Chapter 1: Mathematical Preliminaries

刘保东
山东大学，计算机科学与技术学院
Email:baodong@sdu.edu.cn

教材、辅助教材与考试考核方法

- **教材:** 数值分析（第七版，影印版），Richard L.Burden,J.Douglas Faires. 高等教育出版社，2001.
- **参考书:**
  1. 数值方法：MATLAB版（原书第四版），[美] Mathews, J.H., Fink, K.D.; 周璐等译，北京：电子工业出版社，2010.
  2. 数值计算引论(第2版)，白峰杉，高等教育出版社，2010年
  3. 数值分析（原书第2版）,[美] Timothy Sauer 著；裴玉茹，马赓宇译, 机械工业出版社,2014年
  4. Numerical Recipes in C++，PRESS, WILLIAM H. TEUKOLSKY, SAUL A. VETTERLING, WILLIAM T. FLANNERY, BRIAN P,CAMBRIDGE UNIV PRESS,2005.
  5. Scientific Computing: An Introductory Survey, Second Edition，McGraw-Hill,2002, 清华大学出版社出版影印发行
- **成绩计算:** 数值实验报告（40%）+平时考勤（10%）+期末闭卷考试(50%)

What is Scientific Computing?

- **Equivalent description**
  - **Numerical analysis**
  - **Numerical Methods or Computational Methods**
  - **Numerical Mathematics**
  - **Numerical Computing**
  - **Scientific Computing** etc.

- **Main Contents:**
  1. **Methods** using computer to solve **mathematical problems** in science and engineering, which mostly are **continuous**.
  2. **design and analysis of algorithms** for different mathematical problems.
  3. **Theory and Application** of Numerical Approximation Techniques

Why do we need to learn Scientific Computing methods?

- Many mathematical problems arising in the science and engineering, such as derivatives, integrals, nonlinearities, Linear Algebra problems, differential equations, etc. are **difficult to solve**.
- The fast development of PC techniques and widespread use of computers made it possible to solve mathematical problems with the help of computers.

Why do we need to learn Scientific Computing methods?

- Many mathematical problems arising in the science and engineering, such as derivatives, integrals, nonlinearities, Linear Algebra problems, differential equations, etc. are **difficult to solve**.
- The fast development of PC techniques and widespread use of computers made it possible to solve mathematical problems with the help of computers.

- Deals continuous quantities with numerical approximate techniques;
- Considers **effects** of approximations, such as **error**, **convergence**, **uniqueness**, **existence**.
- **Assessment** on algorithm: **efficiency**, **reliability**, **accuracy**, etc.

Features of Scientific Computing:

- Deals continuous quantities with numerical approximate techniques;
- Considers **effects** of approximations, such as **error**, **convergence**, **uniqueness**, **existence**.
- **Assessment** on algorithm: **efficiency**, **reliability**, **accuracy**, etc.

Features of Scientific Computing:

- Deals continuous quantities with numerical approximate techniques;
- Considers **effects** of approximations, such as **error**, **convergence**, **uniqueness**, **existence**.
- **Assessment** on algorithm: **efficiency**, **reliability**, **accuracy**, etc.

What does this book concern?

- **Approximation Methods** for solving equation(s);
- **Polynomial and interpolation approximation**;
- **Numerical Differentiation and Integration**.
- **Numerical methods for ODE or PDE**
- **Eigenvalues and Eigenvectors**

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Steps for solving Mathematical Problems in computational simulations

1. **Mathematical modelling**, usually equations.
2. **Design algorithms** to solve these equations.
3. Implement algorithms in computer software.
4. **Run the software**
5. **Represent** the computed results in forms or graphical visualization.
6. **Interpret and validate**(解释和验证) the computed results.

Definition 1.1 Limit of a function(函数的极限)

Let f be a function defined on a set X of real numbers. Then f has the **limit L** at x_o, written

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

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 $0 < |x - x_o| < \delta$.

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.

- Especially, let $C(X)$ denote the set of all functions that are continuous on the set X.
- When X is an closed interval $[a, b]$, the set of all functions that are continuous on the interval $[a, b]$ is denoted by $C[a, b]$.

Definition 1.3

Let $\{x_n\}_{n=1}^{\infty}$ be an infinite sequence of real or complex number. The sequence converges to a number x (Limit) if, for any $\varepsilon > 0$, there exists a positive integer $N(\varepsilon)$, such that implies

$$|x_n - x| < \varepsilon,$$

whenever $n > N(\varepsilon)$.

