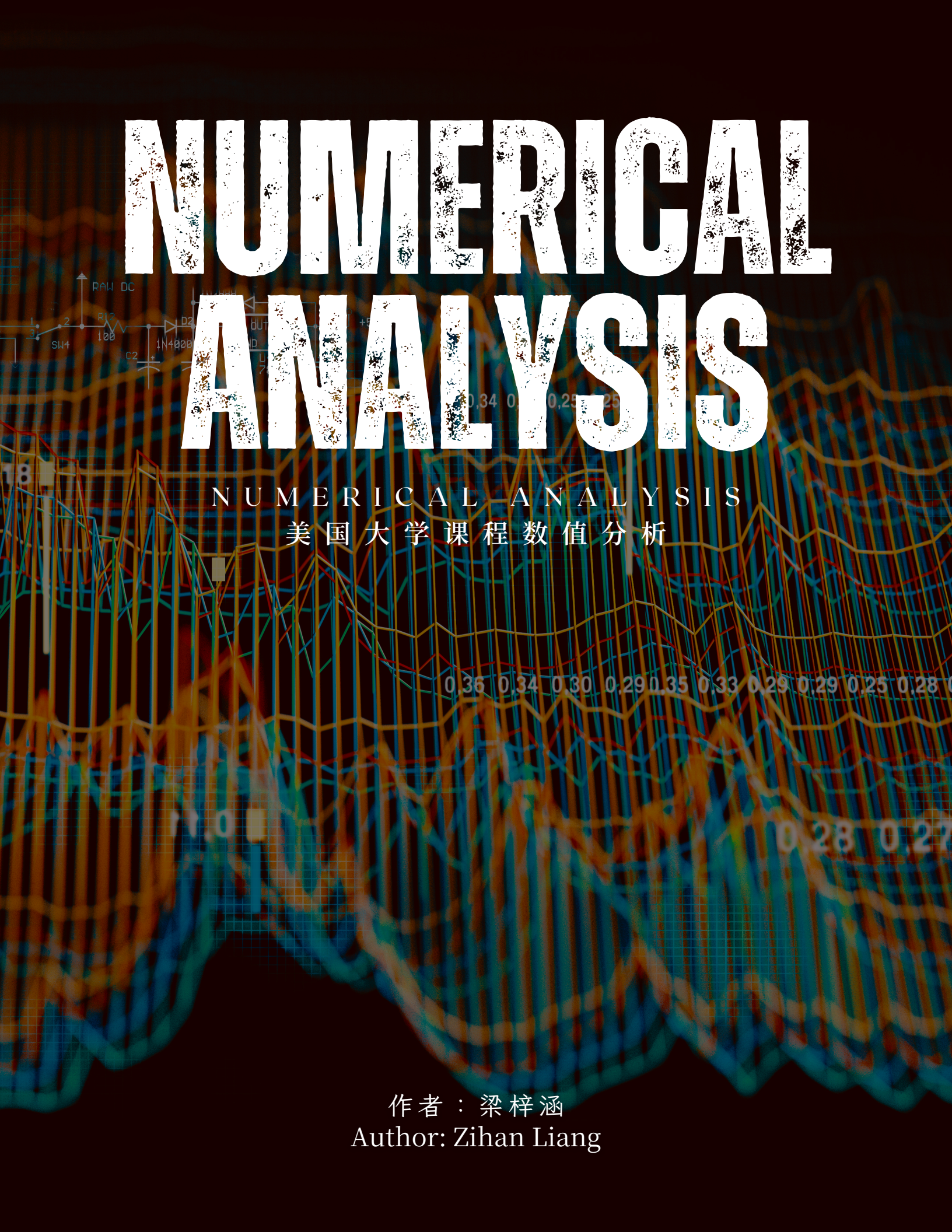
NUMERICAL ANALYSIS

NUMERICAL ANALYSIS

美国大学课程数值分析



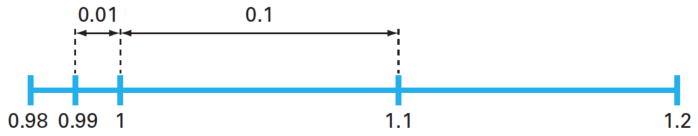
作者：梁梓涵
Author: Zihan Liang

第二单元 浮点数的运算 Chapter 2 Computing with Floating Point Numbers

十进制数和浮点数 Decimal and Floating Point Numbers

1. 科学计数法 Scientific Notation
   a. 表示 Expression
   $$x = \pm(d_1.d_2 \dots d_t d_{t+1} \dots) \times 10^e$$
   where d_i are integers, $0 \le d_i \le 9$, with $d_1 \neq 0$, and e is an integer exponent.
   b. 案例 Example
   $$x = 1.9652 \times 10^6$$
2. 电脑中表示数字的方法：浮点数 Floating Point Numbers
   a. 表示 Expression
   $$fl(x) = \pm(d_1.d_2 d_3 \dots d_t) \times 10^e$$
   b. 浮点数的一些特征 Characteristics of Floating Point Numbers
      i. Cannot represent infinite number of decimal digits. 不能表示无限的小数位数，表达式中的t就表示了对小数位数的限制。
      ii. Cannot represent infinitely large (positive or negative) exponent. 浮点数的指数e受到限制，不能任意大或者任意小。
      iii. Set of all floating points is subset of $\mathbb{R}$. 所有浮点数都属于实数集$\mathbb{R}$。
3. 十进制和其他进制 Decimal and Other Systems
   a. 十进制表示法 Base-10 (Decimal)
   $$fl(x) = \pm(d_1.d_2 d_3 \dots d_t) \times 10^e = \pm(\frac{d_1}{10^0} + \frac{d_2}{10^1} + \cdots + \frac{d_t}{10^{t-1}}) \times 10^e$$
   案例：对于一个十进制数字1.9652×10^6，
   $$fl(x) = 1.9652 \times 10^6 = (\frac{1}{10^0} + \frac{9}{10^1} + \frac{6}{10^2} + \frac{5}{10^3} + \frac{2}{10^4}) \times 10^6$$
   b. 对于其他进制的数字 General Base β
   $$fl\big((x)_\beta\big) = \pm(d_1.d_2 d_3 \dots d_t) \times \beta^e = \pm\left(\frac{d_1}{\beta^0} + \frac{d_2}{\beta^1} + \cdots + \frac{d_t}{\beta^{t-1}}\right) \times \beta^e$$
   where d_i integers, $0 \le d_i < \beta$, $d_1 \neq 0$.
   c. 二进制 Base-2
      i. 在计算机中，其使用 on/off switches，这是一个二进制系统 binary or base-2 system。
      $$fl(x)_2 = \pm(d_1.d_2 d_3 \dots d_t) \times 2^e = \pm\left(\frac{d_1}{2^0} + \frac{d_2}{2^1} + \cdots + \frac{d_t}{2^{t-1}}\right) \times 2^e$$
      其中，请注意以下细节
      - d_i integers, $0 \le d_i < 2$, i.e. $d_i = 0,1$.
      - $d_1 \neq 0$.
      - t表示有效数字位数，determined by number of bits available.

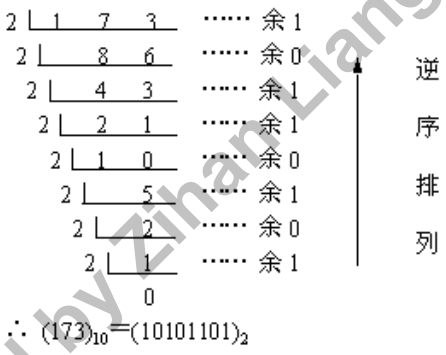
ii. t表示有效数字位数，其可能会导致四舍五入误差 roundoff error。比如对于$x = 93.75$这个数字，如果$t = 3$，则$fl(x) = 9.38 \times 10^1$，这导致了数字的不精确。



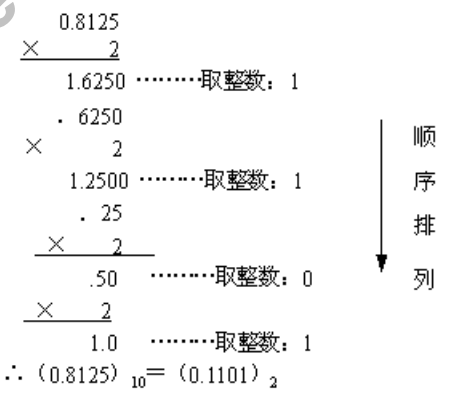
d. 十进制与二进制的转化 Decimal-to-Binary Conversion

i. Step 1: 把十进制数字分为整数部分和小数部分。对于整数部分，使用“除 2 取余，逆序排列”法；对于小数部分，使用“乘 2 取整，顺序排列”法。

ii. Step 2: 整数部分“除 2 取余，逆序排列”法。



iii. Step 3: 小数部分的“乘 2 取整，顺序排列”法。



e. 十进制与任意进制的转化 Conversion of Decimal and Arbitrary Systems

i. Step 1: 将十进制数字用十进制表示法进行表示。

$$fl(x) = \pm(d_1.d_2d_3 \dots d_t) \times 10^e = \pm(\frac{d_1}{10^0} + \frac{d_2}{10^1} + \cdots + \frac{d_t}{10^{t-1}}) \times 10^e$$

ii. Step 2: 根据原十进制数字，填写对应的进制数字。

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	…
3^0	3^{-1}	3^{-2}	3^{-3}	3^{-4}	3^{-5}	3^{-6}	3^{-7}	3^{-8}	3^{-9}	3^{-10}	…
…											

具体填写方法为: 如果对应的进制数字小于等于十进制数字的对应位数，那么进一位。比如将$1/5 = 0.2$转化为 2 进制。

0	0	0	1	1	0	0	1	1	0	0	…
2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	…

$2^0 = 1$大于0.2，因此写 0；

$2^{-1} = 0.5$大于0.2，因此写 0；

$2^{-2} = 0.25$大于0.2，因此写 0；

$2^{-3} = 0.125$小于0.2，因此写 1；

$2^{-4} = 0.0625$小于$0.2 - 0.125$，因此写 1

……

iii. 根据要求，将进制数字转化为规定的位数，并使用科学计数法表示。

4. 浮点数的表示方法 Floating-Point Representation

a. 双精度浮点数（64 位）Double-Precision Floating-Point (64 Bit)

$$(-1)^{sign} \times (1.mantissa) \times 2^{biased\ exponent\ (exponent-1023)}$$

i. 符号位 Sign Bit: 第 63 位，表示数的正负，0 表示正，1 表示负。

ii. 指数部分 Biased Exponent: 第 62 位到第 52 位，用 11 为来表示指数的大小。

iii. 尾数部分 Mantissa: 第 51 位到第 0 位，用 52 位来存储尾数的有效位，即小数部分。

b. 单精度浮点数（32 位）Single-Precision Floating-Point (32 Bit)

$$(-1)^{sign} \times (1.mantissa) \times 2^{biased\ exponent\ (exponent-127)}$$

i. 符号位 Sign Bit: 1 位。

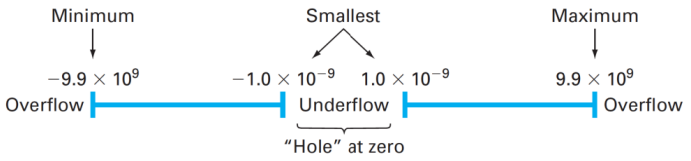
ii. 指数部分 Biased Exponent: 8 位。

iii. 尾数部分 Mantissa: 23 位。

c. 浮点数的有限表示范围 Finite Range of Floating-Point Numbers: 浮点数的表示范围是有限的，计算机无法显示所有的实数，只能表示有限范围内的数字。

i. 上溢 Overflow: 如果数值超过最大值，会溢出，直接变成无穷大。

ii. 下溢 Underflow: 当数值非常接近 0 时，会下溢为 0，导致丢失精度。



d. 浮点数间隔随数值增大而增大 Floating-Point Intervals Increase as the Value Increases

i. 如果我们使用科学计数法表示浮点数，那么浮点数可以被表示为

$$(-1)^{sign} \times (1.mantissa) \times 2^{biased\ exponent}$$

ii. 此时，当数字越大时，浮点数中的指数 Exponent 就会越大。由于科学计数法，其尾数部分 Mantissa 保持固定的尾数，指数越大，相邻的数值的间隔（差距）也就变的更大。因此，较大的数字在每次变化时，数值的变化幅度也会变大。Interval between numbers increases as numbers grow in magnitude.

e. 机器精度 Machine Epsilon ϵ_m: 最小可表示的差值，指的是计算机在浮点数表示中，1 和它能表示的下一个数之间的差值。其是浮点数计算中不可避免的误差范围。

$$\epsilon_m = \beta^{-t}$$

i. 对于双精度浮点数 Double-Precision Floating-Point，其$\epsilon_m = 2^{-52} \approx 10^{-16}$。

ii. 假设机器精度为 2^{-52}，即 $\epsilon_m \approx 10^{-16}$。如果我们试图在计算机上表示 1.000000000000001 和 1.0000000000000002，那么这两个数是可以被区分开的，因为它们的差距是 10^{-15}，大于机器精度。但如果我们试图表示 1 和 1.00000000000000001，它们之间的差值是 10^{-17}，小于机器精度，计算机会把它们看作相等。

5. 整数 Integers

a. 整数的定义 Definition of Integers 用于表示没有小数部分的数值（即整数值），可以是 signed（有符号）或 unsigned（无符号）。

b. 有符号整数 Signed Integers

i. 有符号整数 Signed Integers 可以表示正数、负数和零。在计算机中，使用二进制补码 Two's Complement 来表示负数。

ii. 位分配 Bit Distribution: 有符号整数的最高位（即最左边的一位）通常用于表示符号，其余位用于表示数值大小。对于最左边的一位：

- 0 表示正数。
- 1 表示负数。

iii. 取值范围 Range of Values: 假设是 n 位的有符号整数，则数值范围为

$$-2^{n-1}\ to\ 2^{n-1} - 1$$

例如，对于 8 位有符号整数，范围是

$$-128\ to\ 127$$

其中 -128 是用补码表示的负数，127 是最大正数。

iv. 二进制补码 Two's Complement: 负数的表示方式是通过将正数取反加一来得到补码形式。例如对于 4 位有符号整数，

- 0010_2表示+2
- 1110_2表示-2（取正数 2 的补码: 2 的二进制位0010_2，取反为1101_2，加 1 为1110_2）

c. 无符号整数 Unsigned Integers
    i. 无符号整数 Unsigned Integers 只能表示非负整数，因此不需要用位来表示符号。
    ii. 位分配 Bit Distribution: 无符号整数的所有位都用于表示数值，因此可以表示更大的正数范围。
    iii. 取值范围 Range of Values: 假设是 n 位的有符号整数，则数值范围为

$$0\ to\ 2^n - 1$$

例如，对于 8 位无符号整数，范围是

$$0\ to\ 255$$

6. 浮点运算 Floating-Point Arithmetic
    a. 实数的封闭性 Closure of Real Numbers: 所有实数在加法、减法、乘法和除法（除以零除外）运算下都是封闭的 closed under addition, subtraction, and multiplication, and division。也就是说，任何两个实数通过这些运算得到的结果仍然是实数。
    b. 双精度浮点数的不封闭性 Non-Closure of Double-Precision Floating-Point Numbers: 双精度浮点数集合在这些运算下并不总是封闭的。这意味着两个双精度浮点数的运算结果可能无法被精确地表示为另一个双精度浮点数。
    c. 浮点数运算的定义 Definition of Floating Point Arithmetic
        i. 封闭浮点数的集合 Closed Set of Floating Point Numbers: 因为浮点数运算无法保证结果总是精确地落在浮点数的集合中，所以引入了一个近似的操作方式。在计算中，对两个浮点数进行加法、减法、乘法或除法时，结果通常会被舍入到最接近的可以用双精度浮点数表示的值。
        ii. 封闭方式 Closure Method
            - Step 1: 精确的代数运算（首先，进行实际的代数运算，不考虑浮点数的表示限制。）

$$x + y \quad x - y \quad x \times y \quad x \div y$$

            - Step 2: 浮点舍入：将这个精确的代数运算结果舍入到最接近的可以用浮点数表示的数。

$$x \oplus y := fl_{DP}(x + y)$$
$$x \ominus y := fl_{DP}(x - y)$$
$$x \otimes y := fl_{DP}(x \times y)$$
$$x \oslash y := fl_{DP}(x/y)$$

            where $fl_{DP}(z)$ is the double precision number closest to the real number z.
            - Floating point arithmetic is inherently approximate.
        iii. 浮点舍入规则 Floating Point Rounding Rule: 我们首先把转换后的浮点数表示为

$$fl_{DP}(z) = z(1+\mu)$$

即，任意被转化的浮点数，可以被写为$z(1+\mu)$，其中z表示精准的代数结果，而μ则表示舍入因子 Rounding Factor。我们的舍入标准是

$$|\mu| \leq \frac{\epsilon_{DP}}{2} or\ |\mu| < \epsilon_{DP}$$

7. 误差的传播 Propagation of Errors
    a. 在电脑运算中，有两种 Errors，分别是:
        i. 数值中固有的误差 Error inherent in the numbers: 指的是由于表示数值时使用了有限精度，因此不可避免的误差。
        ii. 由算术运算引入的误差 Error introduced by the arithmetic: 这是指在进行加减乘除等运算时，由于有限精度和舍入误差而产生的额外误差。
    b. 灾难性抵消 Catastrophic Cancelation: 在计算机算术中，减去两个接近的双精度数值时可能会导致显著的精度损失。

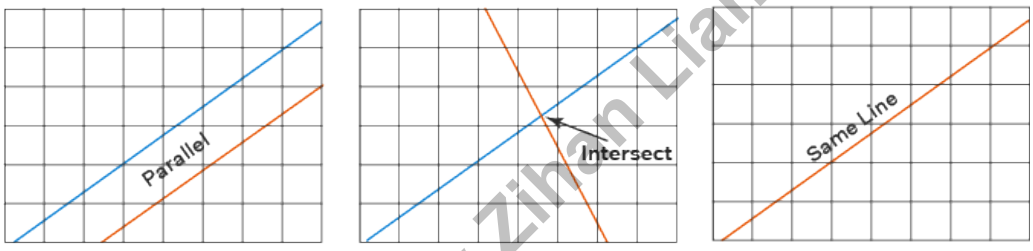
第三单元 线性系统的解 Chapter 3 Solution of Linear Systems

线性系统 Linear Systems

1. 线性方程组回顾 Recall of System of Linear Equation
   a. 线性方程组和矩阵 SLE and Matrix: Linear system of equations can be written as $Ax = b$.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

   A is the matrix of coefficients.
   x is the vector of unknowns.
   b is the vector of constants.
   b. 线性方程组的解 Solutions of SLE



      i. When two lines are parallel, the corresponding system of linear equations has no solution because the lines do not intersect.
      ii. When two lines intersect at a point, the corresponding system of linear equations has a unique solution.
      iii. When two lines coincide, the corresponding system of linear equations has infinitely many solutions and any points on this line is equal to the solution to the SLE.
   c. 线性方程组的特解 Unique Solution of SLE
      i. 对于一个线性方程组，若其存在特解 Unique Solution，则其两个线性方程的斜率 Slope 不能一致。

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$
$$\frac{a_{11}}{a_{12}} \neq \frac{a_{21}}{a_{22}}$$

      ii. 我们可以由此推出如下结论
         - Determinant of A, i.e. $a_{11}a_{22} - a_{12}a_{21}$ for 2×2 matrix, must be nonzero.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- A must be nonsingular, one and only one solution.
- Columns of A must be linearly independent. (No column is linear combination of others.)
- A must be invertible.

d. 特殊的矩阵 Special Matrices

i. 单位矩阵 Identity Matrix

$$I = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

All entries on diagonal are 1.

Any matrix multiplies an identity matrix is itself.

ii. 零矩阵 Zero Matrix

$$O = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

All entries are 0.

Any matrix multiplies a zero matrix is 0.

iii. 对角矩阵 Diagonal Matrix

$$A = \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{nn} \end{pmatrix}$$

The only non-zero entries are at diagonal and matrix is in square.

iv. 下三角矩阵 Lower Triangular Matrix

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 6 \end{pmatrix}$$

The portion above the diagonal is only 0. We can use forward substitution to get the solution of $Ax = b$, where A is a lower triangular matrix.

