4}x\right)$$
$$= \frac{191}{192} + \frac{9}{8}x + \frac{13}{24}x^2.$$

However,

$$|P_3(x) - P_2(x)| = \left|\frac{1}{6}\tilde{T}_3(x)\right| = \frac{1}{6}\cdot\left(\frac{1}{2}\right)^2 = \frac{1}{24} \leq 0.042.$$

which—when added to the already accumulated error bound of 0.0283—exceeds the tolerance of 0.05. Consequently, the polynomial of least degree that best approximates e^x on $[-1,1]$ with an error bound of less than 0.05 is

$$P_3(x) = \frac{191}{192} + x + \frac{13}{24}x^2 + \frac{1}{6}x^3.$$

Table 8.9 lists the function and the approximating polynomials at various points in $[-1,1]$. Note that the tabulated entries for P_2 are well within the tolerance of 0.05, even though the error bound for $P_2(x)$ exceeded the tolerance.

Table 8.9

x	e^x	$P_4(x)$	$P_3(x)$	$P_2(x)$	$\lvert e^x - P_2(x)\rvert$
−0.75	0.47237	0.47412	0.47917	0.45573	0.01664
−0.25	0.77880	0.77881	0.77604	0.74740	0.03140
0.00	1.00000	1.00000	0.99479	0.99479	0.00521
0.25	1.28403	1.28402	1.28125	1.30990	0.02587
0.75	2.11700	2.11475	2.11979	2.14323	0.02623