Noted by

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

or $x_n \to x \quad$ as $\quad n \to \infty$.

Theorem 1.4

If f is a function defined on a set 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

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

Definition 1.5

If f is a function defined in an open interval containing x_0, then 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 $f'(x_0)$ ia called the **derivative** of $f(x)$ at x_0.

Notes:

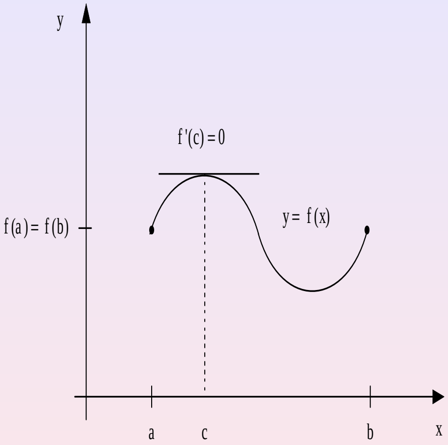
- $C^n(X)$ denote the set of all functions that have n continuous derivatives on X.
- Especially $C^\infty(X)$ denote the set of all functions that have derivatives of all orders on X

Theorem 1.6

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

Theorem 1.7 (Rolle's Theorem):

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


y
f'(c)=0
f(a)= f(b)
y= f(x)
a
c
b
x

Theorem 1.8 (Mean Value Theorem – 均值定理)

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

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

Theorem 1.9 (Extreme Value Theorem – 极值定理)

- If $f \in C[a, b]$, then $c_1, c_2 \in [a, b]$ exist with $f(c_1) \le f(x) \le f(c_2)$ for each $x \in [a, b]$.
- If, in addition, 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.

Definition 1.10

The **Riemann Integral** of a function on an interval $[a, b]$ is the following limit, provided it exists:

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

where the numbers $x_0, x_1, x_2, \cdots, x_n$ satisfy $a = x_0 \leq x_1 \leq x_1 \leq \cdots \leq x_n = b$, and where $\Delta x_i = x_i - x_{i-1}$ for each $i = 1, 2, \cdots, n$ and z_i is an arbitrarily chosen in the interval $[x_{i-1}, x_i]$.

Especially, if we choose $z_i = x_i$ and $\Delta x_i = (b - a)/n$, then in this case

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

Theorem 1.11 (Weighted Mean Value Theorem for the Integral)

If $f \in C[a, b]$, the Riemann Integral of g exists on the $[a, b]$, and $g(x)$ does not change sign on $[a, b]$, then there exists a number in (a, b) with

$$\int_a^b f(x)g(x)\mathrm{d}x = f(c)\int_a^b g(x)\mathrm{d}x. \blacksquare$$

When $g(x) \equiv 1$, this theorem give the average value of the function f over the interval $[a, b]$.

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

Theorem 1.12 (Generalized Rolle's Theorem)

Suppose $f \in C[a, b]$ is n times differentiable on (a, b). If $f(x)$ is zero at the $n+1$ distinct numbers $x_0, x_1, x_2, \cdots, x_n$ in the $[a, b]$, then a number c in the (a, b) exists with

$$f^{(n)}(c) = 0. \blacksquare$$

Theorem 1.13 (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 $f(c) = K$.

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

$$\begin{aligned} P_n(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \\ &+ \cdots + \frac{f^n(x_0)}{n!}(x - x_0)^n \\ &= \sum_{0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k, \end{aligned}$$

and

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

Example 1: see $(\sqrt{3})^2 = 3$

- this is true in traditional arithmetic in algebra or calculus, but if we use calculator or computer to do, what will happen?
- In our traditional mathematical world, we permit number with an infinite number of digits;
- But in arithmetic, we define $\sqrt{3}$ as an unique positive number, so when it is multiplied by itself, we can get 3.
- In computer computation, $\sqrt{3}$ first is represented with a fixed, finite number of digits, which may be very closed to its exact value. This means only rational (有理数) number can be presented exactly.

Example 1: see $(\sqrt{3})^2 = 3$

- this is true in traditional arithmetic in algebra or calculus, but if we use calculator or computer to do, what will happen?
- In our traditional mathematical world, we permit number with an infinite number of digits;
- But in arithmetic, we define $\sqrt{3}$ as an unique positive number, so when it is multiplied by itself, we can get 3.
- In computer computation, $\sqrt{3}$ first is represented with a fixed, finite number of digits, which may be very closed to its exact value. This means only rational (有理数) number can be presented exactly.