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

正向代入法 Forward Substitution:

$$1x_1 = 1 \quad x_1 = 1$$

我们将x_1的值带入$2x_1 + 4x_2 = 2$来得到x_2，并继续得到x_3。

v. 上三角矩阵 Upper Triangular Matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix}$$

The portion below the diagonal is only 0. We can use backward substitution to get the solution of $Ax = b$, where A is a upper triangular matrix.

$$\begin{pmatrix}1 & 2 & 3\\ 0 & 4 & 5\\ 0 & 0 & 6\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}=\begin{pmatrix}1\\ 2\\ 3\end{pmatrix}$$

逆向代入法 Backward Substitution:

$$6x_3 = 3 \;\; x_3 = 0.5$$

我们将x_3的值带入$4x_2 + 5x_3 = 2$来得到x_2，并继续得到x_1。

2. 解线性方程组 Simply Solved Linear Systems
   a. 逆向代入法算法 Backward Substitution Algorithms
      i. 假设我们已经得到一个 REF 矩阵（上三角矩阵 Upper Triangular Matrix），我们可以使用 MATLAB 中的逆向代入法算法 Backward Substitution Algorithms 来求解出结果。
      ii. 下图是列方向算法 Row-Oriented Algorithm 和行方向算法 Column-Oriented Algorithm 的 MATLAB 伪代码。

Row–Oriented	*Column–Oriented*

Input: matrix $A = [a_{i,j}]$
vector $b = [b_i]$
Output: solution vector $x = [x_i]$
 	
for $i = n$ downto 1 do
  for $j = i + 1$ to n
  $b_i := b_i - a_{i,j}x_j$
  next j
  $x_i := b_i/a_{i,i}$
next i
	Input: matrix $A = [a_{i,j}]$
	vector $b = [b_j]$
	Output: solution vector $x = [x_j]$
	
	for $j = n$ downto 1 do
	  $x_j := b_j/a_{j,j}$
	  for $i = 1$ to $j - 1$
	  $b_i := b_i - a_{i,j}x_j$
	  next i
	next j

      iii. 列方向算法 Row-Oriented Algorithm: 从矩阵的最后一列开始逐列进行处理，每一列逐项计算对应的解。
         - Step 1: 从最后一列开始，计算x_n。比如，

$$\begin{pmatrix}4 & 2\\ 0 & 3\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}-2\\ 3\end{pmatrix}$$

           在这个矩阵计算中，我们通过第二列可得$a_{2,2} = 3$，而对应的$b_2 = 3$。

$$x_2 = \frac{b_2}{a_{2,2}} = \frac{3}{3} = 1$$

         - Step 2: 逐步向前，直到第一列。对于每一列需要针对后面的b进行更新。

$$b_i = b_i - a_{i,j} \times x_j$$

           比如，现在回到第一列，处理第一行$a_{1,1} = 4$，而对应的$b_1 = -2$。更新结果如下

$$b_1 = b_1 - a_{1,2} \times x_2 = -4$$

- Step 3: 列出每一列的式子，计算剩余x_i。比如，

$$x_1 = \frac{b_1}{a_{1,1}} = \frac{-4}{4} = -1$$

iv. 行方向算法 Column-Oriented Algorithm: 从矩阵的最后一行开始向上逐行处理，每一行使用前面已经解出的解来计算当前未知行的未知数。

- Step 1: 从最后一行开始，计算x_n。比如，

$$3x_2 = 3 \quad x_2 = \frac{3}{3} = 1$$

- Step 2: 回到上一行，利用已知的x_n，计算x_{n-1}。比如，

$$4x_1 + 2x_2 = -2$$
$$x_1 = -1$$

b. 逆向代入法成本分析 Backward Substitution Cost: 给定$n \times n$矩阵 A，我们通过如下公式计算乘法、减法和除法的成本

i. 乘法次数 Multiplications

$$\frac{n(n-1)}{2}$$

ii. 减法次数 Subtractions

$$\frac{n(n-1)}{2}$$

iii. 除法次数 Divisions

$$n$$

3. 矢量的范数 Vector Norms

a. 矢量范数的性质 Properties of A Norm of v, i.e. $||v||$

i. 非负性 Non-Negativity: $||v|| \geq 0$ for all vectors $v \in \mathbb{R}^n$, and : $||v|| = 0$ if and only if $v = 0$.

ii. 三角不等式 Triangle Inequality: $||v + w|| \leq ||v|| + ||w||$ for all vectors $v \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$.

iii. 齐次性 Homogeneity: $||cv|| = |c|||v||$ for all vectors $v \in \mathbb{R}^n$ and all scalars c.

b. 矢量的三个范数 Three Vector Norms

i. 曼哈顿距离 Manhattan Distance ℓ_1

$$||x||_1 = |x_1| + |x_2| + \cdots + |x_n| = \sum_{i=1}^{n} |x_i|$$

ii. 欧几里得范数 Euclidean Norm ℓ_2

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2} = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$$

iii. 无穷范数 Infinite Norm ℓ_∞: 向量中各个元素绝对值的最大值。

$$||x||_\infty = \max_{1\le i\le n}\{|x_i|\}$$

4. 矩阵的范数 Matrix Norms
   a. 矩阵范数的性质 Properties of A, i.e. $||A||$
      i. 非负性 Non-Negativity: $||A|| \ge 0$ for all matrices $A \in \mathbb{R}^{m\times n}$, and : $||A|| = 0$ if and only if $A = 0$.
      ii. 三角不等式 Triangle Inequality: $||A + B|| \le ||A|| + ||B||$ for all vectors $A, B \in \mathbb{R}^{m\times n}$.
      iii. 齐次性 Homogeneity: $||cA|| = |c|||A||$ for all vectors $A \in \mathbb{R}^{m\times n}$ and all scalars c.
      iv. 其他性质 Other Properties:
         - 矩阵范数与向量范数之间的关系 Relationship between Matrix and Vector Norms

$$||Ax|| \le ||A|| \cdot ||x||$$

         - 两个矩阵相乘的范数 The Norm of the Multiplication of Two Matrices

$$||A \cdot B|| \le ||A|| \cdot ||B||$$

   b. F 范数 Frobenius Norm: 矩阵中所有元素的平方和的平方根。

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

   c. 诱导范数 Induced Norms
      i. $p = 1$，最大列和范数，定义为矩阵所有列中元素绝对值和的最大值。

$$||A||_1 = \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|$$

      ii. $p = \infty$，最大行和范数，定义为矩阵所有行中元素绝对值和的最大值。

$$||A||_\infty = \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|$$

      iii. $p = 2$，谱范数，定义为 A 矩阵 SVD 分解后的最大奇异值σ_1。

$$||A||_2 = \sigma_1$$

   d. 条件数 Condition Number
      i. 条件数的定义 Definition of Condition Number

$$\kappa(A) = ||A|| \cdot ||A^{-1}||$$

      对于 2-范数，其可以表示为

$$\kappa(A) = \left|\left|A\right|\right|_2 \cdot \left|\left|A^{-1}\right|\right|_2 = \frac{\sigma_1}{\sigma_n}$$

其中σ_1是 A 矩阵的最大奇异值，σ_n是矩阵的最小奇异值。

ii. 条件数的含义 Meaning of Condition Number: $\kappa(A)$帮助我们判断矩阵在求解 Linear System 时的灵敏度。

$$\frac{\left|\left|x - \hat{x}\right|\right|}{\left|\left|x\right|\right|} \leq \kappa(A) \cdot \frac{\left|\left|r\right|\right|}{\left|\left|b\right|\right|}$$

如上公式说明:

- The larger the condition number, the more the error is enlarged.
- Even if the residuals are small, the error can be large if the condition number of the matrix is large.

iii. 良态矩阵和病态矩阵 Well-Conditioned and Ill-Conditioned Matrix

- If A is well-conditioned, i.e. $\kappa(A) \approx 1$, small r implies accurate $\hat{x}$.
- If A is ill-conditioned, i.e. $\kappa(A)$ is large, A is nearly singular and small r does not imply accurate $\hat{x}$.

5. LU 分解 LU Factorization

a. 定义 Definition: LU 分解将一个方阵 A 分解为两个矩阵的乘积

$$A = LU$$

其中 L 是一个下三角矩阵 Lower Triangular Matrix，其对角线元素均为 1。U 是一个上三角矩阵 Upper Triangular Matrix。

b. 步骤 Steps

i. Step 1: 运用高斯消元法 Gaussian Elimination 将矩阵化简成阶梯形矩阵 Row-Echelon Form (REF)。请注意，在化简过程中不能“交换行”。原矩阵的 REF 形式就是其 U 矩阵（上三角矩阵 Upper Triangular Matrix）。

ii. Step 2: 在上述把原矩阵化简成 REF 矩阵的过程中，记录所有的初等行变换 Elementary Row Operations (EROs)步骤。并把每一步的 ERO 分别单独应用在一个 Identity Matrix 上，形成 Elementary Matrix。有几步就有几个 Elementary Matrices，分别是E_1, E_2 ...。

iii. Step 3: 将所有 Elementary Matrices 变成其逆矩阵 Inverse Matrices。只需要把除了 Identity Matrix 之外的对应元素变成其相反数即可。

iv. Step 4: 通过如下公式求解 L 矩阵（下三角矩阵 Lower Triangular Matrix）。

$$E_1^{-1}E_2^{-1}E_3^{-1}E_4^{-1} \ldots E_k^{-1} = L$$

c. 另一种方法计算 L 矩阵 Another Method to Compute the L Matrix

i. Step 1: 对于 A 矩阵到 U 矩阵，我们只使用唯一一个 ERO 进行，即“Replacing R_i by $R_i - kR_j$”且保证$i > j$。

ii. Step 2: 根据我们每一步“Replacing R_i by $R_i - kR_j$”中的 k 值来得到 L 矩阵。

iii. 案例 Example:

$$\begin{bmatrix} 2 & 2 & 3 \\ 5 & 9 & 10 \\ 4 & 1 & 2 \end{bmatrix}$$

We use $R_2 = R_2 - \frac{5}{2}R_1$ $(k = \frac{5}{2})$

$$\begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & \frac{5}{2} \\ 4 & 1 & 2 \end{bmatrix}$$

We use $R_3 = R_3 - 2R_1$ $(k = 2)$

$$\begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & \frac{5}{2} \\ 0 & -3 & -4 \end{bmatrix}$$

We use $R_3 = R_3 + \frac{3}{4}R_2$ $(k = -\frac{3}{4})$

$$\begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & \frac{5}{2} \\ 0 & 0 & -\frac{17}{8} \end{bmatrix} = U$$

At this point, k=5/2 is used to form $a_{2,1}$, k=2 is used to form $a_{3,1}$ and k=-3/4 is used to form $a_{3,2}$.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{2} & 1 & 0 \\ 2 & -\frac{3}{4} & 1 \end{bmatrix}$$

d. 根据 LU 分解求解线性方程组的解 Solving the System of Linear Equations Based on LU Decomposition

$$Ax = b \quad LUx = b$$

Let $Ux = y$

$$Ly = b$$

We use forward substitution to solve y.

$$Ux = y$$

We use backward substitution to solve x.

6. 部分主元高斯消元法和 PA=LU 分解 Gaussian Elimination with Partial Pivoting and PA = LU Factorization

a. 定义 Definition: 在进行高斯消元时，如果进行了行交换操作，我们可以将原来的矩阵 A 转化为一个置换矩阵 Permutation Matrix 和矩阵的 LU 分解。

$$PA = LU$$

其中 P 是置换矩阵 Permutation Matrix，它记录了行交换的操作。L 是一个下三角矩阵 Lower Triangular Matrix，其对角线元素均为 1。U 是一个上三角矩阵 Upper Triangular Matrix。

b. 为什么要使用部分主元 Why We Using Partial Pivoting?
   i. **Avoid Denominators of 0.** If the main diagonal element of the current column is 0, then eliminating it directly will result in a divide by 0 error.
   ii. **Avoid Too Small Divisor.** During Gaussian elimination, if the element on the main diagonal is very small, using it directly for the elimination operation may lead to unstable values or even overflow during the calculation.
   iii. **Improve Numerical Accuracy**. Partial Pivoting can reduce the rounding error introduced during Gaussian elimination and thus improve the accuracy of the solution.
   iv. **Prevent Matrix Ill-Conditioning.** For some ill-conditioning matrices, direct application of the Gaussian elimination may lead to highly inaccurate solution results. Partial Pivoting can improve the situation.
c. 步骤 Steps
   i. Step 1: 在第一列中，需要确保绝对值最大的元素在最上面（运用行变换），作为主元 Pivot。将行变换也应用在 Identity Matirx 形成置换矩阵 Permutation Matrix 1。
   ii. Step 2: 运用高斯消元法 Gaussian Elimination 将第一列除了主元 Pivot 之外的元素变成 0。
   iii. Step 3: 我们持续关注第二列。依旧需要保证绝对值最大的元素作为主元 Pivot（运用行变换）。将行变换也应用在 Identity Matirx 形成置换矩阵 Permutation Matrix 2。
   iv. Step 4: 运用高斯消元法 Gaussian Elimination 将第二列除了主元 Pivot 之外的元素变成 0。
   v. Step 5: 以此类推，我们得到了 PA 的 U 矩阵（上三角矩阵 Upper Triangular Matrix）。
   vi. Step 6: 上述高斯消元法 Gaussian Elimination 过程中，我们依旧会得到的 Elements Matrices 和 Permutation Matrices。我们需要根据 Elements Matrices 和 Permutation Matrices 出现的顺序来计算 L 矩阵。

$$P = P_2 P_1$$

$$L = P(\text{第一个出现的 E/P 矩阵})^{-1}(\text{第二个出现的 E/P 矩阵})^{-1} \ldots$$

d. 根据 PA-LU 分解求解线性方程组的解 Solving the System of Linear Equations Based on PA-LU Decomposition

$$P(Ax = b)$$

Let $PAx = Pb = d$

Now, we have

$$PAx = d \quad LUx = d$$

Let $Ux = y$

$$Ly = d$$

Slove y.

$$Ux = y$$

Slove x.

7. 科列斯基分解 Cholesky Factorization
   a. 定义 Definition: 一个$n \times n$的矩阵 A 是对称正定的 Symmetric Positive Definite，当且仅当其可以分解为
   $$A = R^T R$$
   其中，R 是一个上三角矩阵 Upper Triangular Matrix，并且其对角线元素为正数。
   b. 对称正定矩阵 Symmetric Positive Definite (SPD) Matrix
      i. 对称性 Symmetric
      $$A = A^T$$
      ii. 正定性 Positive Definite
         - 方法一: All its eigenvalues λ is positive.
         - 方法二:
         $$x^T Ax \geq 0 \quad \forall x \in \mathbb{R}^n$$
         $$x^T Ax = 0 \iff x = 0$$
   c. 科列斯基分解步骤 Steps of Cholesky Factorization
      i. Step 1: 确定矩阵 A 是对称正定的 Symmetric Positive Definite。
      ii. Step 2: 确定 R 即上三角矩阵的形式
      $$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} \quad R^T = \begin{bmatrix} r_{11} & 0 & 0 \\ r_{12} & r_{22} & 0 \\ r_{13} & r_{23} & r_{33} \end{bmatrix}$$
      iii. Step 3: 写出$R^T R$
      $$R^T R = \begin{bmatrix} r_{11}^2 & r_{11}r_{12} & r_{11}r_{13} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 & r_{12}r_{13} + r_{22}r_{23} \\ r_{11}r_{13} & r_{12}r_{13} + r_{22}r_{23} & r_{13}^2 + r_{23}^2 + r_{33}^2 \end{bmatrix}$$
      iv. Step 4: 与矩阵 A 对应项对比，得到方程组，逐步解出r_{ij}，在解方程组的过程中，若有两根，取正根。
   d. 根据 Cholesky 分解求解线性方程组的解 Solving the System of Linear Equations Based on Cholesky Factorization
   $$Ax = b \quad R^T Rx = b$$

Let $Rx = y$

$$R^T y = b$$

Slove y by forward substitution.

$$Rx = y$$

Slove x by backward substitution.

8. QR 分解 QR Factorization
   a. 定义 Definition: 对于一个$m \times n$的矩阵 A，可以将它分解为两个矩阵 Q 和 R

   $$A = QR$$

   Q 是一个正交矩阵 Orthogonal Matrix；R 是一个上三角矩阵 Upper Triangular Matrix。
   b. 正交矩阵 Orthogonal Matrix

   $$Q = [q_1\ q_2 \dots q_n]$$
   $$q_i^T q_j = \begin{cases} 1, \text{if i} = \text{j} \\ 0, \text{if i} \neq \text{j} \end{cases}$$
   $$Q^T Q = I$$

   c. 根据 QR 分解求解线性方程组的解 Solving the System of Linear Equations Based on QR Factorization

   $$Ax = b \quad QRx = b$$

   Let $Rx = y$

   $$Qy = b \quad y = Q^T b$$

   Slove y.

   $$Rx = y$$

   Slove x.
9. 奇异值分解 Singular Value Decomposition
   a. 定义 Definition: 对于一个任意的$m \times n$矩阵 A，奇异值分解 SVD 将其分解为三个矩阵的乘积

   $$A = U\Sigma V^T$$

   U 是$m \times m$正交矩阵 Orthogonal Matrix，V 是$n \times n$正交矩阵 Orthogonal Matrix。Σ是$m \times n$的对角矩阵 Diagonal Matrix，其中对角线上的元素为 A 的奇异值 Singular Values。

   $$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

   其中$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$为矩阵的 A 非零奇异值 Singular Values，r 是矩阵 A 的 Rank。
   b. 根据 SVD 分解求解线性方程组的解 Solving the System of Linear Equations Based on SVD Factorization

   $$Ax = b \quad A = U\Sigma V^T$$

We have

$$U\Sigma V^T x = b$$
$$U^T(U\Sigma V^T x = b)$$
$$U^T U\Sigma V^T x = U^T b$$
$$\Sigma V^T x = U^T b = d$$

Let $V^T x = y$

$$\Sigma y = d$$

Slove y.

$$V^T x = y$$

Slove x.

第四单元 曲线拟合 Chapter 4 Curve Fitting

多项式插值 Polynomial Interpolation

1. 定义 Definition: 多项式插值 Polynomial Interpolation 的目标就是精确地拟合所有给定的数据点。我们希望找到一个多项式$p(x)$，使得该多项式在每个给定的数据点(x_i, f_i)处满足$p(x_i) = f_i$。
    a. 多项式的阶数和多项式插值 Order of Polynomials and Polynomial Interpolation: 通过提高多项式的阶数（例如，n阶多项式能够通过$n+1$个点），我们可以确保这个多项式能够精确地经过所有的数据点。
    b. 多项式插值与最小二乘法 Polynomial Interpolation and Least Squares: 这与一般的拟合（比如最小二乘法）不同，拟合方法可以近似数据点，但并不需要完全通过所有点，而多项式插值的要求则是精确地通过所有点。
2. 多项式插值的唯一性定理 Polynomial Interpolation Uniqueness Theorem
    a. 定理内容 Theorem Content: When the nodes $\{x_i\}_{i=0}^N$ are distinct, there is a unique polynomial, the interpolating polynomial $p_N(x)$, of degree N that interpolates to the data $\{(x_i, f_i)\}_{i=0}^N$.
    b. 定理含义 Theorem Implication: 在节点不重合的情况下，能够找到且只有一个多项式可以通过所有的数据点，这保证了插值结果的确定性。即，当有 N+1 个不同的数据点时，存在唯一一个 N 次多项式能够完全插值这些点。

幂函数多项式插值 Power Function Polynomial Interpolation

1. 幂函数多项式插值条件 Power Function Polynomial Interpolation Conditions
    a. Condition 1: 每个插值点(x_i, f_i)必须满足给定的多项式形式。这意味着插值多项式$p_M(x)$在每个x_i处的值都必须等于f_i。
    b. Condition 2: 插值多项式的表达形式为
    $$p_M(x) = a_0 + a_1 x + \cdots + a_M x^M$$
    这是一个 M 次多项式的形式，其中$a_0, a_1, \ldots, a_M$是多项式的系数。
2. 幂函数插值方程系统 Power Function System of Interpolated Equations: 对于每个数据点$(x_0, f_0), \ldots, (x_N, f_N)$，插值多项式需要满足以下方程：
$$p_M(x_0) = a_0 + a_1 x_0 + \cdots + a_M x_0^M = f_0$$
$$p_M(x_1) = a_0 + a_1 x_1 + \cdots + a_M x_1^M = f_1$$
$$\ldots$$
$$p_M(x_N) = a_0 + a_1 x_N + \cdots + a_M x_N^M = f_N$$
这些方程表达了多项式在每个给定的x_i处必须通过对应的f_i。
3. 如何进行幂函数的多项式插值 How to Do Power Function Polynomial Interpolation

a. Step 1: 根据题目内容寻找插值点。根据插值点的个数判断插值多项式的基本形式（多项式的阶数）。插值点的个数−1等于多项式的阶数。
b. Step 2: 根据插值条件$p(x_i) = f_i$，写出插值系统方程。
c. Step 3: 我们可以通过矩阵来解如上的插值系统方程，即一个线性方程组。插值系统方程可以写成矩阵形式，即

$$p_M(x_0) = a_0 + a_1x_0 + \cdots + a_Mx_0^M = f_0$$
$$p_M(x_1) = a_0 + a_1x_1 + \cdots + a_Mx_1^M = f_1$$
$$\cdots$$
$$p_M(x_N) = a_0 + a_1x_N + \cdots + a_Mx_N^M = f_N$$

可以被转化为

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^M \\ 1 & x_1 & x_1^2 & \dots & x_1^M \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{bmatrix}$$

其中，左侧的矩阵叫做 Vandermonde Mateix，记作 V。

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^M \\ 1 & x_1 & x_1^2 & \dots & x_1^M \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^M \end{bmatrix}$$

d. Step 4: 通过解线性方程组的方法找出系数$a_0, a_1, \dots a_N$，并写出最后得到的多项式。

4. 案例 Example: 假设有三个点$(1,2), (2,3), (3,5)$。请找到一个多项式插值这三个点。
   a. Step 1: Nodes are

   $$(x_0, f_0) = (1,2), (x_1, f_1) = (2,3), (x_2, f_2) = (3,5).$$

   We need a quadratic polynomial to interpolate these three points.

   $$p(x) = a_0 + a_1x + a_2x^2$$

   b. Step 2: Now, we can create the System of Interpolated Equations.
   When $x_0 = 1, p(1) = 2$

   $$a_0 + a_1(1) + a_2(1)^2 = 2$$
   $$a_0 + a_1 + a_2 = 2$$

   When $x_1 = 2, p(2) = 3$

   $$a_0 + a_1(2) + a_2(2)^2 = 3$$
   $$a_0 + 2a_1 + 4a_2 = 3$$

   When $x_2 = 3, p(3) = 5$

   $$a_0 + a_1(3) + a_2(3)^2 = 5$$
   $$a_0 + 3a_1 + 9a_2 = 5$$

   c. Step 3: Now, we can create the SLE that contains Vandermonde Matrix.

$$\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 4\\ 1 & 3 & 9\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ a_2\end{bmatrix}=\begin{bmatrix}2\\ 3\\ 5\end{bmatrix}$$

d. Step 4: Via solving this SLE, we can get

$$a_0=2, a_1=-\frac{1}{2}, a_2=\frac{1}{2} \qquad p(x)=2-\frac{1}{2}x+\frac{1}{2}x^2$$

5. 多项式插值唯一性的证明 Proof of the Polynomial Interpolation Uniqueness: 证明多项式插值的唯一性，需证明当$x_0, x_1, \dots, x_N$互不相同时，Vandermonde 矩阵是可逆的，即证明行列式

$$\det(V)\neq 0$$

Vandermonde 矩阵的行列式可以表示为

$$\det(V)=\prod_{0\le i<j\le N}(x_j-x_i)$$

这个行列式是各插值点$x_0, x_1, \dots, x_N$的差乘积。

如果这些点x_i都是互不相同的，那么$x_j-x_i\neq 0$对每一对i和j都成立，因此$\det(V)\neq 0$。这意味着 Vandermonde 矩阵是 Non-Singular，所以线性方程组有唯一解。

插值多项式的牛顿形式与拉格朗日形式 Newton Form and Lagrange Form of The Interpolating Polynomial

1. 牛顿形式 Newton Form
   a. 牛顿插值多项式 Newton Form of the Interpolating Polynomial $p_N(x)$可以写成如下形式

$$p_N(x)=b_0+b_1(x-x_0)+b_2(x-x_0)(x-x_1)+\cdots \\ +b_N(x-x_0)(x-x_1)\dots(x-x_{N-1})$$

这里的$b_0, b_1, \dots, b_N$是我们需要确定的系数。

牛顿多项式 Newton Form 的优点是可以在每次添加新数据点时，只需要更新一项，而不需要重新计算整个多项式。

   b. 插值条件 Interpolation Conditions: 如何确定系数b_i
      i. Condition: 多项式在每个插值点x_i上都要等于对应的函数值f_i。
      ii. 当$x=x_0$时，$p_N(x_0)=b_0=f_0$，即系数b_0就是第一个点的函数值。
      iii. 当$x=x_1$时，带入牛顿多项式的表达式

$$p_N(x_1)=b_0+b_1(x_1-x_0)$$

通过该方程求解b_1。

      iv. 以此类推，求得剩下的系数。

c. 算法复杂度 Algorithmic Complexity: 求解这些系数所需的计算量（flops，即浮点运算次数）随着多项式的次数 N 的增加以N^2的速度增长。这意味着当多项式的次数较大时，计算复杂度也会相应增加。

2. 牛顿多项式求解案例 Example of Finding Newton Form of the Interpolating Polynomial: 使用(2,88), (8,81), (4,88), (5,88), (6,87)这五个数据来寻求牛顿多项式。

a. Step 1: 确定b_0

$$(x_0, f_0) = (2,88) \quad b_0 = f_0 = 88 \quad p(x) = 88$$

b. Step 2: 带入第二个点$(x_1, f_1) = (8,81)$，根据插值条件，多项式在$x = 8$处的数值是 81。

$$p(8) = b_0 + b_1(8-2)$$

已知$b_0 = 88$和$p(8) = 81$带入方程

$$81 = 88 + b_1(8-2) \quad b_1 = -\frac{7}{6}$$

c. Step 3: 带入第三个点$(x_2, f_2) = (4,88)$，根据插值条件，多项式在$x = 4$处的数值是 88。

$$p(4) = b_0 + b_1(4-2) + b_2(4-2)(4-8)$$

$$88 = 88 - \frac{7}{6}(4-2) + b_2(4-2)(4-8)$$

$$b_2 = -\frac{7}{24}$$

d. Step 4: 带入第四个点$(x_3, f_3) = (5,88)$，根据插值条件，多项式在$x = 5$处的数值是 88。

$$p(5) = b_0 + b_1(5-2) + b_2(5-2)(5-8) + b_3(5-2)(5-8)(5-4)$$

$$88 = 88 - \frac{7}{6}(5-2) - \frac{7}{24}(5-2)(5-8) + b_3(5-2)(5-8)(5-4)$$

$$b_3 = -\frac{7}{72}$$

e. Step 5: 带入第五个点$(x_4, f_4) = (6,87)$，根据插值条件，多项式在$x = 6$处的数值是 87。

$$p(6) = b_0 + b_1(6-2) + b_2(6-2)(6-8) + b_3(6-2)(6-8)(6-4) + b_4(6-2)(6-8)(6-4)(6-5)$$

$$b_4 = \frac{23}{144}$$

f. Step 6: 因此牛顿插值多项式的最终表达式为

$$p(x) = 88 - \frac{7}{6}(x-2) - \frac{7}{24}(x-2)(x-8) - \frac{7}{72}(x-2)(x-8)(x-4) + \frac{23}{144}(x-2)(x-8)(x-4)(x-5)$$

3. 拉格朗日形式 Lagrange Form

a. 拉格朗日插值多项式 Lagrange Form $p_N(x)$ 的表达式为

$$p_N(x) = f_0\ell_0(x) + f_1\ell_1(x) + \cdots + f_N\ell_N(x)$$

其中，$f_0, f_1, \ldots, f_N$是插值点的函数值；$\ell_0(x), \ell_1(x), \ldots, \ell_N(x)$是拉格朗日基函数 Lagrange Basis Function。

b. 拉格朗日基函数 Lagrange Basis Function

$$\ell_k(x) = \prod_{j=0, j\neq k}^{N} \frac{(x - x_j)}{(x_k - x_j)}$$

i. 案例 Example: 对于给定的五个插值点(2,88),(8,81),(4,88),(5,88),(6,87)，如何构造基函数$\ell_k(x)$。

- Step 1: 由于拉格朗日插值多项式从$f_0\ell_0(x)$开始，则我们需要计算到$f_4\ell_4(x)$。因此，我们需要计算四个基函数，分别是$\ell_0(x), \ell_1(x), \ell_2(x), \ell_3(x), \ell_4(x)$。
- Step 2: 遵从口诀“上未知、下实际、分别减、乘起来”。比如对于$\ell_0(x)$，实际点为$x_0 = 2$。

$$\ell_0(x) = \frac{x-8}{2-8}\cdot\frac{x-4}{2-4}\cdot\frac{x-5}{2-5}\cdot\frac{x-6}{2-6}$$

即可求出$\ell_0(x)$，根据该方法求出每一个基函数。

- Step 3: 根据拉格朗日插值公式，写出多项式$p(x)$。

ii. 拉格朗日基函数的性质 Properties of Lagrange Basis Function

- 在给定插值点 Nodes x_i上，拉格朗日基函数满足

$$\ell_i(x = x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

- 该性质保证插值多项式$p(x)$在$x = x_i$处，正好取数据点f_i的值，而并不会收到其他数据点的影响。

切比雪夫多项式 Chebyshev Polynomial

1. 切比雪夫多项式的形式 Form of Chebyshev Polynomial

a. 多项式插值的通式 General Form of Polynomial Interpolation

$$p_N(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x) = \sum_{i=0}^{n} a_i\phi_i(x)$$

其中a_i是待确定的系数；$\phi_i(x)$是多项式的基函数 Basis Function。而在这里，我们使用切比雪夫多项式的基函数$T_j(x)$在该通式中。

b. 切比雪夫多项式的基函数 Basis Function of Chebyshev Polynomial

$$T_j(x) = \cos(j \arccos(x)), j = 0,1,2, \ldots$$

i. 前几个切比雪夫多项式基函数的具体形式为

$$T_0(x) = \cos(0 \cdot \arccos(x)) = 1$$

$$T_1(x) = \cos(1 \cdot \arccos(x)) = x$$
$$T_2(x) = \cos(2 \cdot \arccos(x)) = 2x^2 - 1$$

ii. 切比雪夫基函数的递归定义 Recursive Definition: 使用前一个多项式和该多项式求解后一个多项式

$$T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x), j = 1,2, \dots$$

iii. 切比雪夫多项式基函数的定义域 Domain: 对于任意j的 Chebyshev Polynomial Basis Function，其数据的x值必须位于区间$[-1,1]$。如果数据位于其他区间，如$[a, b]$，可以通过缩放将数据映射到$[-1,1]$。

c. 切比雪夫多项式求解案例 Example of Finding Chebyshev Polynomial: 使用$(-1,-1), (0,-2), (1,3)$这三个数据来寻求切比雪夫多项式。

i. Step 1: 对于本题目，$x_0 = -1, x_1 = 0, x_2 = 1$都在区间$[-1,1]$，因此不需要映射。

ii. Step 2: 使用前三个 Chebyshev Polynomial Basis Function，即

$$T_0(x) = \cos(0 \cdot \arccos(x)) = 1$$
$$T_1(x) = \cos(1 \cdot \arccos(x)) = x$$
$$T_2(x) = \cos(2 \cdot \arccos(x)) = 2x^2 - 1$$

iii. Step 3: 写出$p(x)$。

$$p(x) = a_0 + a_1 x + a_2(2x^2 - 1)$$

我们需要求解的是a_0, a_1, a_2。

iv. Step 4: 将数据点带入$p(x)$。

当$x = -1$时，

$$-1 = a_0 + a_1(-1) + a_2(2(-1)^2 - 1)$$
$$a_0 - a_1 + a_2 = -1$$

当$x = 0$时，

$$-2 = a_0 + a_1(0) + a_2(2(0)^2 - 1)$$
$$a_0 - a_2 = -2$$

当$x = 1$时，

$$3 = a_0 + a_1(1) + a_2(2(1)^2 - 1)$$
$$a_0 + a_1 + a_2 = 3$$

v. Step 5: 通过线性方程组解出a_0, a_1, a_2。

$$\begin{cases} a_0 - a_1 + a_2 = -1 \\ a_0 - a_2 = -2 \\ a_0 + a_1 + a_2 = 3 \end{cases}$$

2. 切比雪夫点 Chebyshev Points

a. 定义 Definition: 切比雪夫点 Chebyshev Points 就是切比雪夫多项式的零点 Chebyshev Polynomial Zeros x_i，其可以使用如下公式表示

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), i = 1,2,3, \dots, n$$

这里，n是题目要求的点数，i表示第i个零点的编号。 这些零点位于区间$[-1,1]$内，并且均匀分布在$\cos(\theta)$的角度范围内。

b. 切比雪夫点的运用 Application of Chebyshev Points: Polynomial interpolation using Chebyshev points **reduces** the interpolation error. This is mainly due to the fact that the distribution of Chebyshev points **makes the behavior of the interpolating polynomial more stable** over the entire interval. This **avoids numerical instability** that may occur during the interpolation process.

多项式插值的误差 The Error in Polynomial Interpolation

1. 插值误差公式 Interpolation Error Formula

$$f(x) - p_N(x) = \frac{\omega_{N+1}(x)}{(N+1)!} f^{(N+1)}(\xi_x)$$

a. 这个公式描述了多项式插值$p_N(x)$和实际函数$f(x)$之间的误差，其表明在某个点x处，插值多项式和真实函数直接的差异。

b. $\omega_{N+1}(x)$: 这是一个与插值点 Nodes 相关的函数

$$\omega_{N+1}(x) = (x - x_0)(x - x_1) \dots (x - x_N)$$

该函数表示x与插值点$x_0, x_1, \dots, x_N$的距离的乘积。如果x离插值点越远，$\omega_{N+1}(x)$的值就会越大，从而误差也会增大。

c. $(N+1)!$: 这是阶乘项，N是插值多项式的阶数。

d. $f^{(N+1)}(\xi_x)$: 这是函数$f(x)$的$N+1$阶导数。导数越高，函数变化越剧烈，误差越大。

2. 误差的最大值 Maximum Value of Error: 为了得到插值的最大误差，我们使用以下不等式，这是多项式插值误差边界 Polynomial Interpolation Error Bound。

$$\max_{x\in[a,b]} |f(x) - p_N(x)| \le \max_{x\in[a,b]} |\omega_{N+1}(x)| \cdot \frac{\max_{z\in[a,b]} \left|f^{(N+1)}(z)\right|}{(N+1)!}$$

$$= |b-a|^{N+1} \cdot \frac{\max_{z\in[a,b]} \left|f^{(N+1)}(z)\right|}{(N+1)!}$$

这表明插值误差的上界受两个因素的影响:

a. $\omega_{N+1}(x)$: 插值点的选择（即x_i的位置）。

b. $f^{(N+1)}(z)$: 函数高级导数的最大值。

3. 选择切比雪夫点 Choosing Chebyshev Point

a. 为了尽可能减小误差，通常建议选择 Chebyshev 点作为插值点。Chebyshev 点可以**有效降低$\omega_{N+1}(x)$的最大值**，从而改善插值的精度。

b. 不同的插值方法和插值结果的准确性 Different Interpolation Methods and Accuracy of Interpolation Results

i. Computation of the polynomial interpolating function and its evaluation **can produce error**.

ii. 基于 Vandermonde 矩阵的多项式插值 Polynomial Interpolation Based on Vandermonde Matrix: 更大的 Vandermonde 矩阵会 ill-conditioned。随着插值点数量的增加，Vandermonde 矩阵的条件数 Condition Number 会变大，从而导致 Numerical Instability，并且增加误差。

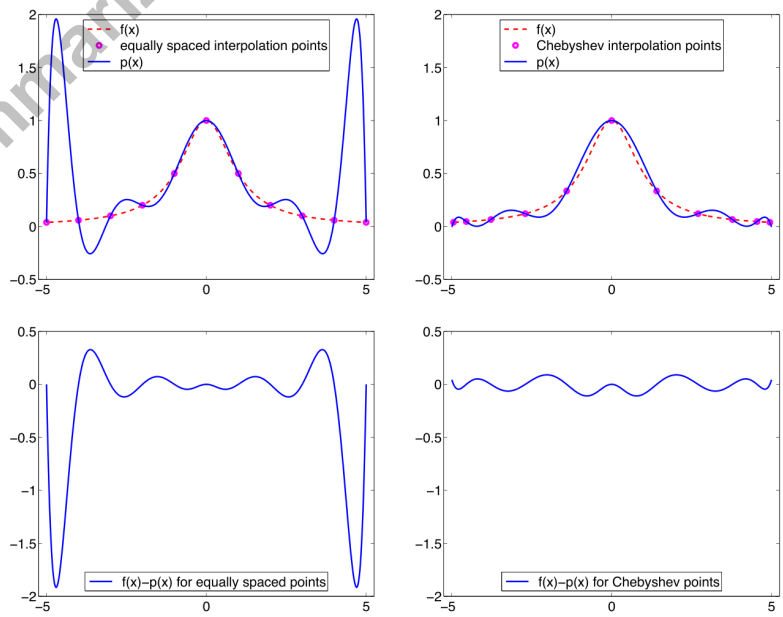
iii. 牛顿方法的病态性 Ill–Conditioned Newton Approach: 随着插值点数量的增加，Newton 方法的数值条件性甚至比 Vandermonde 系统还差。

iv. 端点处的插值 Ends of the Interval of Interpolation: 当我们在一个区间内进行插值时，通常插值误差在区间的端点处最大。这是因为插值多项式$p(x)$通常在区间内靠近插值点时，误差较小；而当在区间的两端时，点x离某些插值点更远，导致误差增大。

c. 插值误差的比较 Comparison of Interpolation Errors

i. Vandermonde、Newton、Lagrange 方法随着n的增加，误差逐渐增大，尤其在n大于 60 时，误差急剧增大，数值不稳定性显著。

ii. 龙格现象 Runge's Phenomenon: 龙格现象 Runge's Phenomenon 是指在高阶多项式插值中，尤其是使用等距插值点 Equally Spaced Interpolation Points 时，随着插值点数量 Number of Nodes 的增加，插值多项式在区间端点处会出现剧烈的波动，导致误差增大。这种现象在区间两端尤为明显，即误差主要集中在区间的边界处，而在区间中部插值的效果较好。



iii. Chebyshev 点则表现出了较好的稳定性，即使在$n = 100$时，误差依然相对较小。

多项式样条函数 Polynomial Splines

1. 使用样条函数插值的原因 The Reason of Using Polynomial Splines: Global interpolation often leads to larger errors because it uses a single high-degree polynomial to fit all data points, causing issues like the Runge phenomenon (oscillations at the edges) and numerical instability due to sensitivity to small changes in data. In contrast, polynomial splines divide the interval into smaller sections, using low-degree polynomials for each, ensuring better stability and accuracy in fitting local data patterns.
2. 线性样条插值 Linear Splines
    a. 定义 Definition: 线性样条插值 Linear Splines 是 simplest instance of continuous piecewise polynomial interpolation，它的每个插值区间上使用的是一条直线。也就是说，在每两个相邻的插值点之间，函数值通过一条直线连接。
    b. 函数值的连续性 Continuity of Function Values: 这种方法保证了在插值点 Nodes 的值是连续的，但是导数可能不连续。
    c. 线性样条的表达式 Expressions for Linear Splines: 在每个插值区间$[x_i, x_{i+1}]$上，
    $$f(x) = f(x_i) + m_i(x - x_i)$$
    其中，m_i是直线的斜率。这说明，在每个插值区间上，Linear Splines 就是一条简单的直线。
    d. 缺点 Disadvantage: Linear splines are simple, **but they are often not "smooth" enough**. The use of straight line segments between interpolation points ensures that the function values are continuous at the interpolation points, but the derivatives are not continuous, which makes the interpolated function visually crease or corner.
    e. 误差上界 Error Bound of Linear Splines
    $$\max_{x\in[a,b]} \left|f(x) - S_{1,N}(x)\right| \le \frac{h^2}{8} \cdot \max_{x\in[a,b]} \left|f^{(2)}(x)\right|$$
        i. 最大误差 $\max_{x\in[a,b]} \left|f(x) - S_{1,N}(x)\right|$: 这是函数$f(x)$和 linear splines $S_{1,N}(x)$之间的最大误差，也就是我们希望估计的插值误差。
        ii. 区间长度h: 这是插值点之间的最大距离
        $$h = \max_i(x_{i+1} - x_i)$$
        iii. 二阶导数的最大值 $\max_{x\in[a,b]} \left|f^{(2)}(x)\right|$: $f^{(2)}(x)$是函数$f(x)$的二阶导数。
        iv. Error Bound of Linear Splines 说明其 Error 随着h的增大而增大，并且其收到二阶导数的影响，如果函数变化剧烈，误差会增加。
3. 三次样条插值 Cubic Spline Interpolation
    a. 定义 Definition: 三次样条插值 Cubic Spline Interpolation 要求$f(x)$是$C^2[a,b]$函数。$C^2[a,b]$说明插值函数 Interpolating Function $f_i(x)$及其一阶、二阶导数在整个区间$[a,b]$上是连续的 continuous。这解决了 Linear Splines 的问题，即保证了 Interpolating Function 在每个区间的过渡是平滑的。

b. 插值条件 Interpolation Condition: 对于每个插值区间$[x_i, x_{i+1}]$，都需要满足如下条件
   i. 函数值连续性 Continuity of Function Values: 对于每个区间的两端，插值多项式 Interpolating Polynomial $f_i(x)$在点x_i和x_{i+1}处必须等于真实的函数值$f(x_i)$和$f(x_{i+1})$，即
   $$f_i(x_i) = f(x_i),\ \ f_i(x_{i+1}) = f(x_{i+1}),\ \ i = 0,1,\dots,n-1$$
   ii. 一阶导数连续性 Continuity of First-Order Derivatives: 插值多项式 Interpolating Polynomial $f_i(x)$的导数在每个插值点上必须连续
   $$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}),\ \ i = 0,1,\dots,n-2$$
   iii. 二阶导数的连续性 Continuity of Second-Order Derivatives: 插值多项式 Interpolating Polynomial $f_i(x)$的二阶导数在每个插值点上必须连续
   $$f_i''(x_{i+1}) = f_{i+1}''(x_{i+1}),\ \ i = 0,1,\dots,n-2$$
c. 三次样条插值的形式 Expressions for Cubic Spline Interpolation
   $$f(x) = f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$
   其中a_i, b_i, c_i, d_i是要通过插值点 Nodes 求解的系数。
d. 常见的边界条件 Common Boundary Conditions
   i. 自然边界条件 Natural Boundary Conditions: 假设二阶导数在两端为0，即在x_0和x_N处有
   $$f_1''(x_0) = 0, f_N''(x_N) = 0$$
   其方法的逼近精度没有其他边界条件高。
   ii. 二阶导数条件 Second Derivative Conditions: 在边界点指定二阶导数的具体值（必须知道真实函数在两端的二阶导数值才可以使用）
   $$f_1''(x_0) = f''(x_0), f_N''(x_N) = f''(x_N)$$
   其逼近精度较好，误差可以达到$O(h^4)$的精度。
   iii. 一阶导数条件 First Derivative Conditions: 在边界点指定一阶导数的具体值（必须知道真实函数在两端的一阶导数值才可以使用）
   $$f_1'(x_0) = f'(x_0), f_N'(x_N) = f'(x_N)$$
   其和 Second Derivative Conditions 一样，逼近精度较好，误差可以达到$O(h^4)$的精度。
   iv. 非节点条件 Not-A-Knot Conditions: 在第二个和倒数第二个点（即x_1和x_{N-1}）要求三阶导数连续
   $$f_1'''(x_1) = f_2'''(x_1), f_{N-1}'''(x_{N-1}) = f_N'''(x_{N-1})$$
   这种条件不在端点施加导数条件，而是在内部点进行约束。其可以让曲线更加平滑，是很多软件的默认条件。
e. 求解案例 Example: 现在有三个插值点$x_0 = 0, f(x_0) = 1.1; x_1 = 1, f(x_1) = 0.9; x_2 = 2, f(x_2) = 2.0$，请通过这些点，构建 Cubic Spline Interpolation。

i. Step 1: 根据插值区间，写出所有的$f_i(x)$。

$$f_0(x) = a_0x^3 + b_0x^2 + c_0x + d_0 \text{ on } [0,1]$$
$$f_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1 \text{ on } [1,2]$$

ii. Step 2: 通过函数值连续性 Continuity of Function Values 构建四个式子

$$f_0(0) = 1.1 = a_0 \cdot 0^3 + b_0 \cdot 0^2 + c_0 \cdot 0 + d_0 = d_0$$
$$f_0(1) = 0.9 = a_0 \cdot 1^3 + b_0 \cdot 1^2 + c_0 \cdot 1 + d_0 = a_0 + b_0 + c_0 + d_0$$
$$f_1(1) = 0.9 = a_1 \cdot 1^3 + b_1 \cdot 1^2 + c_1 \cdot 1 + d_1 = a_1 + b_1 + c_1 + d_1$$
$$f_1(2) = 2.0 = a_1 \cdot 2^3 + b_1 \cdot 2^2 + c_1 \cdot 2 + d_1 = 8a_1 + 4b_1 + 2c_1 + d_1$$

iii. Step 3: 通过一阶导数连续性 Continuity of First-Order Derivatives 和二阶导数的连续性 Continuity of Second-Order Derivatives 构建两个式子

$$f_i'(x) = 3a_ix^2 + 2b_ix + c_i$$
$$f_i''(x) = 6a_ix + 2b_i$$

我们需要在$x_1 = 1, f(x_1) = 0.9$确保两个连续性

$$3a_0 + 2b_0 + c_0 = 3a_1 + 2b_1 + c_1$$
$$6a_0 + 2b_0 = 6a_1 + 2b_1$$

iv. Step 4: 通过 Natural Boundary Conditions、Second Derivative Conditions、First Derivative Conditions 或者 Not-a-knot Conditions 来构建最后两个式子。在该题的例子中，由于没有超过 3 个点，并且不知道原函数及其二阶导数具体值，因此只能使用 Natural Boundary Conditions。

$$f''(x_0) = f''(x_n) = 0$$
$$2b_0 = 0, 12a_1 + 2b_1 = 0$$

f. 方程系统的唯一解 Unique Solution of the System of Equations

i. 对于三次样条插值 Cubic Spline Interpolation，我们需要在每个子区间（两个插值点之间）上定义一个三次多项式，为了确保这些多项式的系数，我们需要构建一个方程系统。

ii. 假设我们有N个区间，即$N+1$个节点，那么每个区间上需要4个条件（因为每个区间的多项式是 3 次的），因此一共需要$4N$个方程。

iii. The system of 4N linear constraints has a unique solution as long as knots are distinct.

g. 误差界限 Error Bound

$$\max_{x\in[x_{i-1},x_i]} |f(x) - p_3(x)| \le Ch^4 \cdot \max_{x\in[a,b]} \left|f^{(4)}(x)\right|$$

i. 这是**两个相邻插值点之间的最大误差公式**。$f(x)$为Real Function，而$p_3(x)$是 Cubic Spline Interpolation Polynomial。

ii. C 为一个常数 Constant。

iii. h是插值点之间的最大距离$h = \max_{\text{i}} |x_i - x_{i-1}|$。

iv. $f^{(4)}(x)$是函数的四阶导数，反应函数的光滑性。

v. 当插值点分布的间距h较小时，误差会以h^2的速率快速减小。

vi. 如果我们正确的设定 first and second derivative value 或者使用 not-a-knot cubic splines，Cubic Spline Interpolation 就可以完美重现三阶多项式，即其在拟合两个点之间的三次多项式的时候是没有误差的。

最小二乘拟合 Least Squares Fitting

1. 为什么有时候使用最小二乘拟合而不是插值 Why Least Squares Fitting?
   a. 希望寻求数据中的趋势 Seeking trend in data.
   b. 数据本身可能有测量错误 Data may contain measurement errors.
   c. 给到的参数较少 Desire model function that depends on only a few parameters x_j.
   d. 更直接和效率构建方程 Construction of function “straightforward and efficient”.
2. 最小二乘多项式的定义 Definition of Least Squares Polynomial
   a. 我们定义一个任意拟合数据的多项式$q_M(x)$，其是一个 Polynomial of Degree M。假设f_r是实际值，那么误差就是
   $$q_M(x_r) - f_r$$
   b. 我们可以定义一个误差度量，即平方误差和 Sum of the Squares of Errors，来说明 How well $q_M(x)$ fits data。
   $$\sigma(q_M) = \sum_{r=0}^{N} \{q_M(x_r) - f_r\}^2$$
   c. 现在，我们可以设一个$p_M(x)$，是 Polynomial of Degree M。当
   $$\sigma(p_M) \leq \sigma(q_M) \; \forall \; q_M(x)$$
   我们称，$p_M(x)$是 M 次的最小二乘多项式 Least Squares Polynomial。
   d. 关键点 Key Points
      i. 最小二乘法 Least Squares Fitting 寻找的多项式是让误差度量σ_M最小的多项式，因此我们定义$\sigma(p_M) \leq \sigma(q_M) \; \forall \; q_M(x)$。
      ii. 尽管 Least Squares Polynomial $p_M(x)$产生了最小的误差度量σ_M，这不意味着其一定是对数据的“最佳拟合 Best Fit”，其可能出现过拟合或欠拟合 Overfitting or Underfitting 的情况。
3. 如何寻找最小二乘多项式 How to Find Least Squares Polynomial?
   a. 幂级数的多项式形式 Polynomial Form of Power Series
   $$p_M(x) = a_0 + a_1 x + \cdots + a_M x^M$$
   其中，$a_0, a_1, \ldots, a_M$是待定的未知系数。
   b. 误差函数 Error Function: 误差度量σ_M是关于这些未知系数a_i的二次函数。
   $$\sigma_M = \sum_{r=0}^{N} \{p_M(x_r) - f_r\}^2$$

c. 如何寻找a_i使得σ_M最小？How to Find a_i such that σ_M Is Minimized?