Example 1: see $(\sqrt{3})^2 = 3$

- this is true in traditional arithmetic in algebra or calculus, but if we use calculator or computer to do, what will happen?
- In our traditional mathematical world, we permit number with an infinite number of digits;
- But in arithmetic, we define $\sqrt{3}$ as an unique positive number, so when it is multiplied by itself, we can get 3.
- In computer computation, $\sqrt{3}$ first is represented with a fixed, finite number of digits, which may be very closed to its exact value. This means only rational (有理数) number can be presented exactly.

Example 1: see $(\sqrt{3})^2 = 3$

- this is true in traditional arithmetic in algebra or calculus, but if we use calculator or computer to do, what will happen?
- In our traditional mathematical world, we permit number with an infinite number of digits;
- But in arithmetic, we define $\sqrt{3}$ as an unique positive number, so when it is multiplied by itself, we can get 3.
- In computer computation, $\sqrt{3}$ first is represented with a fixed, finite number of digits, which may be very closed to its exact value. This means only rational (有理数) number can be presented exactly.

Roundoff error(舍入误差):

- For computer storage, one standard is made by IEEE , which is Binary Floating Arithmetic Standard 754-1985:
- **Format:** single, double, or extended precision
- **64-bit(binary digit) representation for a real number:**
  - **Representation(浮点数格式):**
    由三部分组成：符号+指数+尾数
  - The first bit is a **sign** indicator, denoted s, This is followed by an 11-bit exponent(指数), c, called the **characteristic**, and a 52-bit binary fraction, f, call the **mantissa**(尾数).The base for the exponent is 2.
  - Using this system, a floating-point number can be shown with the form:

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

Example 2: consider the machine number

$$0\ 10000000011\ 101110010001\underbrace{00\cdots00}$$

- s=0
  $c = 1{\cdot}2^{10}+0{\cdot}2^{9}+\cdots+0{\cdot}2^{2}+1{\cdot}2^{1}+1{\cdot}2^{0} = 1024+2+1 = 1027$
  $f = 1{\cdot}\left(\frac{1}{2}\right)^{1}+1{\cdot}\left(\frac{1}{2}\right)^{3}+1{\cdot}\left(\frac{1}{2}\right)^{4}+1{\cdot}\left(\frac{1}{2}\right)^{5}+1{\cdot}\left(\frac{1}{2}\right)^{8}+1{\cdot}\left(\frac{1}{2}\right)^{12}$
- So the machine number precisely represents the decimal number(十进制数)

$$\begin{aligned}
& (-1)^{s}2^{c-1023}(1+f) \\
= \ & (-1)^{0}\cdot 2^{1027-1023} \\
& \times\left(1+\frac{1}{2}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}+\frac{1}{256}+\frac{1}{4096}\right) \\
= \ & 27.56640625.
\end{aligned}$$

- **underflow (下溢):** number less than $2^{-1023} \cdot (1 + 2^{-52})$; cause to zero.
- **overflow (上溢):** number greater than $2^{1024} \cdot (2 - 2^{-52})$, cause to halt.
- **Normalized decimal floating-point form:标准十进制浮点数**

$$\pm 0.d_1 d_2 \cdots d_k \times 10^n,$$

where $1 \leq d_1 \leq 9$,and $0 \leq d_i \leq 9$ for each $i = 1, 2, \cdots, k$.

- Numbers of this form are called k**-digit decimal machine numbers**— k位十进制机器数.
- The left digits $d_{k+1} d_{k+2} \cdots$ can be treated by **chopping(截断) or rounding（舍入）methods**.

Measurement of Error(误差的测度)

Definition 1.15

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

Definition 1.16

The number p^* is said to approximate p to t **significant digit**(有效位数) (or figures) if t is the largest nonnegative integer for which

$$\frac{|p - p^*|}{|p|} < 5 \times 10^{-t}.$$

- **Algorithm:** an algorithm is a procedure that describes, in an unambiguous(明确的) or clear manner, a **finite sequence of steps** to be performed in a specified order.
- **Key techniques for algorithm:** looping and condition-control method:
- **Description:** pseudo-code method.

Example 1: to compute $\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n$

Algorithm

INPUT: $N, x_1, x_2, \cdots, x_n$.
OUTPUT: SUM$=\sum_{i=1}^{n} x_i$.
Step1 Set SUM=0
Step2 For $i = 1, 2, \cdots, N$ do
    set SUM=SUM$+x_i$
Step 3 OUTPUT (SUM);
    STOP.