i. 基本思想 Basic Ideas: 对误差函数 Error Function 对于每个系数a_i分别求偏导，并让偏导数为 0。

$$\frac{\partial \sigma_M}{\partial a_0} = 0, \frac{\partial \sigma_M}{\partial a_1} = 0, \dots, \frac{\partial \sigma_M}{\partial a_M} = 0$$

ii. 正规方程 Normal Equations: 这些导数方程组合起来形成了一个线性方程组，被称为正规方程 Normal Equations。通过解 Normal Equations，我们可以得到最优的a_i，从而最小化误差。

$$a_0 \sum_{r=0}^{N} x_r^j + a_1 \sum_{r=0}^{N} x_r^{j+1} + \cdots + a_M \sum_{r=0}^{N} x_r^{j+M} = \sum_{r=0}^{N} f_r x_r^j, j = 0,1, \dots, M$$

$$Aa = b$$

$$\begin{bmatrix} \sum_{r=0}^{N} 1 & \sum_{r=0}^{N} x_r & \dots & \sum_{r=0}^{N} x_r^M \\ \sum_{r=0}^{N} x_r & \sum_{r=0}^{N} x_r^2 & \dots & \sum_{r=0}^{N} x_r^{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{r=0}^{N} x_r^M & \sum_{r=0}^{N} x_r^{M+1} & \dots & \sum_{r=0}^{N} x_r^{2M} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \sum_{r=0}^{N} f_r \\ \sum_{r=0}^{N} f_r x_r \\ \vdots \\ \sum_{r=0}^{N} f_r x_r^M \end{bmatrix}$$

其中 N 是数据点总数减 1。

iii. 矩阵 A 的正定性 A Is Symmetric and Positive Definite: 保证在最小二乘法中所求的解是误差函数的全局最小点，也就是说明我们的拟合是最优且稳定的。

d. 使用其他基函数在最小二乘法 Other Basis Functions in Least Squares Fitting

$$p_M(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_M\phi_M(x)$$

$$\begin{bmatrix} \sum_{r=0}^{N}\phi_0(x_r)^2 & \sum_{r=0}^{N}\phi_0(x_r)\phi_1(x_r) & \dots & \sum_{r=0}^{N}\phi_0(x_r)\phi_M(x_r) \\ \sum_{r=0}^{N}\phi_1(x_r)\phi_0(x_r) & \sum_{r=0}^{N}\phi_1(x_r)^2 & \dots & \sum_{r=0}^{N}\phi_1(x_r)\phi_M(x_r) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{r=0}^{N}\phi_M(x_r)\phi_0(x_r) & \sum_{r=0}^{N}\phi_M(x_r)\phi_1(x_r) & \dots & \sum_{r=0}^{N}\phi_M(x_r)^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{r=0}^{N} f_r\phi_0(x_r) \\ \sum_{r=0}^{N} f_r\phi_1(x_r) \\ \vdots \\ \sum_{r=0}^{N} f_r\phi_M(x_r) \end{bmatrix}$$

i. 我们可以使用其他基函数 Basis Functions 来代替幂级数，在上述公式中$\phi_M(x)$就是选取的基函数，其可以是拉格朗日基函数 Lagrange Basis Function、切比雪夫基函数 Chebyshev Basis Function 或其他。

ii. 不过，随着基函数数量的增加，矩阵的病态性 ill-conditioning 可能会更加严重，意味着求解可能会变的不稳定。

e. 案例一 Example 1: 有四个数据点$(1,1),(2,2),(3,2),(4,4)$，请使用二次多项式$p(x) = a_0 + a_1x + a_2x^2$来拟合这些数据点。

i. Step 1: 定义误差函数

$$\sigma_M = \sum_{r=0}^{N}\{p_M(x_r) - f_r\}^2 = \sum_{r=0}^{N}(a_0 + a_1x_r + a_2x_r^2 - f_r)^2$$

ii. Step 2: 对σ_M对a_0, a_1, a_2分别求导，并让导数为 0，得到 3 个线性方程。

$$\frac{\partial\sigma_M}{\partial a_0} = 2\sum_{r=0}^{N}(a_0 + a_1x_r + a_2x_r^2 - f_r) = 0$$

$$4a_0 + \sum_{r=0}^{N}a_1x_r + \sum_{r=0}^{N}a_2x_r^2 = \sum_{r=0}^{N}f_r$$

$$\frac{\partial\sigma_M}{\partial a_1} = 2\sum_{r=0}^{N}x_r(a_0 + a_1x_r + a_2x_r^2 - f_r) = 0$$

$$a_0\sum_{r=0}^{N}x_r+a_1\sum_{r=0}^{N}x_r^2+a_2\sum_{r=0}^{N}x_r^3=\sum_{r=0}^{N}f_rx_r$$

$$\frac{\partial\sigma_M}{\partial a_2}=2\sum_{r=0}^{N}x_r^2(a_0+a_1x_r+a_2x_r^2-f_r)=0$$

$$a_0\sum_{r=0}^{N}x_r^2+a_1\sum_{r=0}^{N}x_r^3+a_2\sum_{r=0}^{N}x_r^4=\sum_{r=0}^{N}f_rx_r^2$$

iii. Step 3: 形成线性方程组（正规方程）

$$\begin{bmatrix}4 & \sum_{r=0}^{3}x_r & \sum_{r=0}^{3}x_r^2\\ \sum_{r=0}^{3}x_r & \sum_{r=0}^{3}x_r^2 & \sum_{r=0}^{3}x_r^3\\ \sum_{r=0}^{3}x_r^2 & \sum_{r=0}^{3}x_r^3 & \sum_{r=0}^{3}x_r^4\end{bmatrix}\begin{bmatrix}a_0\\a_1\\a_2\end{bmatrix}=\begin{bmatrix}\sum_{r=0}^{3}f_r\\ \sum_{r=0}^{3}f_rx_r\\ \sum_{r=0}^{3}f_rx_r^2\end{bmatrix}$$

$$\begin{bmatrix}4 & 10 & 30\\ 10 & 30 & 100\\ 30 & 100 & 354\end{bmatrix}\begin{bmatrix}a_0\\a_1\\a_2\end{bmatrix}=\begin{bmatrix}9\\27\\91\end{bmatrix}$$

iv. Step 4: 求解线性方程组即可得到a_0, a_1, a_2。

f. 案例二 Example 2: 有四个数据点$(-1,1),(-0.5,2),(0.5,2),(1,4)$，请使用切比雪夫基函数 Chebyshev Basis Function 来拟合这些数据点。

i. 我们希望拟合的多项式是:

$$p(x)=a_0T_0(x)+a_1T_1(x)+a_2T_2(x)$$

ii. 正规方程可以写成矩阵形式

$$\begin{bmatrix} \sum_{r=0}^{N} T_0(x_r)^2 & \sum_{r=0}^{N} T_0(x_r)T_1(x_r) & \sum_{r=0}^{N} T_0(x_r)T_2(x_r) \\ \sum_{r=0}^{N} T_1(x_r)T_0(x_r) & \sum_{r=0}^{N} T_1(x_r)^2 & \sum_{r=0}^{N} T_1(x_r)T_2(x_r) \\ \sum_{r=0}^{N} T_2(x_r)T_0(x_r) & \sum_{r=0}^{N} T_2(x_r)T_1(x_r) & \sum_{r=0}^{N} T_2(x_r)^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{r=0}^{N} f_r T_0(x_r) \\ \sum_{r=0}^{N} f_r T_1(x_r) \\ \sum_{r=0}^{N} f_r T_2(x_r) \end{bmatrix}$$

在这道题中，我们可以知道$T_0(x) = 1, T_1(x) = x, T_2(x) = 2x^2 - 1$。对于给定的数据点，我们代入计算即可。

4. 通过设计矩阵构建正规方程 Constructing Normal Equations by Design Matrix
   a. 设计矩阵 Design Matrix: 对于 Basis Function 是 Power Function 的情况，可以设矩阵 V 是设计矩阵 Design Matrix，其包含数据点x_i的所有幂次信息。

$$V = \begin{bmatrix} 1 & x_0 & \dots & x_0^M \\ 1 & x_1 & \dots & x_1^M \\ 1 & x_2 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \dots & x_N^M \end{bmatrix}, V \in \mathcal{R}^{n \times m}$$

   b. 广义上的设计矩阵 Generalized Design Matrix: 当 Basis Function 不是基函数，而是其他函数时，我们的设计矩阵 Design Matrix V 为

$$V = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_m) & \phi_1(x_m) & \cdots & \phi_n(x_m) \end{pmatrix}, V \in \mathcal{R}^{n \times m}$$

   c. 构建正规方程 Constructing Normal Equations: 我们最初对于误差度量的定义是$\sigma(q_M) = \sum_{r=0}^{N} \{q_M(x_r) - f_r\}^2$，我们也可以使用二范数 2-Norm 来表示。

$$\| Va - f \|_2^2 = \sum_{r=0}^{N} (p_M(x_r) - f_r)^2$$

由于二范数 2-Norm 的平方和等价于误差项的平方和，因此我们可以得出上述等式。最小二乘法 Least Squares Fitting 目标是最小化这种误差平方和，即

$$\min_a \| Va - f \|_2^2$$

通过对上式中的a求导并令导数为 0，可以得到正规方程 Normal Equations。

根据 2-Norm 的公式$\| x \|_2 = \sqrt{x^T x}$，因此

$$\| Va - f \|_2^2 = (Va - f)^T (Va - f) = a^T V^T V a - 2 f^T V a + f^T f$$

$$\frac{\partial}{\partial a}(a^T V^T V a - 2 f^T V a + f^T f) = 2 V^T V a - 2 V^T f$$

$$2 V^T V a - 2 V^T f = 0 \Rightarrow V^T V a = V^T f$$

因此，我们可以将原 Normal Equations 写为$V^T V a = V^T f$。

d. 案例 Example: 有四个数据点$(1,1), (2,2), (3,2), (4,4)$，请使用二次多项式$p(x) = a_0 + a_1 x + a_2 x^2$来拟合这些数据点。

i. Step 1: 构建设计矩阵 Design Matrix 和向量f。

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix}$$

ii. Step 2: 构建正规方程 Normal Equations。

$$V^T V a = V^T f$$

$$V^T V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} = \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix}$$

$$V^T f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 9 \\ 27 \\ 91 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 9 \\ 27 \\ 91 \end{bmatrix}$$

iii. Step 3: 解正规方程 Normal Equations。

5. QR 分解解最小二乘法问题 QR Decomposition for Solving Least Squares Problems

a. 为什么使用 QR 分解 Why QR Decomposition: 如果 V 是 ill conditioned 的，可能意味着$V^T V a = V^T f$的误差可能会增大。这个时候，我们可以使用 QR 分解来解决 Least Squares Problems。

b. 如何使用 QR 分解 How to Use QR Decomposition:

$$V = QR$$

$Q \in \mathcal{R}^{n \times n}$ is orthogonal; $R \in \mathcal{R}^{n \times m}$ is upper triangular.

因此，我们可以通过解如下式子来变相解 Normal Equations

$$Ra = Q^T f$$

在求解的过程中，由于 V 不是方阵，这导致了 R 也并非方阵的问题。

为了得出近似解，我们只需要考虑 R 上面的$m \times m$方阵的部分，并且只需要考虑$Q^T f$的前 m 项。通过这样的方法来得出近似解。具体推导过程如下：

$$\begin{aligned} ||Va - f||_2^2 &= ||QRa - f||_2^2 \\ &= ||Q^T(QRa - f)||_2^2 \\ &= ||Q^T QRa - Q^T f||_2^2 \\ &= ||Ra - Q^T f||_2^2 \\ &= \left\| \begin{bmatrix} R_m \\ 0 \end{bmatrix} a - \begin{bmatrix} b \\ c \end{bmatrix} \right\|_2^2 \\ &= ||R_m a - b||_2^2 + ||c||_2^2 \end{aligned}$$

我们可以通过 QR 分解将$||Va - f||_2^2$转化为$||R_m a - b||_2^2 + ||c||_2^2$。其中$R_m$是一个$m \times m$的方阵，b 是$Q^T f$的前 m 项，c 是剩余项。在具体计算和估计的过程中，我们可以省略$||c||_2^2$这一项。

c. 案例 Example: 有四个数据点$(1,1), (2,2), (3,2), (4,4)$，请使用二次多项式$p(x) = a_0 + a_1 x + a_2 x^2$来拟合这些数据点。

i. Step 1: 构建设计矩阵 Design Matrix 和向量f。

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix}$$

ii. Step 2: 针对 V 进行 QR 分解。

iii. Step 3: 计算$Q^T f$，并只保留前 3 行。对于 R，也只保留其3×3方阵的部分。

iv. Step 4: 根据$Ra = Q^T f$，求解出a。

第五单元 微分与积分 Chapter 5 Differentiation and Integration

微分 Differentiation

1. 为什么要使用数值微分一个方程 Why Differentiate A Function Numerically?
   a. Derivatives may be complicated to determine analytically.
   b. Function may not be explicitly known.
2. 微分的标准定义 Formal Definition of a Derivative

a. 微分的表达式 Expression for Differentiation

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

在上述公式中，假设$f(x)$是 Differentiable，且$f(x)$ defined in interval centered at x。

b. 如何在 MATLAB 中选择h How Do We Choose h?

i. 离散化误差或截断误差 Discretization Error or Truncation Error: 这些误差随着h的减小而减小。如果函数f具有足够的平滑性（即函数的导数存在且连续），那么将h取得越小，Discretization Error 也会越小。因此，在理论上，我们希望h尽可能缩小，以减小误差。

ii. 灾难性取消 Catastrophic Cancellation: 但是，在 MATLAB 中，当h非常小的时候，数值计算会受到 Catastrophic Cancellation 的限制。因此，实际上过小的h会引发数值问题。

iii. 如何选择 How to Choose: 当h远小于$\sqrt{\epsilon_m}$时，舍入误差 Roundoff Error 开始占主导地位。此时，继续减少h不在有效地降低 Discretization Error，反而会不断增加 Roundoff Error。

3. 前向差分法和后向差分法 Forward Difference and Backward Difference

a. 前向差分法 Forward Difference: Forward Difference 就是使用$f(a+h)$的 Taylor Series Expansion 推得$f'(a)$。对于 Taylor Series Expansion，我们只保留到二阶导数项。

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2} f''(\xi)$$

因此，我们可以推得

$$f'(a) = \frac{f(a+h) - f(a)}{h} - \frac{h}{2} f''(\xi)$$

其中，$\frac{h}{2} f''(\xi)$是误差项，并且贡献了主要误差。对于前向差分法 Forward Difference，我们只使用$f(a+h)$和$f(a)$进行逼近，其误差精度是$O(h)$。

前向差分法 Forward Difference 的估计公式为

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

b. 后向差分法 Backward Difference: Backward Difference 就是使用$f(a-h)$的 Taylor Series Expansion 推得$f'(a)$。对于 Taylor Series Expansion，我们只保留到二阶导数项。

$$f(a-h) = f(a) - hf'(a) + \frac{h^2}{2} f''(\xi)$$

因此，我们可以推得

$$f'(a) = \frac{f(a) - f(a-h)}{h} + \frac{h}{2}f''(\xi)$$

其中，$\frac{h}{2}f''(\xi)$是误差项，并且贡献了主要误差。对于后向差分法 Backward Difference，我们只使用$f(a-h)$和$f(a)$进行逼近，其误差精度也是$O(h)$。

后向差分法 Backward Difference 的估计公式为

$$f'(a) \approx \frac{f(a) - f(a-h)}{h}$$

c. 高阶单边差分法 One-Sided Difference (Higher Order)

i. 之前的单边差分 Previous One-Sided Difference: 对于之前的 One-Sided Difference，我们都使用$f(a+h)$或者$f(a-h)$进行估计，也就是在a点的左边（后向差分 Backward Difference）或者右边（前向差分 Forward Difference）取一个h，仅使用了一个点，这个方法的误差是$O(h)$。

ii. 高阶单边差分法 One-Sided Difference (Higher Order): 我们使用了单边两个步长h来近似$f'(a)$。对于单侧取两个h来近似，误差精度是$O(h^2)$。

- 高阶前向差分公式 Higher Order Forward Difference: 使用了点$f(a), f(a+h), f(a+2h)$，即在a点右侧取了两个h。

$$f'(a) = \frac{-3f(a) + 4f(a+h) - f(a+2h)}{2h}$$

- 高阶后向差分公式 Higher Order Backward Difference: 使用了点$f(a), f(a-h), f(a-2h)$，即在a点左侧取了两个h。

$$f'(a) = \frac{f(a-2h) - 4f(a-h) + 3f(a)}{2h}$$

4. 中间差分法 Centered Difference

a. 一阶导数的中间差分法 Centered Difference for First Order Derivatives: 对于一阶导数的 Centered Difference，我们使用$f(a+h)$的 Taylor Series 减去$f(a-h)$的 Taylor Series 得到。

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{3!}f^{(3)}(\xi_1)$$

$$f(a-h) = f(a) - hf'(a) + \frac{h^2}{2}f''(a) - \frac{h^3}{3!}f^{(3)}(\xi_2)$$

在上述 Taylor Series Expansion 中，我们保留到三阶导数项。这是因为当我们使用$f(a+h) - f(a-h)$时，偶次项（例如$\frac{h^2}{2}f''(a)$）会互相抵消，而奇次项（例如$hf'(a)$和$\frac{h^3}{3!}f^{(3)}(a)$）会被保留下来。这种抵消的效果让误差项提升到$O(h^2)$水平，这使得 Centered Difference 有更高的精度。

$$f(a+h)-f(a-h)=2hf'(a)+\frac{2h^3}{3!}f^{(3)}(\xi)$$

$$f'(a)=\frac{f(a+h)-f(a-h)}{2h}-\frac{h^2}{3!}f^{(3)}(\xi)$$

其中，$\frac{h^2}{3!}f^{(3)}(\xi)$是误差项，并且贡献了主要误差。一阶导数的中间差分法 Centered Difference Method for First Order Derivatives 的估计公式为

$$f'(a)\approx\frac{f(a+h)-f(a-h)}{2h}$$

b. 二阶导数的中间差分法 Centered Difference for Second Order Derivatives: 对于二阶导数的 Centered Difference，我们使用我们使用$f(a+h)$的 Taylor Series 加去$f(a-h)$的 Taylor Series 得到。

$$f(a+h)=f(a)+hf'(a)+\frac{h^2}{2}f''(a)+\frac{h^3}{3!}f^{(3)}(a)+\frac{h^4}{4!}f^{(4)}(\xi_1)$$

$$f(a-h)=f(a)-hf'(a)+\frac{h^2}{2}f''(a)-\frac{h^3}{3!}f^{(3)}(a)+\frac{h^4}{4!}f^{(4)}(\xi_2)$$

在上述 Taylor Series Expansion 中，我们保留到四阶导数项。这是因为当我们使用$f(a+h)+f(a-h)$时，奇次项（例如$hf'(a)$和$\frac{h^3}{3!}f^{(3)}(a)$）会互相抵消，而偶次项（例如$\frac{h^2}{2}f''(a)$和$\frac{h^4}{4!}f^{(4)}(a)$）会被保留下来。这种抵消的效果让误差项提升到$O(h^2)$水平，这使得 Centered Difference 有更高的精度。

$$f(a+h)+f(a-h)=2f(a)+h^2f''(a)+\frac{h^4}{12}f^{(4)}(\xi)$$

$$f''(a)=\frac{f(a+h)-2f(a)+f(a-h)}{h^2}-\frac{h^2}{12}f^{(4)}(\xi)$$

其中，$\frac{h^2}{12}f^{(4)}(\xi)$是误差项，并且贡献了主要误差。二阶导数的中间差分法 Centered Difference Method for Second Order Derivatives 的估计公式为

$$f''(a)\approx\frac{f(a+h)-2f(a)+f(a-h)}{h^2}$$

c. 中心四阶差分法 Centered Fourth Order Difference

i. 之前的中间差分 Previous Centered Difference: 对于之前的 Centered Difference，我们都使用$f(a+h)$和$f(a-h)$进行估计，也就是在a点的左右各取一个h，仅使用了两个点，这个方法的误差是$O(h^2)$。

ii. 中心四阶差分法 Centered Fourth Order Difference: 该公式会在a点的左右各取两个h，也就是使用$f(a-2h),f(a-h),f(a+h),f(a+2h)$四个点进行估计，这种方法的误差是$O(h^4)$。

$$f'(a)=\frac{1}{12h}(f(a-2h)-8f(a-h)+8f(a+h)-f(a+2h))$$

5. 误差对比 Comparison of Errors

a. 精度对比 Precision Comparison: 观察公式中的误差项中的h，h的阶数减 1 为$O(h)$的阶数。

Method Name	**Truncation Error**	**Precision**
Forward Difference	$-\frac{h}{2}f''(c)$	$O(h)$
Backward Difference	$\frac{h}{2}f''(c)$	$O(h)$
Second Order Forward Difference	$-\frac{h}{3}f'''(c)$	$O(h^2)$
Second Order Backward Difference	$\frac{h}{3}f'''(c)$	$O(h^2)$
Centered Difference for First Order Derivatives	$-\frac{h^2}{6}f'''(c)$	$O(h^2)$
Centered Difference for Second Order Derivatives	$-\frac{h^2}{12}f^{(4)}(c)$	$O(h^2)$
Centered Fourth Order Difference for First Order Derivatives	$-\frac{h^4}{30}f^{(5)}(c)$	$O(h^4)$

b. 高阶的中心差分方法的好处 Benefits of Higher-Order Centered Difference: 我们通过一个案例$f(x)=e^x$ at $x=0$ ($f'(x)=1$)，来说明几个误差的对比。下图分别是 Forward Difference、Centered Difference 和 Centered Fourth Order Difference 的对比。

h	$\frac{e^h-1}{h}-1$	$\frac{e^h-e^{-h}}{2h}-1$	$\frac{-e^{2h}+8e^h-8e^{-h}+e^{-2h}}{12h}-1$
0.1	5.17e-2	1.67e-3	3.33e-6
0.01	5.02e-3	1.67e-5	3.33e-10
0.001	5.0e-4	1.67e-7	-4.54e-14
0.0001	5.0e-5	1.67e-9	-2.60e-13
0.00001	5.0e-6	1.21e-11	-3.63e-12

i. Forward Difference 的误差随着h减小成线性缩小，符合$O(h)$的定义。
ii. Centered Difference 的误差呈现出二阶精度，即随着h^2减小，符合$O(h^2)$的定义。
iii. 四阶中心差分 Centered Fourth Order Difference 的误差远小于前两种方法，在小h下误差几乎趋近于零（误差随着h^4减小）。
iv. 较高阶差分公式（如四阶中心差分）在小h下具有更高精度。
v. 随着h的减小，较高阶的中心差分方法的误差收敛速度更快。

积分 Integration

1. 积分的标准定义与数值积分求积法则 Definition of Integration and Quadrature Rule
   a. 积分的标准定义 Definition of Integration

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

这是一个定积分 Definite Integral，其中f是一个可积函数 Integrable Function，$[a,b]$是一个有限区间 Finite Interval。

b. 数值积分求积法则 Quadrature Rule

$$\int_a^b f(x)dx \approx \sum_{j=0}^{n} w_j f(x_j)$$

数值积分求积公式 Quadrature Rule 用于通过加权求和的方式来近似计算函数在某一区间上的定积分 Definite Integral。其通过选取区间上的节点 Nodes x_j 以及相应的权重 Weights w_j，将积分转化为这些节点处函数值的加权和 weighted sum of function values at nodes。

c. 插值求积法 Interpolatory Quadrature: 我们使用上章的 Lagrange Interpretation Polynomial，构建出一个通过已知数据点的 Interpretation Polynomial $p_N(x)$。我们设真实积分为$I(f)$，我们通过 Interpretation Polynomial 来近似积分

$$I(f) \approx I(p_N)$$

插值多项式的基本公式是

$$p_N(x) = f_0\ell_0(x) + f_1\ell_1(x) + \cdots + f_N\ell_N(x)$$

在此处，我们可以将对$f(x)$的积分进行转化——对原函数$f(x)$的积分就相当于对 Interpretation Polynomial $p_N(x)$的积分。

$$\int_a^b f(x)dx \approx \int_a^b p_N(x)dx = \int_a^b \sum_{j=0}^{n} f(x_j)L_j(x)dx$$
$$= \sum_{j=0}^{n} f(x_j)\int_a^b L_j(x)dx$$

此时，我们发现$\sum_{j=0}^{n} f(x_j)$被提出，原方程保留了$\int_a^b L_j(x)dx$，即 Lagrange Basis Function 的积分。我们可以将 Lagrange Basis Function 的积分进行预先计算，这就是我们 Quadrature Rule 中的权重 Weights w_j。

$$w_j = \int_a^b L_j(x)dx$$

因此，我们有如下式子，被称为 Interpolatory Quadrature

$$I(f) \approx I(p_N) = I(\sum_{i=0}^{N} f(x_i)\ell_i(x)) = \sum_{i=0}^{N} I(\ell_i(x))f(x_i) = \sum_{i=0}^{N} w_i f(x_i) \equiv R(f)$$

d. 牛顿-科特斯求积法则 Newton-Cotes Rule: 牛顿-科特斯求积法则 Newton-Cotes Rule 是特殊的 Interpolatory Quadrature，其通过 Interpretation Polynomial 来近似函数，从而计算定积分。具体来说，其使用在区间$[a,b]$上选择$N+1$个等距 Nodes $(x_0, x_1, x_2, \ldots, x_N)$ 从而构造一个多项式。该多项式在每个 Nodes 处的函数值与原函数$f(x)$相同。我们使用该多项式的积分作为原积分的近似

值。Newton-Cotes Rule 可以根据 Nodes 是否包括 Endpoints 分为 Closed Formula 和 Open Formula。

i. 闭合公式 Closed Formula: Nodes 包括 Endpoints a, b。
   - 2 Points Closed Newton-Cotes Rule: Trapezoidal Rule（使用两个 Nodes，近似的多项式为一阶多项式）
   - 3 Points Closed Newton-Cotes Rule: Simpson's Rule（使用三个 Nodes，近似的多项式为两阶多项式）

ii. 开放公式 Open Formula: Nodes 不包括括 Endpoints a, b。
   - 1 Point Open Newton-Cotes Rule: Midpoint Rule（使用一个节点，即区间中点处取样近似积分，近似的多项式为 Constant）

2. 中点法 Midpoint Rule

a. 中点法的推导 Derivation of Midpoint Rule: 中点法 Midpoint Rule 是选择一个 Node x_0为区间的中点，并将整个区间长度$b-a$作为 Weight w_0。

$$\text{Quadrature Rule: } \int_a^b f(x)dx \approx \sum_{j=0}^{n} w_j f(x_j)$$

$$\text{Node: } x_0 = \frac{a+b}{2}$$

$$\text{Weight: } w_0 = b-a$$

$$\text{Midpoint Rule: } \int_a^b f(x)dx \approx w_0 \cdot f(x_0) = (b-a) \cdot f\left(\frac{a+b}{2}\right)$$

b. 中点法的标准形式 Standard Form of the Midpoint Method

$$I_f \approx R(f) = (b-a) f\left(\frac{b+a}{2}\right)$$

3. 梯形法 Trapezoidal Rule

a. 梯形法的推导 Derivation of Trapezoidal Rule: 梯形法 Trapezoidal Rule 是选择区间的两个端点$x_0 = a, x_1 = b$，并且两个端点的权重都为区间长度的一半。

$$\text{Weight: } w_0 = w_1 = \frac{b-a}{2}$$

$$\int_a^b f(x)dx \approx w_0 \cdot f(x_0) + w_1 \cdot f(x_1) = \frac{b-a}{2} \cdot f(a) + \frac{b-a}{2} \cdot f(b)$$

除此之外，我们也可以使用 Interpolatory Quadrature 推导 Trapezoidal Rule，其是 2 Points Closed Newton-Cotes Rule。两个 Lagrange Polynomial 分别是

$$l_0(x) = \frac{x-b}{a-b}, l_1(x) = \frac{x-a}{b-a}$$

根据 Interpolatory Quadrature Rule Weights Formula

$$w_0 = \int_a^b L_0(x)dx = \int_a^b \frac{x-b}{a-b}dx = \frac{b-a}{2}$$

$$w_1 = \int_a^b L_1(x)dx = \int_a^b \frac{x-a}{b-a}dx = \frac{b-a}{2}$$

根据 Quadrature Rule，我们可以得到

$$\int_a^b f(x)dx \approx w_0 \cdot f(x_0) + w_1 \cdot f(x_1) = \frac{b-a}{2} \cdot f(a) + \frac{b-a}{2} \cdot f(b)$$

b. 梯形法的标准形式 Standard Form of the Trapezoidal Method

$$I_f \approx R(f) = \frac{b-a}{2}[f(a) + f(b)]$$

4. 复合求积法 Composite Quadrature Methods

a. 复合求积法的逻辑 Logic of Composite Quadrature Method: 我们将区间$[a,b]$分为较多较小的子区间，然后在每个子区间上应用某种求积公式，然后将结果加总。

i. 区间划分 Interval Dividing: 将$[a,b]$等分为r个小区间，每个小区间长度为$h = \frac{b-a}{r}$。其可以表示为$[x_{i-1}, x_i]$，其中$x_i = a + ih$。

ii. 累加结果 Cumulative Result

$$\int_a^b f(x)dx \approx \sum_{i=1}^{r} \int_{x_{i-1}}^{x_i} f(x)dx$$

b. 复合中点法 Composite Midpoint Method: 如果我们将 Midpoint Method 套入 Composite Quadrature Method，Midpoint Method 中的b是x_i，a是x_{i-1}。

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx hf\left(\frac{x_{i-1} + x_i}{2}\right) \quad \int_a^b f(x)dx \approx h\sum_{i=1}^{r} f(a + (i - 1/2)h)$$

c. 复合梯形法 Composite Trapezoidal Method: 如果我们将 Trapezoidal Method 套入 Composite Quadrature Method，Trapezoidal Method 中的b是x_i，a是x_{i-1}。

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{2}\sum_{i=1}^{r} [f(x_{i-1}) + f(x_i)] = \frac{h}{2}[f(x_{i-1}) + f(x_i)]$$

5. 误差分析 Error Analysis

a. 中点法和梯形法的误差 Error of Midpoint and Trapezoidal Method

$$\text{Midpoint Error} = \frac{f''(\xi_1)}{24}(b-a)^3 \ \ (\text{DOP} = 1)$$

$$\text{Trapezoidal Error} = -\frac{f''(\xi_2)}{12}(b-a)^3 \ \ (\text{DOP} = 1)$$

b. 复合中点法和复合梯形法的误差 Error of Composite Midpoint and Composite Trapezoidal Method: $||f''||_\infty$表示函数$f(x)$的二阶导数在区间$[a,b]$上的最大值；h表示子区间的最大长度。

$$\text{Error of Comp Trapezoidal Method} = |E(f)| \leq \frac{||f''||_\infty}{12}(b-a)h^2 \ \ (\text{DOP} = 1)$$

$$\text{Error of Comp Midpoint Method} = |E(f)| \leq \frac{||f''||_\infty}{24}(b-a)h^2 \ \ (\text{DOP} = 1)$$

c. 复合求积法总误差 Total Error of Composite Quadrature Methods: 过减小子区间长度h，可以降低总误差，因为误差随h^q减小。

6. 待定系数法 Method of Undetermined Coefficients
   a. 待定系数法的目的 Purpose of the Method of Undetermined Coefficients: 为了求出幂函数 Basis Function 为幂函数 Power Function 的情况下，Quadrature Methods 所需要的 Weight w_j。
   b. 待定系数法的过程 Steps for the Method of Undetermined Coefficients
      i. Step 1: 选择幂函数作为 Basis Function，例如$f(x) = 1, x, x^2, \dots, x^n$。
      ii. Step 2: 将每个幂函数代入求积公式，使其等于该幂函数的实际积分。这将为每个基函数产生一个方程。
      iii. Step 3: 通过解这些方程组，可以确定 Weight w_j的具体值。
   c. 案例 Example: What choice of the weights w_0, w_1, and w_2 maximizes the DOP of the rule $R(f) = w_0 f(-1) + w_1 f(0) + w_2 f(1)$ used to estimate the integral $I(f) = \int_{-1}^{1} f(x)\,dx$?
      i. Step 1: 选择$f(x) = 1, x, x^2, \dots, x^n$作为 Basis Function。
         - 0^{th}: $\int_{-1}^{1} 1\,dx = 2$
         - 1^{st}: $\int_{-1}^{1} x\,dx = 0$
         - 2^{nd}: $\int_{-1}^{1} x^2\,dx = \frac{2}{3}$
         - 3^{rd}: $\int_{-1}^{1} x^3\,dx = 0$
         - 4^{th}: $\int_{-1}^{1} x^4\,dx = \frac{2}{5}$
         - …
      ii. Step 2 & Step 3: 书写线性方程组，可以确定 Weight w_j的具体值。由于我们只需要w_0, w_1, and w_2，因此我们选择前 3 个式子构建 SLE 即可。

$$\int_{-1}^{1} 1\,dx = 2 = w_0(1) + w_1(1) + w_2(1)$$

$$\int_{-1}^{1} x\,dx = 0 = w_0(-1) + w_1(0) + w_2(1)$$

$$\int_{-1}^{1} x^2\,dx = \frac{2}{3} = w_0(-1)^2 + w_1(0)^2 + w_2(1)^2$$

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{2}{3} \end{bmatrix} \quad \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ \frac{1}{3} \end{bmatrix} \ (w_i \text{ is the weights.})$$

$$R(f) = \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1)$$

d. 如何通过待定系数法得出其精度 How to Derive the Degree of Precision by the Method of Undetermined Coefficients

i. 方法 Method: 需要找到在第几阶的时候精确值$I(f)$和估计值$R(f)$不等，此时其对应的点-1（即上一个可被精确估计的点）就是 Degree of Precision。

ii. 案例 Example: 对于之前的例子

$$\int_{-1}^{1} 1\,dx = 2 = \frac{1}{3}(1) + \frac{4}{3}(1) + \frac{1}{3}(1)$$

$$\int_{-1}^{1} x\,dx = 0 = \frac{1}{3}(-1) + \frac{4}{3}(0) + \frac{1}{3}(1)$$

$$\int_{-1}^{1} x^2\,dx = \frac{2}{3} = \frac{1}{3}(-1)^2 + \frac{4}{3}(0)^2 + \frac{1}{3}(1)^2$$

$$\int_{-1}^{1} x^3\,dx = 0 = \frac{1}{3}(-1)^3 + \frac{4}{3}(0)^3 + \frac{1}{3}(1)^3$$

$$\int_{-1}^{1} x^4\,dx = \frac{2}{5} \neq \frac{1}{3}(-1)^4 + \frac{4}{3}(0)^4 + \frac{1}{3}(1)^4$$

由于在第四阶的时候，精确值和估计值不等，精确值等于 2/5，估计值等于 2/3。此时，我们的 Degree of Precision 是 3。

7. 皮亚诺定理与积分误差 Peano's Theorem and the Error in Integration

a. 假设现在有一个 Quadrature Methods $R(f) = \sum_{i=0}^{N} w_i f(x_i)$来近似积分$I(f) = \int_a^b f(x)dx$，并且该规则的 Degree of Precision 为m。假设被积函数$f(x)$及其前$m+1$阶导数都在区间$[a,b]$上存在且连续。

b. Peano 核函数$K(x)$: 在这种情况下，存在一个与被积函数$f(x)$无关的函数$K(x)$，称为 Peano 核。这使得积分误差 Error in Integration 可以表示为

$$I(f) - R(f) = \int_a^b K(x) f^{(m+1)}(x)dx$$

如果 Peano 核$K(x)$在积分区间$[a,b]$上不变化，则可以进一步简化误差表达式，此时误差可以写作

$$I(f) - R(f) = \kappa f^{(m+1)}(\eta)$$

其中$\eta \in [a,b]$，且κ是一个与被积函数$f(x)$及其导数无关的常数。

c. 对于梯形法、中点法和辛普森法等常用积分规则（包括高斯和 Lobatto 积分法），我们都可以使用简化版本公式进行计算

$$I(f) - R(f) = \kappa f^{(m+1)}(\eta)$$

此时，使用定积分求解出来精确值，并且使用 Quadrature Rule 所得到的估计值，我们找到其差值。之后，我们使用 Degree of Precision + 1 作为导数的阶数，求得$f^{(m+1)}(\eta)$。之后，我们就可以找到κ。

d. 案例 Example: Calculate the Peano constant κ for Simpson's rule (DOP = 3)

$$R(f) = \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1)$$

对于这道题，我们需要先求解在$f(x) = x^4$时，精确值与估计值的差值

$$I(x^4) - R(x^4) = \frac{2}{5} - \frac{2}{3} = -\frac{4}{15}$$

之后，我们求解$f^{(m+1)}(\eta)$，在本题中$f^{(4)}(x) = 4! = 24$，因此我们可以得到

$$-\frac{4}{15} = 24\kappa \implies \kappa = -\frac{1}{90}$$

8. 高斯法则 Gauss Rules

a. 高斯法则的基本形式 Basic Form of Gauss Rules

$$I(f) := \int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{N} w_i f(x_i) := R(f)$$

如果我们想追求极高的精度，让 DOP 最大化。我们可以使用勒让德多项式 Legendre Polynomials 作为 Quadrature Rule 的 Basis Function，并且让 Legendre Polynomials 的零点（高斯点 Gauss Points）作为节点的 Quadrature Rule。这种方法被称为 Gauss Rules。

b. 高斯积分规则的特性 Properties of Gauss Rules

i. 所有 Weight $w_i > 0$。

ii. 所有 Nodes（高斯点 Gauss Points）都位于区间$(-1, +1)$。

iii. 对称性 Symmetry: Nodes x_i在原点周围对称分布，相应的 Weight w_i也对称。

iv. 节点交织 Interlacing of the Nodes: N点 Gauss Rule 的 Node x_i和$(N+1)$点 Gauss Rule 的 Node $\bar{x}_i$交错排列，满足关系$-1 < x_0 < \bar{x}_0 < x_1 < \bar{x}_1 < \cdots < x_N < +1$。

v. Gauss Rules 本质上是 Interpolatory Quadrature Rules，这说明在 Nodes 确定后，Weights 可以根据拉格朗日基函数的积分求得。

$$w_j = \int_{-1}^{1} L_j(x)dx$$

但是这种方法在实际中并不方便，我们可以得到更加简便的权重公式 Weighting Formula，具体见下文。

vi. $(N+1)$点的高斯规则的精度为$2N+1$。

c. 勒让德多项式 Legendre Polynomials

$$\phi_{j+1}(x)=\frac{2j+1}{j+1}x\phi_j(x)-\frac{j}{j+1}\phi_{j-1}(x), j\geq 1$$

i. 这是 Legendre Polynomials 的递推公式。每一个 Legendre Polynomials 都可以通过前两阶的多项式计算出来。

ii. Legendre Polynomials 是一系列多项式，定义在区间$[-1,1]$上。

iii. 基础的 Legendre Polynomials 是$\phi_0(x)=1$和$\phi_1(x)=x$。

iv. 前几个 Legendre Polynomials 如下

- $\phi_0(x)=1$
- $\phi_1(x)=x$
- $\phi_2(x)=\frac{1}{2}(3x^2-1)$
- $\phi_3(x)=\frac{1}{2}(5x^3-3x)$
- $\phi_4(x)=\frac{1}{8}(35x^4-30x^2+3)$

d. 高斯点 Gauss Points: 勒让德多项式 Legendre Polynomials 的零点。

i. 两点高斯积分 Two-Point Gauss Rule: Two-Point Gauss Rule 说明积分公式中我们使用 2 个 Nodes (Gauss Points)。并且我们应该使用 2 阶 Legendre Polynomial。在这里我们定义一个参数N，$N=2-1=1$。

ii. 三点高斯积分 Three-Point Gauss Rule: Three-Point Gauss Rule 说明积分公式中我们使用 3 个 Nodes (Gauss Points)，因此需要使用三阶 Legendre Polynomial，而参数$N=3-1=2$。

iii. 参数 N 的总结 Summary of Parameter N: 点数或者阶数减 1。

e. 权重公式 Weighting Formula: 对于每个节点x_j，Gauss Points 的权重w_j为

$$w_j=\frac{2\left(1-x_j^2\right)}{[(n+1)\phi_n(x_j)]^2}, j=0,1,\dots,n$$

f. 误差公式 Corresponding Error

$$\int_{-1}^{1}f(x)dx-\sum_{j=0}^{n}w_jf(x_j)=\frac{2^{2n+3}((n+1)!)^4}{(2n+3)!\,((2n+2)!)^2}f^{(2n+2)}(\xi)$$

这里的误差项是$f^{(2n+2)}(\xi)$，其中ξ是$[-1,1]$中的某个值，n是总点数，因此 Gauss Rule 在被积函数是多项式的时候精度非常高，能够在较小的节点下精确积分较高次的多项式。其精度 DOP 是$2N+1$。

g. 如何通过高斯法则近似积分 How to Approximate Integrals by Gauss Rules

i. Step 1: 根据题目要求是几点 Gauss Rule，判断需要使用几阶 Legendre Polynomials。阶数等于点数减一。并写出对应的 Legendre Polynomial 和解出 Gauss Points（Legendre Polynomial 的零点）。

ii. Step 2: 通过公式，计算相应的 Weights（有几个 Gauss Points，就有几个 Weights）。

iii. Step 3: 根据 Quadrature Rule 写出近似积。

h. 案例 Example: 假设在区间$[-1,1]$上近似计算积分$\int_{-1}^{1} f(x)\,dx$，我们选择一个简单的函数$f(x)=x^2$，并且我们使用 2 点高斯积分进行积分，即$N=1$。

i. Step 1: 对于 2-Point Gauss Rules，我们使用$N+1=2$阶的 Legendre Polynomials

$$\phi_2(x)=\frac{1}{2}(3x^2-1)=\frac{3}{2}x^2-\frac{1}{2}$$

我们通过解$\phi_2(x)=0$找到 Nodes

$$\frac{3}{2}x^2-\frac{1}{2}=0 \implies x^2=\frac{1}{3} \implies x=\pm\frac{\sqrt{3}}{3}$$

我们可以找到两个高斯点

$$x_0=-\frac{\sqrt{3}}{3}, x_1=\frac{\sqrt{3}}{3}$$

ii. Step 2: 计算相应的 Weights

$$w_j=\frac{2(1-x_j^2)}{[(n+1)\phi_n(x_j)]^2}$$

$$w_0=\frac{2(1-\frac{1}{3})}{[2\cdot(-\frac{\sqrt{3}}{3})]^2}=\frac{2\cdot\frac{2}{3}}{4\cdot\frac{1}{3}}=1$$

$$w_1=\frac{2(1-\frac{1}{3})}{[2\cdot\frac{\sqrt{3}}{3}]^2}=\frac{2\cdot\frac{2}{3}}{4\cdot\frac{1}{3}}=1$$

$$w_0=1, w_1=1$$

iii. Step 3: 根据 Quadrature Rule 写出近似积

$$\int_{-1}^{1} f(x)dx \approx \sum_{j=0}^{1} w_j f(x_j)=w_0 f(x_0)+w_1 f(x_1)$$

$$=1\cdot f\left(-\frac{\sqrt{3}}{3}\right)+1\cdot f\left(\frac{\sqrt{3}}{3}\right)=1\cdot\left(-\frac{\sqrt{3}}{3}\right)^2+1\cdot\left(\frac{\sqrt{3}}{3}\right)^2$$

$$=\frac{2}{3}$$

i. 表格《一点至六点高斯积分的权重和高斯点》Table “Weights and Gauss Points for Gauss Rules from One to Six Points

Points	Weighting Factors	Function Arguments	Truncation Error
1	$c_0 = 2$	$x_0 = 0.0$	$\cong f^{(2)}(\xi)$
2	$c_0 = 1$ $c_1 = 1$	$x_0 = -1/\sqrt{3}$ $x_1 = 1/\sqrt{3}$	$\cong f^{(4)}(\xi)$
3	$c_0 = 5/9$ $c_1 = 8/9$ $c_2 = 5/9$	$x_0 = -\sqrt{3/5}$ $x_1 = 0.0$ $x_2 = \sqrt{3/5}$	$\cong f^{(6)}(\xi)$
4	$c_0 = (18 - \sqrt{30})/36$ $c_1 = (18 + \sqrt{30})/36$ $c_2 = (18 + \sqrt{30})/36$ $c_3 = (18 - \sqrt{30})/36$	$x_0 = -\sqrt{525 + 70\sqrt{30}}/35$ $x_1 = -\sqrt{525 - 70\sqrt{30}}/35$ $x_2 = \sqrt{525 - 70\sqrt{30}}/35$ $x_3 = \sqrt{525 + 70\sqrt{30}}/35$	$\cong f^{(8)}(\xi)$
5	$c_0 = (322 - 13\sqrt{70})/900$ $c_1 = (322 + 13\sqrt{70})/900$ $c_2 = 128/225$ $c_3 = (322 + 13\sqrt{70})/900$ $c_4 = (322 - 13\sqrt{70})/900$	$x_0 = -\sqrt{245 + 14\sqrt{70}}/21$ $x_1 = -\sqrt{245 - 14\sqrt{70}}/21$ $x_2 = 0.0$ $x_3 = \sqrt{245 - 14\sqrt{70}}/21$ $x_4 = \sqrt{245 + 14\sqrt{70}}/21$	$\cong f^{(10)}(\xi)$
6	$c_0 = 0.1713244923791 70$ $c_1 = 0.360761573048139$ $c_2 = 0.467913934572691$ $c_3 = 0.467913934572691$ $c_4 = 0.360761573048131$ $c_5 = 0.1713244923791 70$	$x_0 = -0.932469514203152$ $x_1 = -0.661209386466265$ $x_2 = -0.238619186083197$ $x_3 = 0.238619186083197$ $x_4 = 0.661209386466265$ $x_5 = 0.932469514203152$	$\cong f^{(12)}(\xi)$

9. 龙贝格法则 Lobatto Rules
   a. 龙贝格法则和高斯法则的区别 Difference between Lobatto and Gauss Rules
      i. 在 Gauss Rule 中，我们默认使用 Gauss Points 作为 Nodes，其不包括积分区间的 Endpoints。但
      ii. 是在 Lobatto Rule 中，Nodes 也包括积分区间的 Endpoints（即包括$[-1,1]$的边界）。不管是 Gauss Rule 还是 Lobatto Rule，其 Nodes 都对称地分布在区间内。
      iii. Lobatto Rules 的 DOP 是$2N-1$。
   b. 龙贝格法则的基本形式 Basic Form of Lobatto Rules

$$I(f) := \int_{-1}^{1} f(x)dx \approx w_0 f(-1) + \sum_{i=1}^{N-1} w_i f(x_i) + w_N f(1) := R(f)$$

   其中，w_0和w_N是区间端点 Endpoints -1 和 1 的 Weights。$x_1, x_2, \dots, x_{N-1}$是区间内部的节点。$w_1, w_2, \dots, w_{N-1}$是区间内部节点的 Weights。
   c. 内部节点的选择 Selection of Internal Nodes: 除去 Endpoints 外，Lobatto Rules 区间内部是 Legendre Polynomial 的导数的零点。Legendre Polynomial 的阶数等于 Lobatto Rules 的点数减 1。比如对于 5 点 Lobatto Rules，我们需要使用 4 阶 Legendre Polynomial 的导数的零点作为 Internal Nodes。如下是前几个 Legendre Polynomial 的导数:
      i. $\phi_0'(x) = 0$
      ii. $\phi_1'(x) = 1$
      iii. $\phi_2'(x) = 3x$

iv. $\phi_3'(x) = \frac{1}{2}(15x^2 - 3)$

v. $\phi_4'(x) = \frac{1}{2}(5x(7x^2 - 3))$

因此，我们可以得到其零点，从而得到 Lobatto Rules 的 Internal Nodes。

n	x_i	x_i	w_i	w_i
3	0	0.00000	$\frac{4}{3}$	1.333333
	± 1	± 1.00000	$\frac{1}{3}$	0.333333
4	$\pm\frac{1}{5}\sqrt{5}$	± 0.447214	$\frac{5}{6}$	0.833333
	± 1	± 1.000000	$\frac{1}{6}$	0.166667
5	0	0.000000	$\frac{32}{45}$	0.711111
	$\pm\frac{1}{7}\sqrt{21}$	± 0.654654	$\frac{49}{90}$	0.544444
	± 1	± 1.000000	$\frac{1}{10}$	0.100000
6	$\sqrt{\frac{1}{21}(7-2\sqrt{7})}$	± 0.285232	$\frac{1}{30}(14+\sqrt{7})$	0.554858
	$\sqrt{\frac{1}{21}(7+2\sqrt{7})}$	± 0.765055	$\frac{1}{30}(14-\sqrt{7})$	0.378475
	± 1	± 1.000000	$\frac{1}{15}$	0.066667

d. 如何确定龙贝格法则的权重 How to Determine the Weights of Lobatto Rules

i. 概述 Overview: 我们使用 Method of Undetermined Coefficients 来寻求 Lobatto Rules 的 Weights，方法步骤和幂函数的 Method of Undetermined Coefficients 保持一致。

ii. 案例 Example: 我们使用 3 点 Lobatto Rule 来近似计算$[-1,1]$区间上积分$\int_{-1}^{1} f(x)\,dx$，选择的节点为$-1, 0, 1$，请求出节点对应的权重w_0, w_1, w_2。

- Step 1: 列出积分近似公式并设置待定系数

$$\int_{-1}^{1} f(x)dx \approx w_0 f(-1) + w_1 f(0) + w_2 f(1)$$

- Step 2: 代入条件，构建方程组

$$\int_{-1}^{1} 1dx = 2 = w_0(1) + w_1(1) + w_2(1)$$

$$\int_{-1}^{1} xdx = 0 = w_0(-1) + w_1(0) + w_2(1)$$

$$\int_{-1}^{1} x^2 dx = \frac{2}{3} = w_0(1) + w_1(0) + w_2(1)$$

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{2}{3} \end{bmatrix} \quad \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{4}{3} \\ \frac{1}{3} \end{bmatrix} \quad (w_i \text{ is the weights.})$$

$$\int_{-1}^{1} x^3\,dx = 0 = \frac{1}{3}(-1)^3 + \frac{4}{3}(0)^3 + \frac{1}{3}(1)^3$$

$$\int_{-1}^{1} x^4\,dx = \frac{2}{5} \neq \frac{1}{3}(-1)^4 + \frac{4}{3}(0)^4 + \frac{1}{3}(1)^4$$

由于在第四阶的时候，精确值和估计值不等，精确值等于 2/5，估计值等于 2/3。此时，我们的 Degree of Precision 是 3。这和我们的$2n-1$的公式相符，因为有 3 个点，n为 2，DOP 应该是 3。

10. 到标准区间的转换 Transformation to the Canonical Interval
    a. 标准区间 Canonical Interval: 标准区间 Canonical Interval 是数值分析和数值积分中的一个固定的参考区间，通常是$[-1,1]$，这样可以简化计算和分析。在高斯定理 Gauss Rule、龙贝格法则 Lobatto Rules、勒让德多项式 Legendre Polynomials 以及切比雪夫多项式 Chebyshev Polynomials 中，Canonical Interval 是$[-1,1]$。
    b. 如何将积分区间转换到标准区间 How to Convert Integral Interval to Canonical Interval
        i. 区间转换 Interval Conversion: 如果我们遇到非$[-1,1]$标准区间的情况，而是遇到了一个任意的$[a,b]$区间。我们可以使用如下公式将积分的上下限进行改变。

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right)\frac{b-a}{2}dt$$

        ii. 节点转换 Node Conversion: 我们可以将标准区间 Canonical Interval 的 Nodes x_j^*转化为区间$[a,b]$上的 Nodes x_j。

$$x_j = \frac{b-a}{2}x_j^* + \frac{b+a}{2}$$

        $\frac{b-a}{2}$是缩放因子 Scaling Factor；$\frac{b+a}{2}$是平移因子 Translation Factor。
        iii. 权重转换 Weight Conversion: 我们可以将标准去见 Canonical Interval 的 Weights w_j^*转化为区间$[a,b]$上的 Weights w_j。

$$w_j = \frac{b-a}{2}w_j^*$$

        $\frac{b-a}{2}$是缩放因子 Scaling Factor。
11. 精度 Degree of Precision

a. 精度的定义 Definition of Degree of Precision (DOP): 一个数值积分公式能够精确积分的多项式的最高次数。具体而言，如果一个数值积分方法能够精确积分所有次数小于或等于n的多项式，但对次数为$n+1$的多项式则不一定精准，那么这个方法的 DOP 等于n。
b. 精度规律总结 Summary of DOP
    i. 方法 Method: 误差项中最高导数次数减一确定 DOP。
    ii. 精度对比 DOP Comparison

Method Name	**Error Function**	**Derivative**	**DOP**
Midpoint	$\frac{f''(\xi_1)}{24}(b-a)^3$	2nd Order	1
Trapezoidal	$-\frac{f''(\xi_2)}{12}(b-a)^3$	2nd Order	1
Comp Midpoint	$\frac{\|\|f''\|\|\infty}{12}(b-a)h^2$	2nd Order	1
Comp Trapezoidal	$\frac{\|\|f''\|\|\infty}{24}(b-a)h^2$	2nd Order	1
Simpson's Rule	$-\frac{f^{(4)}(\xi_3)}{90}\left(\frac{b-a}{2}\right)^5$	4th Order	3
Gauss Rules	$\frac{2^{2n+3}((n+1)!)^4}{(2n+3)!((2n+2)!)^2}f^{(2n+2)}(\xi)$	2n+2 Order	2n+1
Lobatto Rules	$-C\cdot(b-a)^{2n}f^{(2n)}(\xi)$	2n Order	2n-1

第六单元 根的寻找 Chapter 6 Root Finding

求解非线性方程的根的数值方法 Numerical Methods for Roots of Nonlinear Equations

1. 寻找非线性方程根的五种方法 Five Ways to Find Roots of Nonlinear Equations
    a. 固定点迭代法 Fixed-Point Iteration
    b. 牛顿迭代法 Newton Iteration
    c. 割线法 Secant Iteration
    d. 二分法 Bisection
    e. 二次反插值法 Quadratic Inverse Interpolation
2. 每种求根法的重要关注点 Important Concerns for Each Rooting Method
    a. 收敛速率 Convergence Rate
        i. 定义 Definition: 指方法达到期望精度 Desired Accuracy 所需的迭代次数 Number of Iterations。

$$\lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^p}=K$$

        其中
        - $e_n = x_n - x^*$是误差。
        - p是收敛速率 Convergence Rate。
        - $K > 0$是常数。

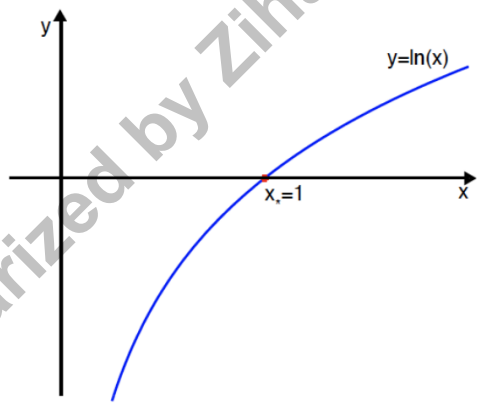
        ii. 分类 Classification
        - 线性收敛 Linear Convergence $p = 1$: 误差按固定比例减少（如 Fixed-Point Iteration、Bisection）。
        - 超线性收敛 Superlinear Convergence $p > 1$且非整数: 误差减少速度会逐步加快，速度介于线性与二次收敛之间（如 Secant Iteration、Quadratic Inverse Interpolation）。
        - 二次收敛 Quadratic Convergence $p = 2$: 误差平方减少（如 Newton Iteration）。

    b. 是否需要导数 Need for Derivatives
        i. 定义 Definition: 方法是否需要计算或依赖函数的导数 Derivative of A Function。
        ii. 关注点 Concerns
        - 如果导数显式表达式难以获得，或计算代价高，则优先选择不需要导数的方法（如 Secant Iteration、Bisection）。
        - 如果导数易得或可以通过自动微分计算，则可以使用 Newton Iteration 等需要导数的方法。

    c. 初始条件的选择 Selection of Initial Conditions

i. 定义 Definition: 方法对初始值或初始区间 Initial Value or Initial Interval 的要求。
ii. 关注点 Concerns
- 二分法 Bisection: 需要初始区间$[a, b]$，并要求$f(a)f(b) < 0$。
- 固定点迭代法 Fixed-Point Iteration: 需要选择 Initial Value x_0足够接近固定点 Fixed Point，否则可能不收敛。
- 牛顿法和割线法 Newton Iteration and Secant Iteration: 对 Initial Value 较敏感，Initial Value 距离 Roots 较远时可能不收敛。

3. 简单根 Simple Root
   a. 定义 Definition: 满足如下三个条件被称为 Simple Root
      i. Condition 1: $f(x^*) = 0$ 函数在x^*处的值为 0，表示x^*是方程的根。
      ii. Condition 2: $f'(x)$在包含x^*的某个开区间内存在，保证函数$f(x)$在这一区域是 Continuous and Differentiable。
      iii. Condition 3: $f'(x^*) \neq 0$ 简单根处导数不为 0。
   b. 几何意义 Geometric Significance: 函数$f(x)$在 Simple Root x^*处穿过x轴。



   c. 本章讨论的范畴 Scope of the Chapter: 通过本章的方法，我们可以确定 Simple Root。

固定点迭代法 Fixed-Point Iteration

1. 根与固定点的关系 Root Finding and Fixed Point
   a. 固定点 Fixed Point: 要求解$x^* = g(x^*)$，即找到使输入值等于输出值的点x^*。
   b. 根问题 Root Finding: 要求解$f(x^*) = 0$，即找到使函数值为 0 的点x^*。
   c. 代数变换 Algebraic Transformation: 为了将根问题 Root Finding $f(x^*) = 0$转化为固定点问题 Fixed Point $x^* = g(x^*)$，可以代数地重新定义一个函数$g(x)$。
2. 如何将根问题转化为固定点问题 How to Convert the Root Finding Problem into the Fixed Point Problem

a. 方法与案例 Methods and Examples
    i. Method 1: 直接分离变量，例如$f(x) = x^2 - 3x + 2 = 0$，我们重新排列可以得到$x = \frac{x^2+2}{3}$，这是一个可能的$g(x) = \frac{x^2+2}{3}$。
    ii. Method 2: 通过$x = x - f(x)$构造，例如对于$f(x) = x^2 - 3x + 2 = 0$，我们可以说$g(x) = x - (x^2 - 3x + 2) = -x^2 + 4x - 2$。
    iii. Method 3: 通过线性变换调整方程，有时我们可以通过重新排列方程找到更简单的形式，比如直接将$f(x)$的一部分移到等式另一边。例如$f(x) = x^2 - 3x + 2 = 0$，我们重新排列可以得到$g(x) = \sqrt{3x - 2}$。
b. 遵循的原则 Principles to Be Followed
    i. 等价性 Equivalence: 转化后的 Fixed Point Problem $x = g(x)$的 Fixed Point x^*必须与原先的 Root Finding Problem 的$f(x) = 0$的根一致。也就是说，如果x^*是$g(x) = x$的解，其必须也是$f(x) = 0$的解，反之亦然。本质上，我们是把$f(x)$与x轴交点的寻找问题转化为$g(x)$与$y = x$交点的寻找问题。
    ii. 收敛性 Convergence: 构造的$g(x)$必须在根x^*的邻域内满足固定点迭代的收敛性条件
    $$|g'(x)| < 1$$
    iii. 连续性 Continuity: $g(x)$不唯一，可以有多种构造形式，但必须在目标根附近连续。

3. 固定点迭代法算法 Fixed-Point Iteration Algorithm
a. 目标 Objective: 通过迭代 Iteration 找到满足$x = g(x)$的 Fixed Point x^*。
b. 步骤 Steps
    i. 选择一个函数$g(x)$，使得 Root Finding Problem $f(x) = 0$等价于 Fixed Point Problem $g(x) = x$。
    ii. 猜测一个 Initial Value x_0。
    iii. 使用迭代公式 Iteration Formula
    $$x_{k+1} = g(x_k), k = 0,1, \dots$$
c. 停止迭代的条件 Conditions for Stopping Iteration
    i. 如果达到预设的误差容限 Error Tolerance，停止迭代。
    $$|x_{k+1} - x_k| < \epsilon$$
    ii. 如果达到最大迭代次数，停止迭代。
d. 案例 Example: 给出一个方程$f(x) = x - e^{-x} = 0$，使用 Fixed-Point Iteration 寻找其根。
    i. Step 1: 将 Root Finding Problem 转化为 Fixed Point Problem
    $$x^* = g(x^*) = e^{-x^*}$$

ii. Step 2: 判断$g(x)$是否满足 Convergence

$$g'(x^*) = |-e^{-x}|$$

其在根x^*的邻域内（即$x \in [0,1]$之间小于 1），说明其满足 Convergence。

iii. Step 3: 选择一个 Initial Value $x_0 = 0.5$。

iv. Step 4: 使用 Iteration Formula

$$\begin{cases} k = 0, x_1 = g(x_0) = e^{-0.5000} \approx 0.6065 \\ k = 1, x_2 = g(x_1) = e^{-0.6065} \approx 0.5452 \\ k = 2, x_3 = g(x_2) = e^{-0.5452} \approx 0.5797 \\ k = 3, x_4 = g(x_3) = e^{-0.5797} \approx 0.5606 \end{cases}$$

迭代继续进行，其逐渐逼近固定点x^*，也就是$f(x)$的根。

4. 固定点迭代的收敛性条件 Convergence Conditions for Fixed-Point Iterations

a. 收敛性条件 Convergence Conditions

i. 固定点条件 Fixed Point Condition: $x^* = g(x^*)$，即x^*是$g(x)$的 Fixed Point。

ii. 误差序列的单调减少 Monotonic Reduction of the Error Series: 我们定义误差为$e_n = |x_n - x^*|$，即当前迭代值与 Fixed Point 的差。我们要求$|e_n|$是单调减小 Monotonic Reduction 的，且随着迭代$n \to \infty$收敛到 0。

b. 误差传播公式 Error Propagation Formula

$$e_{n+1} = x_{n+1} - x^* = g(x_n) - g(x^*) = g'(\eta_n)(x_n - x^*) = g'(\eta_n)e_n$$

其中η_n是位于x_n和x^*之间的某一点。也就是说，下一步的误差e_{n+1}是由当前误差e_n和$g'(\eta_n)$的乘积决定的。这也说明如果

$$|g'(\eta_n)| < 1$$

误差会逐步减小，迭代会收敛。

c. 根问题转化为固定点问题的收敛性原则 Convergence Principle for the Transformation of Root Finding Problems into Fixed Point Problems

i. 收敛性 Convergence: 构造的$g(x)$必须在根x^*的邻域内满足固定点迭代的收敛性条件

$$|g'(\eta_n)| < 1$$

ii. 如何判断收敛性 How to Determine Convergence

- Method 1: 取η_1为x_1附近的值，来判断其绝对值。以我们之前的例子为例

$$g'(\eta_1) = -e^{-\eta_1} \approx -e^{-0.6065} \approx -0.5452$$

$$|g'(\eta_1)| \approx 0.5452 < 1$$

- Method 2: 使用画图计算器，画出对应$|g'(x)|$图像，看其是否在根x^*的邻域是否小于 1。

- Method 3: 使用中值定理 Mean Value Theorem 计算$g'(\eta_n)$。MVT 说明，对于 Continuous and Differentiable Functions $g(x)$，在区间$[x_n, x^*]$内存在一个$\eta_n \in (x_n, x^*)$，使得

$$|g'(\eta_n)| = \left|\frac{g(x_n) - g(x^*)}{x_n - x^*}\right| = \left|\frac{g(x_n) - x^*}{x_n - x^*}\right| < 1$$

以我们之前的例子为例，设$x^* \approx 0.5671$是固定点，$x_n = 0.6065$（即第 1 次迭代的值），我们可以得到

$$g'(\eta_n) = \frac{g(x_n) - x^*}{x_n - x^*} = \frac{0.5452 - 0.5671}{0.6065 - 0.5671}$$

因此可以得到

$$|g'(\eta_n)| < 1$$

5. 固定点的存在性和唯一性 Existence and Uniqueness of Fixed Point
    a. 存在性 Existence: 假设$g(x)$是连续的，并且在区间$[a, b]$上满足$g(a) > a, g(b) < b$，则存在固定点x^*。
        i. 定理证明 Proof of Theorem: 定义函数$\phi(x) = g(x) - x$。当$\phi(a) > 0$且$\phi(b) < 0$时，由连续性和介值定理 Intermediate Value Theorem，必定存在$x^* \in (a, b)$，使得$\phi(x^*) = g(x^*) - x^* = 0$。这说明$g(x) = x$有解，即存在 Fixed Point x^*。
        ii. 结论 Conclusion: 介值定理 Intermediate Value Theorem 保证了在$[a, b]$上$g(x)$必定有一个 Fixed Point。
    b. 唯一性 Uniqueness: 假设$g(x)$在$[a, b]$内可微 Differentiable，并且满足$|g'(x)| \le G < 1, \forall x \in [a, b]$。
        i. 定理证明 Proof of Theorem: 如果有两个 Fixed Points x^*, y^*，即$g(x^*) = x^*$且$g(y^*) = y^*$，可以使用中值定理 Mean Value Theorem 证明其不可能同时存在。根据 MVT，我们可以得到

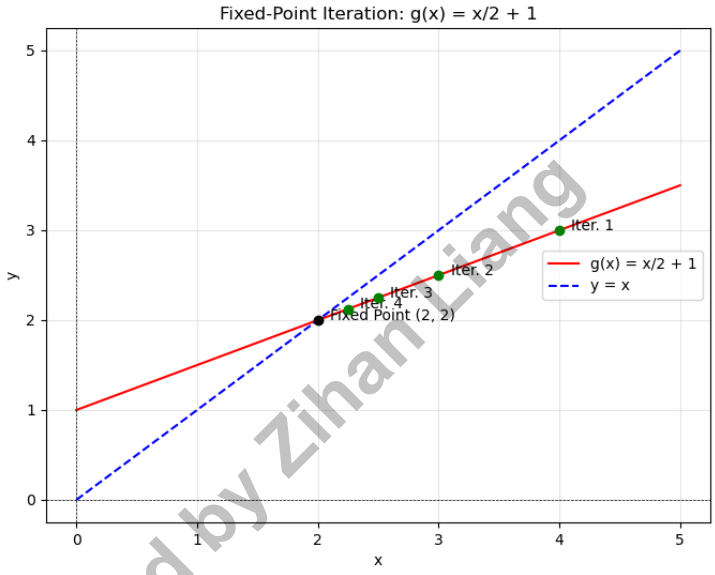
$$|x^* - y^*| = |g(x^*) - g(y^*)| = |g'(\xi)||x^* - y^*|, \xi \in (x^*, y^*)$$

由于$|g'(\xi)| \le G < 1$，则$|x^* - y^*| \le G|x^* - y^*| < |x^* - y^*|$。这只能成立在$|x^* - y^*| = 0$，即$x^* = y^*$。
        ii. 结论 Conclusion: 如果$|g'(x)| < 1$，则 Fixed Point x^*在$[a, b]$内是唯一的。
6. 固定点迭代的收敛速率 Convergence Rate of the Fixed-Point Iterations
    a. 收敛速率与$g'(x^*)$的关系 Convergence Rate Versus $g'(x^*)$
        i. 如果$|g'(x^*)|$越小，收敛越快。
        ii. 如果$g'(x^*) = 0$，则其可能达到 Superlinear 或者 Quadratic Convergence。
        iii. 如果$g'(x^*)$接近 1，收敛速度非常慢。

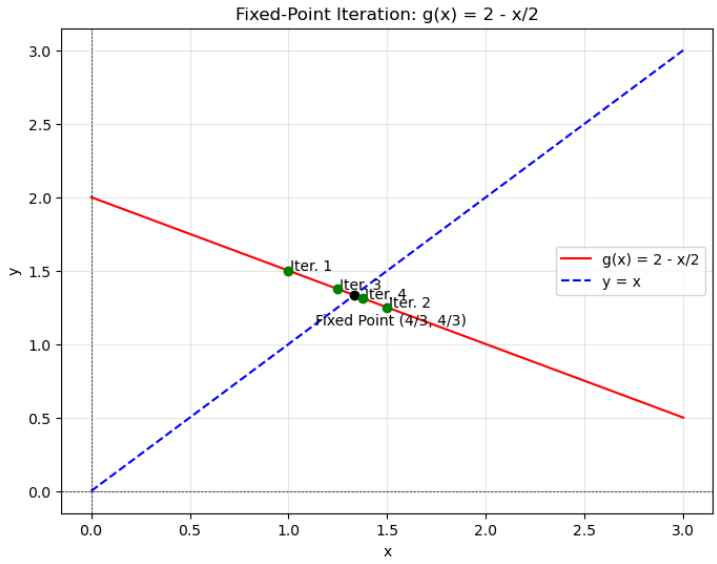
iv. 如果$|g'(x^*)| \geq 1$，迭代发散（不满足收敛性条件 Convergence Conditions）。

b. 单侧收敛与双侧收敛 One-Sided and Two-Sided Convergence

i. 当$g'(x^*) > 0$，误差不会变号，称为单侧收敛 One-Sided Convergence。以$g(x) = \frac{x}{2} + 1$为例，其$g'(x) = \frac{1}{2} > 0$。在图中，我们可以看到，其都在基准线$y = x$的一侧收敛（每一次迭代都在$y = x$的一侧），因此称为 One-Sided Convergence。



ii. 当$g'(x^*) < 0$，误差会在每次迭代中交替变号，称为双侧收敛 Two-Sided Convergence。以$g(x) = 2 - \frac{x}{2}$为例，其$g'(x) = -\frac{1}{2} < 0$。在图中，我们可以看到，其在基准线$y = x$的两侧收敛（每一次迭代在$y = x$不同的一侧），因此称为 Two-Sided Convergence。



c. 固定点迭代的线性与超线性 Fixed-Point Iteration's Linearity and Superlinearity
   i. 如果$g'(x^*) = 0$且$|g'(x^*)| < 1$（收敛性条件 Convergence Conditions）时，误差主导项时高阶项，收敛速度为 Superlinear。
   ii. 如果$g'(x^*) \neq 0$，误差主导项时线性项，收敛速度为 Linear。一般情况下，Fixed-Point Iterations 是线性收敛 Linear Convergence。特殊情况下可能是超线性收敛 Superlinear Convergence，即$g(x)$等价于牛顿法的时候。

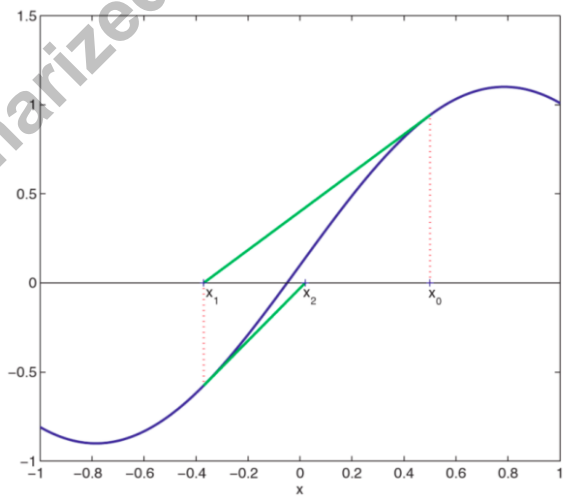
牛顿迭代法 The Newton Iteration

1. 牛顿迭代法的假设条件 Conditions for the Newton Iteration
   a. Condition 1: $f \in C^2[a,b]$，即$f(x)$的一阶和二阶导数存在并连续。
   b. Condition 2: $f'(x) \neq 0$，避免分母为 0 的情况。
2. 泰勒定理与牛顿迭代法算法 Taylor's Theorem and Newton Iteration Algorithm
   a. 泰勒定理 Taylor's Theorem: 通过对函数$f(x)$在点x_k附近进行泰勒展开

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{f''(\xi)(x - x_k)^2}{2}$$

   b. 牛顿迭代法算法 Newton Iteration Algorithm: 使用切线方程（线性近似）的方式代替原始函数，求切线与x轴的交点。

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$



   i. 图中紫色曲线代表目标函数$f(x)$，我们需要求解的是$f(x) = 0$。
   ii. Initial Value x_0是 Newton Iteration 的起始点。
   iii. 绿色线段表示点x_0处$f(x)$的切线 Tangent Line，该 Tangent Line 的交点x_1是下一步迭代的结果。
   iv. 迭代过程重复，将x_1的 Tangent Line 交点计算为x_2，逐步逼近根。
   d. 案例 Example: 求解使用 Newton Iteration 找到$f(x) = x^2 - 2 = 0$的根。

i. Step 1: 选择一个 Initial Value $x_0 = 1$。

ii. Step 2: 求解$f'(x)$，写出相关 Newton Iteration 公式。

$$f'(x) = 2x$$

$$x_{k+1} = x_k - \frac{x_k^2 - 2}{2x_k}$$

iii. Step 3: 迭代计算。

- 第一次迭代

$$x_1 = 1 - \frac{1^2 - 2}{2 \times 1} = 1.5$$

- 第二次迭代

$$x_2 = 1.5 - \frac{1.5^2 - 2}{2 \times 1.5} = 1.4167$$

- 第三次迭代

$$x_3 = 1.4167 - \frac{1.4167^2 - 2}{2 \times 1.4167} = 1.4142$$

迭代继续进行，其逐渐逼近$f(x)$的根。

3. 牛顿迭代法的收敛性与特性 Convergence and Properties of the Newton Iteration

a. 收敛条件 Convergence Conditions

i. 起始点的选择 Selection of Initial Value: Newton Iteration 的 Initial Value x_0需要足够接近函数根x^*。

ii. 原函数的性质 Properties of the Original Function: $f'(x)$在包含函数根x^*的闭区间上是连续的。

b. 收敛速度 Convergence Rate: 当 Error e_k足够小的时候，Newton Iteration 的 Error 会快速减小。每一次迭代，近似解x_k的正确有效数字数目大约翻倍（即达到 Quadratic Convergence）。

c. 单侧收敛 One-Sided Convergence: 当 Error e_k足够小的时候，迭代会从单侧逼近函数根，并且 Error Function 会准确描述误差的行为。

e. 是否需要导数 Need for Derivatives: 牛顿法 Newton Iteration 需要计算导数。对于一些复杂的函数，计算导数$f'(x_k)$可能是困难的，或者需要额外的计算成本。为了解决该问题，我们使用割线法 Secant Iteration 来近似导数代替牛顿法 Newton Iteration 的精准导数。

割线法 Secant Iteration

1. 割线法的假设条件 Conditions for the Secant Iteration

a. Condition 1: 割线法 Secant Iteration 要求目标函数$f(x)$在定义域内连续，以确保函数值$f(x_k)$和$f(x_{k-1})$有意义。

b. Condition 2: 对于每一次迭代，我们要求$f(x_k) - f(x_{k-1}) \neq 0$，否则分母为 0，无法进行迭代。
c. Condition 3: 虽然割线法 Secant Iteration 不需要显式地计算导数$f'(x)$，但隐含假设函数$f(x)$在根x^*附近具有良好的线性近似性 Good Linear Approximability，即$f'(x) \neq 0$且在根附近平滑。

2. 割线法算法 Secant Iteration Algorithm
a. 割线法的迭代公式 Secant Iteration Formula

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

通过最近的两个点x_k和x_{k-1}的函数值来近似导数$f'(x_k)$，避免直接计算导数。Secant Iteration 需要两个 Initial Values 才能够开始迭代。
b. 案例 Example: 求解使用 Secant Iteration 找到$f(x) = x^2 - 2 = 0$的根。
i. Step 1: 选择两个 Initial Values $x_0 = 1, x_1 = 2$。
ii. Step 2: 迭代计算。
- 第一次迭代: 计算x_2

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 2 - \frac{2(2-1)}{2-(-1)} = 1.3333$$

- 第二次迭代: 计算x_3

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} = 1.3333 - \frac{-0.2222(1.3333 - 2)}{-0.2222 - 2} = 1.4$$

- 第三次迭代: 计算x_4

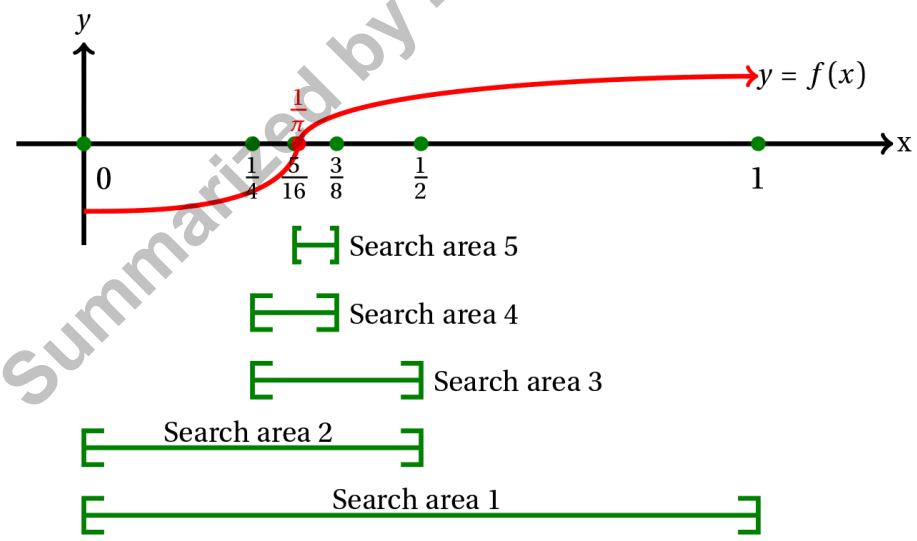
$$x_4 = x_3 - \frac{f(x_3)(x_3 - x_2)}{f(x_3) - f(x_2)} = 1.4 - \frac{-0.04(1.4 - 1.3333)}{-0.04 - (-0.2222)} \approx 1.4142$$

迭代继续进行，其逐渐逼近$f(x)$的根。

3. 割线法的收敛性与特性 Convergence and Properties of the Secant Iteration
a. 收敛条件 Convergence Condition: 如果 Initial Values x_0和x_1足够接近根x^*，Secant Iteration 保证收敛到x^*。
b. 收敛速度 Convergence Rate: Secant Iteration 的 Convergence Rate 是 Superlinear 的，其高于 Linear Convergence，但低于 Newton Iteration 的 Quadratic Convergence。具体而言，$p = \frac{1+\sqrt{5}}{2} \approx 1.618$，即黄金比例 Golden Ratio。
c. Secant Iteration 是 Newton Iteration 的一个变种，避免了直接计算导数的困难，同时提供了较快的收敛速度，但是其没有 Newton Iteration 收敛速度和精度高。

二分法 Bisection Method

1. 二分法的假设条件 Conditions for the Bisection Method: 介值定理 Intermediate Value Theorem 是二分法 Bisection Method 的理论基础。
    a. Condition 1: $f(x)$必须在区间$[x_l, x_u]$上是连续的 Continuous。
    b. Condition 2: 区间端点的函数值$f(x_l)$和$f(x_u)$的符号必须相反
    $$f(x_l)f(x_u) < 0$$
    这意味着区间内存在至少一个 Real Root。
2. 二分法算法 Bisection Method Algorithm
    a. 二分法算法内容 Bisection Method Algorithm Content
        i. Step 1: 划分区间。
            - 给定一个 Initial Interval $[x_l, x_u]$，其中$f(x_l)f(x_u) < 0$。
            - 将区间对半分，计算中点$x_m = \frac{x_l + x_u}{2}$。
        ii. Step 2: 符号变化判断，我们应该评估中点的函数值$f(x_m)$。
            - 如果$f(x_m) = 0$，则x_m是根，迭代结束。
            - 如果$f(x_m)f(x_l) < 0$，则根位于$[x_l, x_m]$，永远取穿过区间作为新的搜索区间 Search Interval。
            - 否则，根位于$[x_m, x_u]$。
    b. 案例 Example



        i. 给定区间[0,1]之间，取中点1/2，发现中点对应函数值为正。穿过区间位于$[0, 1/2]$，这是新的搜索区间 Search Interval。
        ii. 给定区间$[0, 1/2]$之间，取中点1/4，发现中点对应函数值为负。穿过区间位于$[1/4, 1/2]$，这是新的搜索区间 Search Interval。
        iii. 给定区间$[1/4, 1/2]$，取中点3/8，发现中点对应函数值为正。穿过区间位于$[1/4, 3/8]$，这是新的搜索区间 Search Interval。

iv. 给定区间$[1/4, 3/8]$，取中点$5/16$，发现中点对应函数值为负。穿过区间位于$[5/16, 3/8]$，这是新的搜索区间 Search Interval。

v. 以此类推，真实值为$1/\pi$。

3. 停止迭代的条件 Conditions for Stopping Iteration

a. 真实误差 True Error

$$\varepsilon_t = \left| x_{\text{approx}} - x_{\text{true}} \right|$$

i. x_{approx}: 当前近似根。

ii. x_{true}: 真实根。

iii. 真实误差在实际应用中通常无法直接计算，因为x_{true}未知。

b. 近似误差 Approximated Error

$$\varepsilon_a = \left| \frac{x_r^{\text{new}} - x_r^{\text{old}}}{x_r^{\text{new}}} \right| \times 100\%$$

i. x_r^{new}: 最新的中点值。

ii. x_r^{old}: 上一次的中点值。

iii. 近似误差可计算，是终止条件的实际依据。

c. 近似误差作为真实误差的上界 Approximation Error As an Upper Bound on the True Error: 在 Bisection Method 中，True Error 总是小于等于 Approximated Error。因此，Approximated Error 可以作为一个 Safe Upper Bound，用于判断迭代是否精准。

d. 根的范围估计 Estimated Range of Roots

i. 近似根 Approximate Root: 当前搜索区间的中点

$$x_r = \frac{x_l + x_u}{2}$$

ii. 真实根的位置 Location of the True Root: True Root x^*满足$f(x^*) = 0$，其必然位于$[x_l, x_u]$内。并且，True Root 与当前中点x_r（即近似根 Approximate Root）的距离不会超过区间长度Δx的一半。

$$\text{Error of the true root} \leq \frac{\Delta x}{2}$$

假设我们在第n次迭代中，区间长度为$\Delta x = x_u - x_l = 0.1$，则真实根与当前中点的误差最大值为$\Delta x/2 = 0.05$。

e. 终止条件 Termination Condition: ε_t或ε_a小于预设的误差容限 Error Tolerance，停止迭代。

f. 二分法的整体误差 Overall Error in Bisection Method

$$\frac{b-a}{2^k} < \epsilon$$

k是迭代次数。该公式可以计算一共需要迭代多少次达到我们的误差容限 Error Tolerance。

4. 二分法的收敛性与特性 Convergence and Properties of the Bisection Method
    a. 收敛条件 Convergence Condition: 如果区间两端函数值符号相反、真实根位于初始区间内、没有无限震荡或跳跃（没有奇点或间断点）则保证收敛。
    b. 收敛速度 Convergence Rate: Bisection 的 Convergence Rate 是 Linear 的。

二次反插值法 Quadratic Inverse Interpolation

1. 反函数与根问题 Inverse Function and Root Finding Problem
    a. 反函数的概念 Concept of Inverse Function: 如果f有反函数 Inverse Function f^{-1}，则$f(x) = y \Leftrightarrow x = f^{-1}(y)$。如果我们希望找到$f(x) = 0$的根$x^*$，其可以表示为
    $$x^* = f^{-1}(0)$$
    b. 二次反插值法理念 Quadratic Inverse Interpolation Philosophy: 在该方法中，我们可以使用多项式近似反函数 Inverse Function，我们可以使用二次多项式$p_2(y)$来近似$f^{-1}(y)$。然后找到多项式的根，即解$p_2(0) = 0$，即找到近似的反函数在$y = 0$处的根，从而得到根x^*的近似值。
2. 二次反插值法的假设条件 Conditions for the Quadratic Inverse Interpolation
    a. Condition 1: $f(x)$必须在根附近是连续的 Continuous。
    b. Condition 2: 需要选择三个 Initial Values $(x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}), (x_n, y_n)$，且$y_{n-2}$，$y_{n-1}, y_n$不能相等。
    c. Condition 3: 初始的三个点对应的函数值所构成的区间中需要包括$y = 0$的解，这意味着函数在三个点之间发生了符号变化。
    d. Condition 4: 函数$f(x)$在初始的三个点对应的函数值所构成的区间范围内是单调的 Monotonic，或者近似单调 Approximately Monotonic。
3. 二次反插值法算法 Quadratic Inverse Interpolation Algorithm
    a. 算法内容 Algorithm Content
        i. Step 1: 输入数据，假设给定三个点 Initial Values x_{n-2}, x_{n-1}, x_n，以及对应的函数值$f(x_{n-2}) = y_{n-2}, f(x_{n-1}) = y_{n-1}, f(x_n) = y_n$。
        ii. Step 2: 寻找反函数。
        $$x_{n-2} = f^{-1}(y_{n-2}), x_{n-1} = f^{-1}(y_{n-1}), x_n = f^{-1}(y_n)$$
        iii. Step 3: 构造反插值函数 Inverse Interpolation。我们利用拉格朗日插值法 Lagrange Interpolation，构造一个通过这些点的插值多项式。
        $$p_2(y) = \sum_{k=n-2}^{n} L_k(y) f^{-1}(y_k)$$
        $$L_k(y) = \prod_{j \neq k} \frac{y - y_j}{y_k - y_j}$$
        对于三个点$(y_{n-2}, x_{n-2}), (y_{n-1}, x_{n-1}), (y_n, x_n)$，插值函数$p_2(y)$展开为

$$p_2(y) = \frac{(y-y_{n-1})(y-y_{n-2})}{(y_n-y_{n-1})(y_n-y_{n-2})} f^{-1}(y_n) + \frac{(y-y_n)(y-y_{n-2})}{(y_{n-1}-y_n)(y_{n-1}-y_{n-2})} f^{-1}(y_{n-1}) + \frac{(y-y_n)(y-y_{n-1})}{(y_{n-2}-y_n)(y_{n-2}-y_{n-1})} f^{-1}(y_{n-2})$$

iv. Step 4: 寻找根。将$y=0$代入上式，求得插值多项式的根，得到近似的$f(x)=0$的解。

$$x_{n+1} = p_2(0) = \frac{y_{n-1}y_{n-2}}{(y_n-y_{n-1})(y_n-y_{n-2})} x_n + \frac{y_n y_{n-2}}{(y_{n-1}-y_n)(y_{n-1}-y_{n-2})} x_{n-1} + \frac{y_n y_{n-1}}{(y_{n-2}-y_n)(y_{n-2}-y_{n-1})} x_{n-2}$$

v. Step 5: 继续迭代。使用刚刚求出的x_{n+1}和之前的x_n, x_{n-1}继续迭代。

b. 案例 Example: 使用 Quadratic Inverse Interpolation 求解$f(x)=x^3-x-2=0$。

i. Step 1: 三个 Initial Values 假设为$x_0=1, x_1=2, x_2=1.5$，其对应的三个函数值为:

$$f(x_0)=f(1)=-2, f(x_1)=f(2)=4, f(x_2)=f(1.5)=-0.125$$

ii. Step 2: 寻找反函数。

$$x_0 = f^{-1}(y_0) \Rightarrow 1 = f(-2)$$
$$x_1 = f^{-1}(y_1) \Rightarrow 2 = f(4)$$
$$x_2 = f^{-1}(y_2) \Rightarrow 1.5 = f(-0.125)$$

iii. Step 3: 构造反插值函数 Inverse Interpolation。

$$p_2(y) = \frac{(y-4)(y+0.125)}{(-2-4)(-2+0.125)} \cdot 1 + \frac{(y+2)(y+0.125)}{(4+2)(4-0.125)} \cdot 2 + \frac{(y+2)(y-4)}{(0.125+2)(0.125-4)} \cdot 1.5$$

iv. Step 4: 寻找根。

$$x_3 = p_2(0) = \frac{(0-4)(0+0.125)}{(-2-4)(-2+0.125)} \cdot 1 + \frac{(0+2)(0+0.125)}{(4+2)(4-0.125)} \cdot 2 + \frac{(0+2)(0-4)}{(0.125+2)(0.125-4)} \cdot 1.5 \approx 1.5273$$

v. Step 5: 可以选择继续使用x_1, x_2, x_3继续迭代出x_4。

5. 二次反插值法的收敛速度 Convergence Rate: Quadratic Inverse Interpolation 的收敛速度是 Superlinear 的，其$p=1.44$。

寻找非线性方程根的方法对比 Comparison of Methods for Root Finding Problem of Nonlinear Equations

1. 五种方法的对比 Comparison of Five Methods

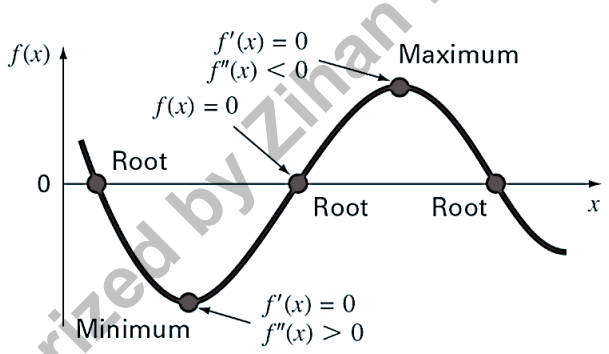
Method	**Derivative**	**Convergence Rate**	**Condition**	**Initial Values Requirement**
Fixed-Point Iteration	No	Linear $p = 1$	Equivalence between $f(x)$ and $g(x)$, convergence $\|g'(x)\| < 1$ and continuity for $g(x)$	Single point, needs to be near the root.
Newton's Iteration	Yes	Quadratic $p = 2$	$f(x) \in C^2[a, b]$ and $f'(x^*) \neq 0$	Single point, needs to be near the root.
Secant Method	No	Superlinear $p = 1.618$	$f(x)$ is continuous and the initial point is near the root.	Two points, need to be near the root.
Bisection Method	No	Linear $p = 1$	$f(x)$ continuous with opposite signs at the ends of the interval.	Two points form an interval. The function in the interval must pass through the zero.
Quadratic Inverse Interpolation	No	Superlinear $p = 1.44$	$f(x)$ is continuous, the three-point function values do not coincide and are close to the root, and the sign changes.	Three points, need to be close to the root. The function must pass through the zero point in the large interval formed by the three points.

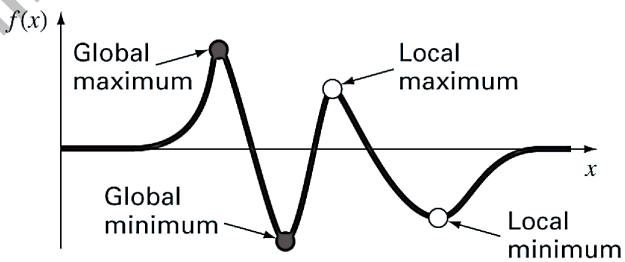
2. 如何选择方法 How to Choose A Methodology
    a. 函数连续但导数不可用:
        i. 如果简单易实现且鲁棒性优先：选择 Bisection Method。
        ii. 如果需要更快的收敛速度：选择 Secant Method。
        iii. 如果函数的反函数易近似：选择 Quadratic Inverse Interpolation。
    b. 函数连续且导数可用: 使用 Newton's Iteration。
    c. 函数行为复杂或初始点远离根: 选择 Bisection Method。
    d. 非线性方程难以重写或需要低成本实现: 选择 Fixed-Point Iteration。

第七单元 单变量优化 Chapter 7 Univariate Minimization

优化 Optimization

1. 通常的优化目标和本单元讨论的问题 Typical Optimization Objectives and Problems Discussed in this Chapter
    a. 优化目标 Typical Optimization Objectives: 优化 Optimization 的通常目标是最大化 Maximize 或者最小化 Minimize 某一数量（比如利润），基于变量（如材料的成本和数量）的选择。但是，优化问题 Optimization Problem 有时也包括约束条件 Constraint（如成本必须为正值）。
    b. 本单元讨论的问题 Problems Discussed in this Chapter
        i. 单变量函数 Single Variable Function 的无约束最小化问题 Minimization Without Constraints
        ii. 如何将这些方法适配于最大化问题 Maximization Problems？
    c. 全局极值与局部极值 Global Extrema vs. Local Extrema





        i. 全局极值 Global Extrema: 包括全局最大值 Global Maximum 和全局最小值 Global Minimum。其是函数在整个定义域上的最大值或最小值。
        ii. 局部极值 Local Extrema: 包括局部最大值 Local Maximum 和局部最小值 Local Minimum。其是函数在某一局部区间上的最大值或最小值。
        iii. 研究重点 Focus: 当前仅关注于通过函数值$f(x)$来寻找一个 Local Minimum。
    d. 假设条件 Assumption

i. 在本章，我们假设函数$f(x)$为 U-Shaped，即其需要满足
   - 在区间$[a, x_m)$上严格递减 Strictly Decreasing，在区间$(x_m, b]$上严格递增 Strictly Increasing。
   - 当函数具有连续的导数 Continuous Derivative 时，如果$f'(x)$在$[a, x_m)$上为负，在$(x_m, b]$上为正。则$f(x)$为 U-Shaped。
   - 当函数具有连续的二阶导数 Continuous Second-Order Derivative 时，如果$f(x)$在区间$[a, b]$内开口向上 Concave Up，并且$f'(x_m) = 0$，则$f(x)$为 U-Shaped。

ii. 但是，在真实情况下，我们无法判断函数是否真正具有 U-Shaped。

2. 优化的四种方法 Four Ways to Find Roots of Nonlinear Equations
   a. 黄金分割搜索法 Golden Section Search
   b. 二次插值搜索法 Quadratic Interpolation Search
   c. 牛顿方法 Newton's Method
   d. 三次插值搜索法 Cubic Interpolation Search

黄金分割搜索法 Golden Section Search

1. 黄金分割搜索法算法 Golden Section Search Algorithm
   a. 初始设定 Initial Setting
      i. 在区间$[a, b]$中选择两个中间点x_2和x_1。
      ii. 评估函数值$f(x_2)$和$f(x_1)$以确定新的搜索区间 Search Interval。
   b. 更新规则 Updated Rules
      i. 如果$f(x_2) \leq f(x_1)$
         - Local Minimum 位于$[a, x_1]$内。
         - 更新 Search Interval 位于$[a, x_1]$内。
      ii. 如果$f(x_2) \geq f(x_1)$
         - Local Minimum 位于$[x_2, b]$内。
         - 更新 Search Interval 位于$[x_2, b]$内。
   c. 如何选择初始设定的x_1和x_2 How do I Select the Initial Settings x_1 and x_2?
      i. 推导过程 Derivation Process: 设整个区间长度为l_0，分成两部分l_1和l_2，满足$l_0 = l_1 + l_2$。黄金比例的比例要求

$$\frac{l_1}{l_2} = \frac{l_0}{l_1}$$

计算比例因子r^*

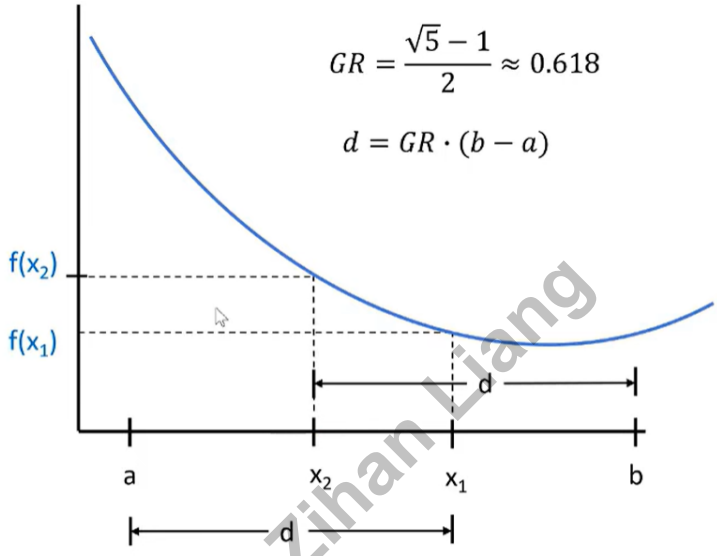
$$r^* \; or \; d = \frac{\sqrt{5} - 1}{2}(x_u - x_l)$$

      ii. x_1和x_2公式 Formula of x_1 and x_2

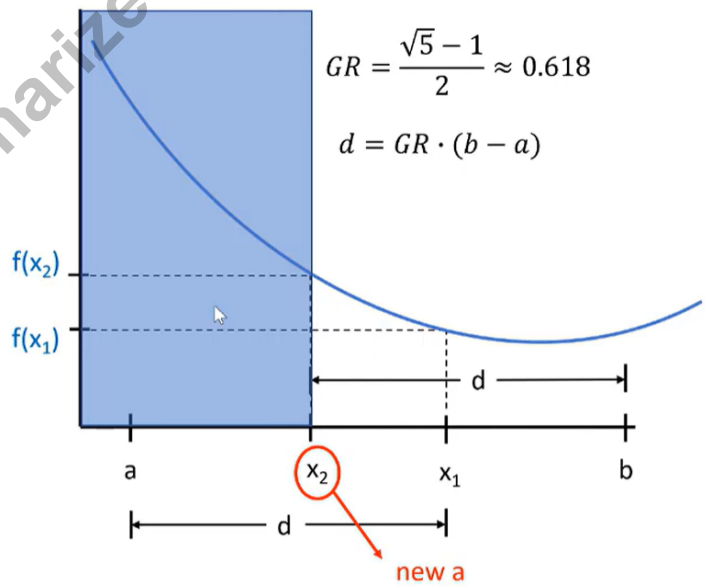
$$\begin{cases} x_1 = x_l + r^* \\ x_2 = x_u - r^* \end{cases}$$

d. 图示解释 Graphical Interpretation

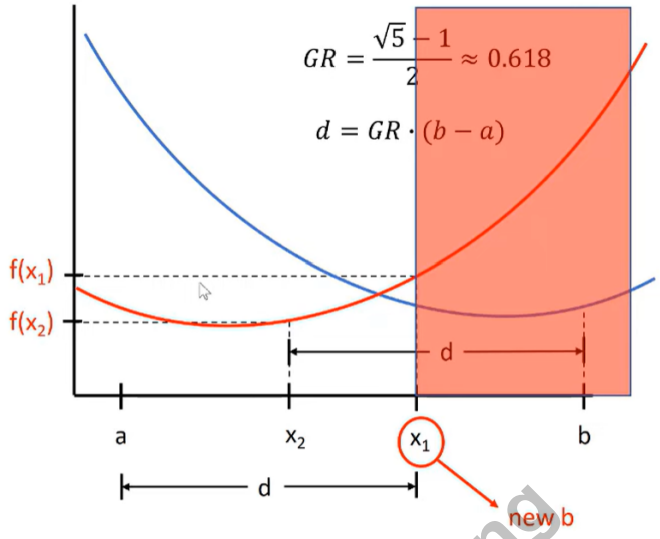
i. Step 1: 在区间$[a, b]$中选择两个中间点x_2和x_1。根据黄金比例公式进行计算。并计算两个中间点对应的函数值$f(x_2), f(x_1)$。



ii. Step 2: 对比$f(x_2), f(x_1)$的大小。如果$f(x_2) \geq f(x_1)$，我们 eliminate all $x < x_2$，这个时候x_2成为新的a，而x_1成为新的x_2，b不变。

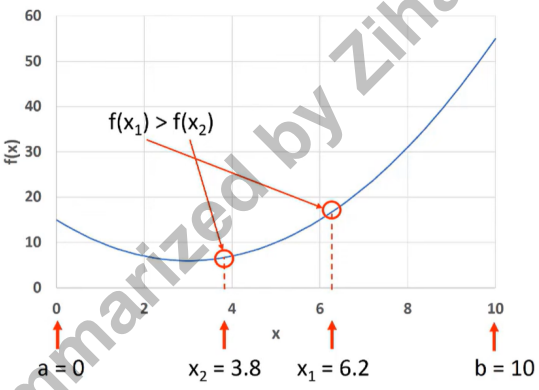


如果$f(x_2) \leq f(x_1)$，我们 eliminate all $x > x_1$，这个时候x_1成为新的b，而x_2成为新的x_1，a不变。



e. 案例 Example: 寻找$f(x) = x^2 - 6x + 15$在$x = 0$和$x = 4$之间的 Minimum。

i. Step 1: 在区间$[a, b]$中选择两个中间点x_2和x_1。根据黄金比例公式进行计算。并计算两个中间点对应的函数值$f(x_2), f(x_1)$。



$GR = \frac{\sqrt{5}-1}{2} \approx 0.618$

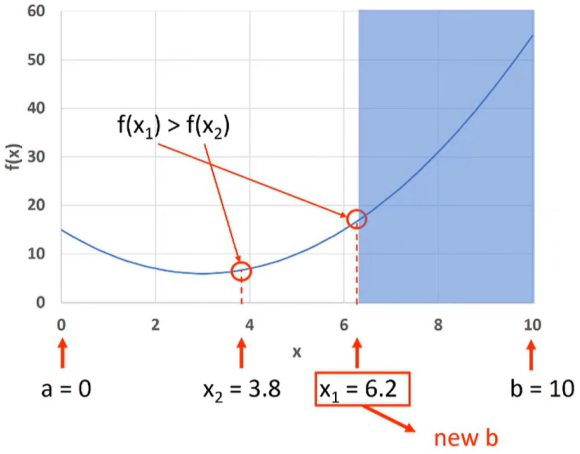
$d = GR \cdot (b-a)$

$d = 0.618 \cdot (10-0) = 6.2$

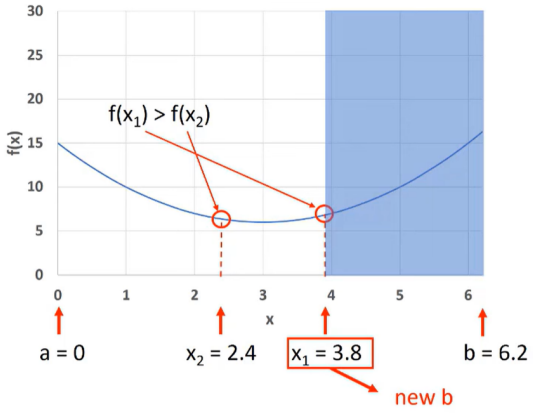
$x_1 = a + d = 0 + 6.2 = 6.2$

$x_2 = b - d = 10 - 6.2 = 3.8$

ii. Step 2: 如果$f(x_2) \leq f(x_1)$，我们 eliminate all $x > x_1$，这个时候x_1成为新的b，而x_2成为新的x_1，a不变。新的 Search Interval 为$[0, 6.2]$。



iii. Step 3: 进入新的 Search Interval $[0, 6.2]$。新的中间点为$x_2 = 2.4, x_1 = 3.8$。此时，$f(x_2) \leq f(x_1)$，我们 eliminate all $x > x_1$，这个时候x_1成为新的b，而x_2成为新的x_1，a不变。新的 Search Interval 为$[0, 3.8]$。



$$GR = \frac{\sqrt{5}-1}{2} \approx 0.618$$

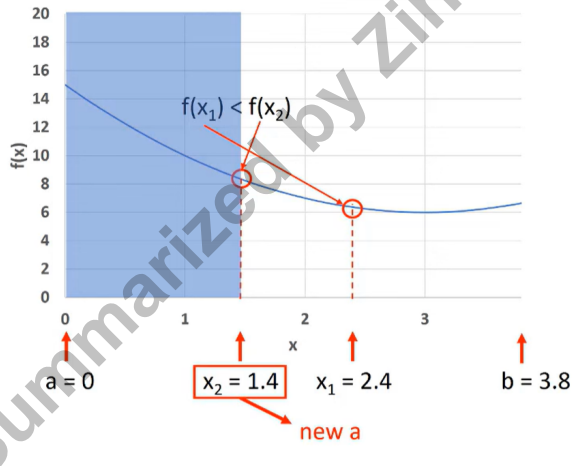
$$d = GR \cdot (b - a)$$

$$d = 0.618 \cdot (6.2 - 0) = 3.8$$

$$x_1 = a + d = 0 + 3.8 = 3.8$$

$$x_2 = b - d = 6.2 - 3.8 = 2.4$$

iv. Step 4: 进入新的 Search Interval $[0, 3.8]$。新的中间点为$x_2 = 1.4, x_1 = 2.4$。此时，$f(x_2) \geq f(x_1)$，我们 eliminate all $x < x_2$，这个时候x_2成为新的a，而x_1成为新的x_2，b不变。新的 Search Interval 为$[1.4, 3.8]$。



$$GR = \frac{\sqrt{5}-1}{2} \approx 0.618$$

$$d = GR \cdot (b - a)$$

$$d = 0.618 \cdot (3.8 - 0) = 2.4$$

$$x_1 = a + d = 0 + 2.4 = 2.4$$

$$x_2 = b - d = 3.8 - 2.4 = 1.4$$

v. Step 5: 以此类推，不断逼近。

2. 黄金分割搜索法的高效性 Efficiency of the Golden Section Search
   a. 在每次迭代中，无需重新评估所有点的函数值，只需利用之前的计算结果。
   b. 每次迭代后，Search Interval 缩小为原区间的61.8%（黄金比例）。

二次插值搜索法 Quadratic Interpolation Search

1. 抛物线模型与近似函数 Parabolic Models and Approximating Functions
   a. 抛物线模型 Parabolic Models: 二次多项式（抛物线）可以很好地近似函数$f(x)$在极值点附近的形状。利用抛物线拟合，可以通过较少的点估计出函数的极值。

b. 抛物线模型方程 Parabolic Model Equation: 使用二次多项式$P_2(x)$来近似函数$f(x)$

$$P_2(x) = A(x-c)^2 + B(x-c) + C$$

这里，A, B, C是拟合的参数，c是抛物线的对称轴附近的一个参考点。

$$A = \frac{(f_a - f_c)\cdot(b-c) - (f_b - f_c)\cdot(a-c)}{(a-c)\cdot(b-c)\cdot(a-b)}$$

$$B = \frac{(f_a - f_c)\cdot(b-c)^2 - (f_b - f_c)\cdot(a-c)^2}{(a-c)\cdot(b-c)\cdot(b-a)}$$

$$C = f(c)$$

c. 极值位置计算 Calculation of Extreme Value

$$d = c - \frac{B}{2A}$$

2. 二次插值搜索法算法 Quadratic Interpolation Search Algorithm
   a. Step 1: 给定搜索区间 Search Interval $[a, b]$，选择中间点$c \in (a, b)$，使得

   $$f(c) < \min\{f(a), f(b)\}$$

   一般情况下

   $$c = \frac{1}{2}(a+b)$$

   b. Step 2: 通过区间左端点$(a, f(a))$、中间点$(c, f(c))$、以及右端点$(b, f(b))$，拟合出抛物线。
   c. Step 3: 计算这条抛物线的极值点（即最小值或最大值）位置d。
   d. Step 4: 更新 Search Interval 并继续迭代。
      i. 如果$f(d) < f(c)$，则更新区间为$[a, c, d]$。
      ii. 如果$f(d) \geq f(c)$，则更新区间为$[c, d, b]$。
   e. Step 5: 停止条件，满足其一。
      i. 当$f(c) \geq f(a)$或$f(c) \geq f(b)$时，算法停止。
      ii. 当两次计算的极小值点足够接近，$|d' - d| < \epsilon$，算法停止。
3. 二次插值搜索法案例 Example of Quadratic Interpolation Search: 使用函数$f(x) = (x-2)^2 + 1$，在$[0,4]$区间内目标是找到该函数的最小值。
   a. Step 1: 确定a, b, c以及其函数值。

   $$a = 0, b = 4, c = 2$$

   $$f(a) = f(0) = (0-2)^2 + 1 = 5$$

   $$f(b) = f(4) = (4-2)^2 + 1 = 5$$

   $$f(c) = f(2) = (2-2)^2 + 1 = 1$$

   b. Step 2: 拟合抛物线。

   $$C = f(c) = 1$$

   $$A = \frac{(5-1)(4-2) - (5-1)(0-2)}{(0-2)(4-2)(0-4)} = 1$$

$$B = \frac{(5-1)(4-2)^2 - (5-1)(0-2)^2}{(0-2)(4-2)(4-0)} = 0$$

因此抛物线最终表达式为

$$P_2(x) = (x-2)^2 + 1$$

c. Step 3: 计算极小值点。

$$d = c - \frac{B}{2A} = 2 - \frac{0}{2(1)} = 2$$

d. Step 4: 由于$f(d) = 1 \geq f(c)$，更新区间为$[a = 2, b = 2, c = 4]$。但是，我们发现$f(c) \geq f(a)$，因此算法停止。

牛顿法 Newton's Method

1. 牛顿法算法 Newton's Method Algorithm
   a. 牛顿法的迭代公式 Iterative Formula for Newton's Method
      i. 牛顿法的优化公式 Optimization Formula

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

      ii. 公式推导 Derivation of Formula: 将目标函数$f(x)$在x_k附近用二次泰勒展开近似

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$

对x求导，得到

$$f'(x) \approx f'(x_k) + f''(x_k)(x - x_k)$$

对上述二次函数求最小值点，即对其导数为零

$$f'(x_k) + f''(x_k)(x - x_k) = 0$$

解得

$$x = x_k - \frac{f'(x_k)}{f''(x_k)}$$

   b. 牛顿法算法 Newton's Method Algorithm
      i. Step 1: 设置 Initial Value x_0。
      ii. Step 2: 计算$f'(x_k), f''(x_k)$。
      iii. Step 3: 使用公式进行计算，得到下一个x值。

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

      iv. Step 4: 检查收敛条件: 如果满足以下条件之一，终止迭代
         - $|f'(x_{k+1})| < \epsilon$ 梯度接近于 0
         - $|x_{k+1} - x_k| < \epsilon$ 更新步长很小
         - 达到最大迭代次数

v. Step 5: 如果未收敛，设置$x_k = x_{k+1}$，重复 Step 2-4。

c. 案例 Example: 寻找函数$f(x) = x^2 - 4x + 5$，寻找其 Local Minimum。

i. Step 1: 设置 Initial Value $x_0 = 0$。

ii. Step 2: 计算$f'(x_k), f''(x_k)$。

$$f'(x) = 2x - 4$$
$$f''(x) = 2$$

iii. Step 3: 牛顿法迭代。

- $k = 0: x_0 = 0, f'(x_0) = 2(0) - 4 = -4, f''(x_0) = 2$

$$x_1 = 0 - \frac{-4}{2} = 2$$

- $k = 1: x_1 = 2, f'(x_1) = 2(2) - 4 = 0$ 梯度为 0，停止迭代。

iv. Result: $x^* = 2, f(x^*) = 2^2 - 4(2) + 5 = 1$

三次插值搜索法 Cubic Interpolation Search

1. 埃尔米特多项式与近似函数 Hermite Polynomials and Approximating Functions

a. 三次插值搜索法的背景 Background of Cubic Interpolation Search

i. 假设函数$f(x)$在区间$[a, b]$内是 U-Shaped 的。

ii. $f'(a)f'(b) < 0$是函数 U-Shaped 的必要条件。

b. 构造三次插值多项式 Constructing Cubic Interpolation Polynomials

i. 为了逼近$f(x)$的 Local Minimum，需要使用三次 Hermite Polynomial。

ii. Hermite Polynomial 通过插值满足以下条件

$$P_3(a) = f(a), P_3(b) = f(b), P_3'(a) = f'(a), P_3'(b) = f'(b)$$

2. 三次插值搜索法算法 Cubic Interpolation Search Algorithm

a. 埃尔米特多项式 Hermite Polynomials

$$P_3(x) = A(x - c)^3 + B(x - c)^2 + C(x - c) + D$$

这是一个标准的三次 Hermite Polynomial，使用$f(a), f(b), f'(a), f'(b)$插值构造。其中c是当前区间$[a, b]$的中点；d是区间长度的一半。

$$c = \frac{a + b}{2}, d = \frac{b - a}{2}$$

三次多项式的系数A, B, C, D使用函数值和导数值计算

$$A = \frac{d(f'(b) + f'(a)) - (f(b) - f(a))}{4d^3}$$
$$B = \frac{f'(b) - f'(a)}{4d}$$
$$C = \frac{3(f(b) - f(a)) - d(f'(b) + f'(a))}{4d}$$
$$D = \frac{f(a) + f(b)}{2} - d\frac{f'(b) - f'(a)}{4}$$

b. 三次插值搜索法算法内容 Algorithm Content

i. Step 1: 根据 Hermite Polynomials 的公式，得到由原始区间$[a, b]$对应的 Hermite Polynomial $P_3(x)$。

ii. Step 2: 对$P_3(x)$求导，并求得$P_3'(x) = 0$的根（最多 2 个根）。

$$x = \frac{a+b}{2} + \frac{-B \pm \sqrt{B^2 - 3AC}}{3A}$$

我们选择较小的根c作为我们的 Local Minimum 的第一次猜测。

iii. Step 3: 停止条件检查。如果$f'(c) < \epsilon$，或$f'(c) \approx 0$时，算法停止，对应c值为 Local Minimum。

iv. Step 3: 如果不满足停止条件，则需要更新 Search Interval，准备第二次迭代。根据第一个猜测c，计算$f'(c)$。

- 如果$f'(c) < 0$，更新 Search Interval 为$[c, b]$。
- 如果$f'(c) > 0$，更新 Search Interval 为$[a, c]$。

v. Step 4: 继续迭代，计算下一个猜测。

c. 案例 Example: 计算$f(x) = (x-2)^2 + 1$在$[1,3]$区间内的 Local Minimum。

i. Step 1: 根据 Hermite Polynomials 的公式，得到三次多项式为

$$p_3(x) = (x-2)^2 + 1$$

ii. Step 2: 对$P_3(x)$求导，并求得$P_3'(x) = 0$的根（最多 2 个根）。

$$p_3'(x) = 2(x-2) = 0 \Longrightarrow x = 2$$

此时$c = 2$是第一次猜测。

iii. Step 3: 更新区间。

$$f(c) = f(2) = 1$$

iv. Step 4: 由于$f'(c) = 0$，迭代停止。Local Minimum 为 2。