Some important concepts on algorithm:

- **Stable(稳定性):** An algorithm is said to be **stable** imply that small changes in the initial data can produce correspondingly small changes in final results.
- Some algorithm are stable only for certain choices of initial data, this case are called **conditionally stable** （条件稳定）.

Growth of Error 误差的增长

Definition 1.17

Suppose that E_0 denotes an initial error and E_n represents the magnitude of an 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 $C > 1$,then the growth of error is called **exponential**.

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

$$|\alpha_n - \alpha| \leq K|\beta_n|,$$

for large n

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

误差的传播

记$\hat{p}, \hat{q}$ 分别为p, q 的近似值，误差分别为$\varepsilon_p, \varepsilon_q$

- 和运算：

$$p + q = (\hat{p} + \varepsilon_p) + (\hat{q} + \varepsilon_q) = (\hat{p} + \hat{q}) + (\varepsilon_p + \varepsilon_q)$$

- 积运算：

$$pq = (\hat{p} + \varepsilon_p)(\hat{q} + \varepsilon_q) = \hat{p}\hat{q} + \hat{p}\varepsilon_q + \hat{q}\varepsilon_p + \varepsilon_p\varepsilon_q$$

- 商运算：

$$\frac{p}{q} - \frac{\hat{p}}{\hat{q}} = \frac{\hat{p} + \varepsilon_p}{\hat{q} + \varepsilon_q} - \frac{\hat{p}}{\hat{q}} = -\frac{\hat{p}\varepsilon_q + \hat{q}\varepsilon_p}{\hat{q}(\hat{q} + \varepsilon_q)}$$

Example 2:

Suppose that for $n \geq 1$,

$$\alpha_n = \frac{n+1}{n^2}, \quad \texttt{and} \quad \hat{\alpha}_n = \frac{n+3}{n^3}.$$

we can see that

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

and

$$|\hat{\alpha}_n - 0| = \frac{n+3}{n^3} \leq \frac{n+3n}{n^3} \leq 4\frac{1}{n^2}$$

so

$$\alpha_n = 0 + O(\frac{1}{n}), \quad \texttt{and} \quad \hat{\alpha}_n = 0 + O(\frac{1}{n^2}).$$

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

$$|F(h) - L| \leq K|G(h)|,$$

for sufficient small h, then we write

$$F(h) = L + O(G(h)).$$

Example 3:

By Taylor formula for sufficient small h, we have

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

since

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

so

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

Well-Posed or ill-posed Problem—适定性与不适定性问题

A **mathematical Problem** is said to be **well-posed** if a **solution**

1. exists,
2. is unique,
3. depends continuously on problem data .

Otherwise, problem is **ill-posed.**

Remarks:

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

Remarks:

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

General Strategy to solve mathematical problems

- Replace difficult problem by easier one having same or closely related solution:
  - infinite $\rightarrow$ finite
  - differential $\rightarrow$ algebraic
  - nonlinear $\rightarrow$ linear
  - complicated $\rightarrow$ simple
  - high - order $\rightarrow$ low - order
- Solution obtained may only approximate that of original problem

General Strategy to solve mathematical problems

- Replace difficult problem by easier one having same or closely related solution:
  - infinite $\longrightarrow$ finite
  - differential $\longrightarrow$ algebraic
  - nonlinear $\longrightarrow$ linear
  - complicated $\longrightarrow$ simple
  - high - order $\longrightarrow$ low - order
- Solution obtained may only approximate that of original problem

Remarks:

1. **Accuracy** of final results reflects all these.
2. The **resulting perturbations** during computation may be **amplified**(放大) by algorithm or the nature of problem.
3. The study of the effects of such approximations on the **accuracy** and **stability** of numerical algorithms is traditionally called **error analysis**.

Remarks:

1. **Accuracy** of final results reflects all these.
2. The **resulting perturbations** during computation may be **amplified**(放大) by algorithm or the nature of problem.
3. The study of the effects of such approximations on the **accuracy** and **stability** of numerical algorithms is traditionally called **error analysis**.

Remarks:

1. **Accuracy** of final results reflects all these.
2. The **resulting perturbations** during computation may be **amplified**(放大) by algorithm or the nature of problem.
3. The study of the effects of such approximations on the **accuracy** and **stability** of numerical algorithms is traditionally called **error analysis**.

Example: Approximations

Computing 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.

Example: Approximations

Computing 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.

Example: Approximations

Computing 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.

Example: Approximations

Computing 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.

Remarks:

1. True value usually unknown, so we **estimate or bound error** rather than compute it exactly
2. **Relative error often taken relative to approximate value**, rather than (unknown) true value.

Assignment: