

# Numerical Analysis

Iydon *de* S<sub>U</sub>ST<sub>E</sub>X

2018 年 11 月 16 日

# 目录

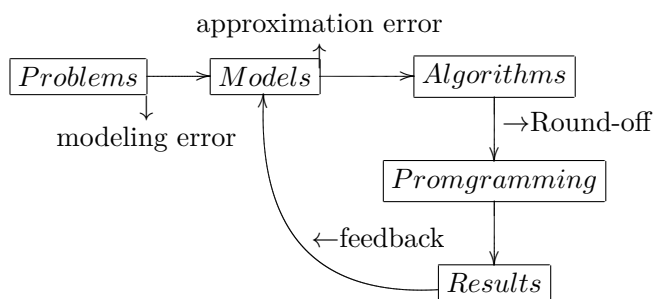
<b>第零章 Preface</b>	<b>5</b>
0.1 Preface . . . . .	5
<b>第一章 Mathematical Preliminaries and Error Analysis</b>	<b>6</b>
1.1 Mathematical Preliminaries and Error Analysis . . . . .	6
1.1.1 Round-off Errors and Computer Arithmetic . . . . .	6
<b>第二章 Solutions of Equations in One Variable</b>	<b>10</b>
2.1 Root-finding problem . . . . .	10
2.1.1 The Bisection Method . . . . .	10
2.1.2 Fixed-Point Iteration . . . . .	11
2.1.3 Newton's Method and Its Extensions . . . . .	13
2.1.4 The Secant Method . . . . .	14
2.1.5 The Method of False Position . . . . .	14
2.2 Error Analysis for Iterative Methods . . . . .	16
<b>第三章 Interpolation and Polynomial Approximation</b>	<b>18</b>
3.1 Interpolation and the Lagrange Polynomial . . . . .	18
3.2 Data Approximation and Neville's Method . . . . .	19
3.2.1 Neville's Method . . . . .	20
3.3 Divided Differences . . . . .	20
3.3.1 Divided Differences Notation . . . . .	20
3.3.2 Forward Differences . . . . .	21
3.3.3 Backward Differences . . . . .	21
3.3.4 Centered Differences . . . . .	22

3.4	Hermite Interpolation . . . . .	22
3.4.1	Hermite Polynomials . . . . .	23
3.4.2	Hermite Polynomials Using Divided Differences . . . .	23
3.5	Cubic Spline Interpolation . . . . .	24
3.5.1	Construction of a Cubic Spline . . . . .	24
3.5.2	Clamped Splines . . . . .	26
<b>第四章</b>	<b>Numerical Differentiation and Integration</b>	<b>27</b>
4.1	Numerical Differentiation . . . . .	27
4.1.1	Three-Point Formulas . . . . .	28
4.1.2	Five-Point Formulas . . . . .	28
4.1.3	Second Derivative Midpoint Formula . . . . .	29
4.1.4	Round-Off Error Instability . . . . .	29
4.2	Richardson's Extrapolation . . . . .	29
4.3	Elements of Numerical Integration . . . . .	30
4.3.1	The Trapezoidal Rule . . . . .	30
4.3.2	Simpson's Rule . . . . .	31
4.3.3	Measuring Precision . . . . .	31
4.3.4	New-ton-Cotes Formulas . . . . .	31
4.4	Composite Numerical Integration . . . . .	33
4.4.1	Round-off Error Stability . . . . .	35
4.5	Romberg Integration . . . . .	35
4.6	Gaussian Quadrature . . . . .	36
4.6.1	Legendre Polynomial . . . . .	36
4.6.2	Gaussian Quadrature on Arbitrary Intervals . . . . .	37
<b>第五章</b>	<b>Initial-Value Problems for Ordinary Differential Equations</b>	<b>38</b>
5.1	The Elementary Theory of Initial-Value Problems . . . . .	38
5.1.1	Well-Posed Problems . . . . .	39
5.2	Euler's Method . . . . .	39
5.2.1	Errors Bounds for Euler's Method . . . . .	40
5.3	Higher-Order Taylor Method . . . . .	41
5.3.1	Taylor Method of Order $n$ . . . . .	42

5.4	Runge-Kutta Methods . . . . .	42
5.4.1	Runge-Kutta Methods of Order Two . . . . .	43
5.4.2	Midpoint Method . . . . .	44
5.4.3	Modified Euler Method . . . . .	44
5.4.4	Higher-Order Runge-Kutta Methods . . . . .	44
5.4.5	Computational Comparisons . . . . .	45
5.5	Error Control and the Runge-Kutta-Fehlberg Method . . . . .	45
5.5.1	收敛性与相容性 . . . . .	45
5.6	Multistep Method . . . . .	47
<b>第六章</b>	<b>Direct Methods for Solving Linear Systems</b>	<b>49</b>
6.1	Linear Systems of Equations and Pivoting Strategies . . . . .	49
6.1.1	Gaussian Elimination with Backward Substitution . . . . .	49
6.1.2	Operation Counts . . . . .	49
6.1.3	Backward substitution *&/ . . . . .	50
6.1.4	Backward substitution +&- . . . . .	50
6.1.5	Gaussian Elimination with Partial Pivoting . . . . .	50
6.1.6	Gaussian Elimination with Scaled Partial Pivoting . . . . .	51
6.2	Matrix Factorization . . . . .	52
<b>第七章</b>	<b>Iterative Techniques in Matrix Algebra</b>	<b>53</b>
7.1	Norms of Vectors and Matrices . . . . .	53
7.1.1	Matrix Norms and Distances . . . . .	54
7.2	Eigenvalues and Eigenvectors . . . . .	54
7.2.1	Spectral Radius . . . . .	55
7.2.2	Convergent Matrices . . . . .	55
7.3	The Jacobi and Gauss-Siedel Iterative Techniques . . . . .	55

# 第零章 Preface

## 0.1 Preface



# 第一章 Mathematical Preliminaries and Error Analysis

## 1.1 Mathematical Preliminaries and Error Analysis

### 1.1.1 Round-off Errors and Computer Arithmetic

#### Binary machine numbers

定义 1.1.1 舍入误差 舍入误差形成原因：进行有限位的运算 (*finite digits arithmetic*)

其中，*IEEE:754-2008* 规定二进制机器数 (*Binary machine numbers*) 中浮点数 (*floating-point*) 存储规范如下：

$$(-1)^S 2^{c-1023} (1 + f)$$
$$\begin{cases} S : 0/1 & \text{signpart} \\ c : 11 \text{ digits} & \text{exponential part.} \\ f : 52 \text{ digits} & \text{mantissa part} \end{cases}$$

由于实数的稠密性，可知找不到比某一个数大的最小的数或小的最大的数，但是在计算机中可以找到，所以计算机不能表示所有的数。

**Decimal machine numbers**

$\pm 0.d_1d_2 \cdots d_n \times 10^n$  其中  $1 \leq d_1 \leq 9$ ,  $0 \leq d_i \leq 9$ ,  $\forall i \geq 2$ 。如果记真实的数为  $y$ , 其浮点数表示为  $fl(y)$ 。

当存在  $y = 0.d_1d_2 \cdots d_kd_{k+1} \cdots \times 10^n$ , 其浮点数表示有如下两种方式:

1. Chopping: chop off digits, say  $d_{k+1}d_{k+2} \cdots$ .
2. Rounding:  $y + 5 \times 10^{n-(k+1)}$ , then chopping.

**例 1.1.1**

$\pi = 3.14159265 \cdots$ , 取 5 位。

- Chopping:  $fl(y) = 0.31415 \times 10^1$ .
- Rounding:  $fl(y) = 0.31416 \times 10^1$ .

 $\pi$ 

**定义 1.1.2** Suppose  $p^*$  is an approximation of  $p$ .

$$\begin{cases} \text{absolute error} &= |p^* - p| \\ \text{relative error} &= \frac{|p^* - p|}{p} \end{cases}$$

**定义 1.1.3** 有效数字 (Significant digits)  $p^*$  is said to approximate  $p$  with  $t$  significant digits. If  $t$  is the largest nonnegative integer, s.t.

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

Chopping floating:

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

$$fl(y) = 0.d_1 \cdots d_k \times 10^n$$

Chopping: (其有效位数至少为  $k-1$ )

$$\frac{|fl(y) - y|}{|y|} = \frac{0.0 \cdots 0 d_{k+1} \cdots \times 10^n}{0.d_1 \cdots d_k d_{k+1} \cdots \times 10^n} \leq 10^{1-k}$$

Rounding: (其有效位数至少为  $k$ )

$$\frac{|fl(y) - y|}{|y|} \leq \frac{0.0 \cdots 1 d_{k+1} \cdots \times 10^n}{0.d_1 \cdots d_k d_{k+1} \cdots \times 10^n} \leq 10^{-k}$$

### Machine Operators

记计算机的加减乘除为  $\oplus \ominus \otimes \oslash$ , 于是有

$$x \oslash y = fl(fl(x) \oslash fl(y))$$

Four cases to avoid:

1. 两个十分接近的数 (two nearly equal)。
2. 分子远大于分母 (numerator » denominator)。
3. 避免大数吃掉小数。

### Nested method (秦九韶算法)

```

input :  $a_0, a_1, \dots, a_n(\text{given})$ ;  $x$ 
output:  $P_n(x)$ 

1  $S_n \leftarrow a_n$ ;
2 for  $k \leftarrow n - 2$  to 0 do
3    $S_k \leftarrow x S_{k+1} + a_k$ ;
4 end
5  $P_n(x) \leftarrow S_0$ ;

```

```

1 def nested(poly:list=[1], x:float=0.0)->float:
2     """
3     Horner nested polynomial calculation.
4

```



```

5      Args:
6          poly: List, store the coefficient of the polynomial.
7          x: Float, specify the variable in the polynomial.
8
9      Returns:
10         Float, result.
11
12     Raises:
13         If 'poly' is empty, raise IndexError.
14         If type(args) does not correspond, raise TypeError.
15     """
16     result = poly[0]
17     for i in range(1, poly.__len__()):
18         result = x*result + poly[i]
19     return result

```

### Convergence (收敛性)

Stable: small change in initial data and the error is small.

若  $E_0$  为初始值误差,  $E_n$  为  $n$  步的误差,

- $E_n \approx C$  (不依赖  $n$ ), 称之为线性。
- $E_n \approx C^n E_0$  则可由  $C$  的取值判断是否稳定。

**定义 1.1.4 Rates of Convergence** 当  $n \rightarrow \infty$ ,  $\alpha_n \rightarrow \alpha$ ,  $\beta_n \rightarrow 0$ , 其中  $|\alpha_n - \alpha| \leq k |\beta_n|$  (与  $n$  的取值无关), 则称  $\alpha_n$  是以  $\beta_n$  的速度收敛到  $\alpha$  的。

$$\alpha_n = \alpha + o(\beta_n).$$

## 第二章 Solutions of Equations in One Variable

### 2.1 Root-finding problem

#### 2.1.1 The Bisection Method

定理 2.1.1 *Intermediate Value Theorem*  $f \in [a, b], \forall k \in f([a, b]), \exists c \in [a, b], s.t. f(c) = k$ 。

```
1 def Bisection(fun, a:float, b:float, max_step:int=128, ...
2     eps:float=1e-6)->float:
3     mid_last = a
4     if fun(a)*fun(b) < 0:
5         for i in range(0, max_step):
6             mid = (a+b) / 2
7             if abs(mid-mid_last)<eps or abs(fun(mid))<eps:
8                 print("Step: %d\nZero: %fc"%(i, mid))
9                 return mid
10            else:
11                if fun(mid)*fun(a)<0:
12                    b = mid
13                else:
14                    a = mid
15            mid_last = mid
16    print('Bisection cannot be convergent within..')
```

the pre-set steps.')

**定理 2.1.2**  $f \in C[a, b]$ (continuous), 根据如上算法,  $P_i$  为 mid 的序列。如果  $\exists \text{ root } P \in [a, b]$ , 则有  $|P_n - P| \leq \frac{b-a}{2^n}$ 。

[Proof]  $|b_n - a_n| = \frac{b-a}{2^{n-1}}$ ,

$$|P_n - P| \leq \frac{1}{2}(b_n - a_n) = \frac{b-a}{2^n}$$

于是  $P_n = P + o(2^{-n})$ 。

□

### 2.1.2 Fixed-Point Iteration

**定义 2.1.1** Fixed-point Iteration 对  $g(P)$ , 如果  $\forall x \in [a, b]$ , 如果  $\exists P$  s.t.  $g(P)=P$ , 则称  $P$  为不动点 (fixed point)。

如果  $g(x) \in C[a, b]$  并且  $g([a, b]) \subset [a, b]$ , there exists at least one  $p \in [a, b]$ , s.t.  $g(p)=p$ 。

**定理 2.1.3** 不动点迭代根的存在唯一性定理  $g(x) \in C[a, b]$ ,  $g([a, b]) \subset [a, b]$ 。  $\forall x \in [a, b]$ , 都有  $g'(x) \leq \kappa < 1$ 。

[Proof]

存在性:

$$\begin{cases} h(a) = g(a) - a \geq 0 \\ h(b) = g(b) - b \leq 0 \end{cases}$$

于是有  $h(a)h(b) \leq 0$ , 则  $\exists p$ , s.t.  $h(p)=0$ 。

唯一性:

假设存在两个根  $P_1, P_2$ , 使得  $P_1 = g(P_1)$ ,  $P_2 = g(P_2)$ , 但是  $P_1 \neq P_2$ 。

$$\begin{aligned} |g(P_1) - g(P_2)| &= |g'(\xi)| |P_1 - P_2|, \quad \xi \in [P_1, P_2]. \\ &\leq \kappa |P_1 - P_2|, \text{contradiction.} \end{aligned}$$

□

**定理 2.1.4** 不动点收敛的充分条件  $g \in C[a, b]$ ,  $g([a, b]) \subset [a, b]$ ,  $g'(x)$  存在, 并且  $|g'(x)| \leq \kappa < 1$ .  $\forall P_0 \in [a, b]$ , 定义序列  $P_i = g(P_{i-1})$ ,  $i = 1, 2, \dots$ , 则  $\lim_{n \rightarrow \infty} P_n = P$  ( $P$  为不动点)。

[Proof]

$$\begin{aligned}
 |P_n - P| &= |g(P_{n-1}) - g(P)| \\
 &= |g'(\xi_{n-1})| |P_{n-1} - P| \\
 &\leq \kappa |P_{n-1} - P| \\
 &\leq \dots \leq \dots \\
 &\leq \kappa^n |P_0 - P| \rightarrow 0.
 \end{aligned}$$

□

其中, 寻找不动点的代码如下:

```

1 def fixed_point(fun, start:float=0, max_step:int=128, ...
2     eps:float=1e-6)->float:
3     new_val = fun(start)
4     for i in range(0, max_step):
5         old_val = new_val
6         new_val = fun(old_val)
7         if -eps < old_val - new_val < eps:
8             print(i)
9             return new_val

```

**定理 2.1.5** 如果满足定理 2.1.4, 则  $p_n$  接近  $p$  的误差可以表示为

$$|P_n - P| \leq \kappa^n \max\{P_0 - a, b - P_0\}$$

并且有

$$\begin{aligned}
 |P_n - P| &= |P_n - P_{n+1} + P_{n+2} - P_{n+3} + \dots| \\
 &\leq |P_n - P_{n+1}| + |P_{n+2} - P_{n+3}| + \dots \\
 &= \kappa^n (1 + \kappa + \kappa^2 + \dots) |P_0 - P_1| \\
 &= \frac{\kappa^n}{1 - \kappa} |P_0 - P_q|
 \end{aligned}$$

### 2.1.3 Newton's Method and Its Extensions

Suppose that  $f \in C^2[a, b]$ .  $p_0 \in [a, b]$  and  $f'(p_0) \neq 0$  and  $|p - p_0|$  is small.

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

$|p - p_0|$  is small, the term involving  $(p - p_0)^2$  is much smaller, then we will get

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

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

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \text{for } n \geq 1.$$

```

1 def Newton_method(f, df, start:float=0.0, max_step:int=32,...
2     sign_dig:int=6)->float:
3     fun = lambda x: x - f(x)/df(x)
4     return fixed_point(fun, start, max_step, sign_dig)
5
6 def fixed_point(fun, start:float, max_step:int,...
7     sign_dig:int)->float:
8     fl = lambda x: round(x, 100)
9     eps = 10**(-sign_dig)
10    new_val = fun(start)
11    for i in range(0, max_step):
12        old_val = fl(new_val)

```

```

13     new_val = fl(fun(old_val))
14     if abs(old_val-new_val)<=2*eps:
15         return (i, new_val)
16     return "Max_step..."

```

**定理 2.1.6** 牛顿法的收敛性 *Let  $f \in C^2[a, b]$ . If  $p \in (a, b)$  is such that  $f(p) = 0$  and  $f'(p) \neq 0$ , then there exists a  $\delta > 0$  such that Newton's method generates a sequence  $\{p_n\}_{n=0}^{\infty}$  converging to  $p$  for any initial approximation  $p_0 \in [p - \delta, p + \delta]$ .*

**[Proof]** 证明基于将牛顿迭代法看作 functional iteration scheme  $p_n = g(p_{n-1})$ , 既然  $f'(p) \neq 0$ , 则  $\exists \delta_1 > 0$  使得  $f'(x) \neq 0$  对于所有的  $x \in [p - \delta_1, p + \delta_1]$ , 于是有  $g(x) = x - \frac{f(x)}{f'(x)}$ . 求导后有

$$g'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2}.$$

于是有  $g'(p) = \frac{f(p)f''(p)}{[f'(p)]^2} = 0$ , 又因为  $g'$  是连续的,  $0 < k < 1$ , 所以  $\exists \delta$ , 有  $0 < \delta < \delta_1$ , 并且  $|g'(x)| \leq k$ , 对于所有的  $x \in [p - \delta, p + \delta]$  都成立。

接下来需要证明  $g$  映射  $[p - \delta, p + \delta]$  到  $[p - \delta, p + \delta]$ 。

$$|g(x) - p| = |g(x) - g(p)| = |g'(\xi)| |x - p| \neq k |x - p| < |x - p| < \delta.$$

综上所述, 存在  $\delta > 0$ , 当  $p_0 \in [p - \delta, p + \delta]$  都有牛顿迭代法收敛到  $p$ .  $\square$

### 2.1.4 The Secant Method

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

### 2.1.5 The Method of False Position

The method generates approximations in the same manner as the Secant method, but it includes a test to ensure that the root is always bracketed between successive iterations.

```

1 def false_position(f, start:list, max_step:int=32, ...
2     eps:float=1e-6) -> float:
3     """
4     False position.
5     -----
6     Args:
7         f: Function.
8         start: List of float, the first iteration point.
9         max_step: Integer, max number of iteration.
10
11     Returns:
12         Float zero.
13         zero / fun(zero)\\sim 0.
14
15     Raises:
16         None.
17     """
18     p = [i for i in start]
19     q = [f(i) for i in start]
20     for i in range(max_step):
21         _p = p[-1] - q[-1]*(p[-1]-p[0])/(q[-1]-q[0])
22         if abs(_p-p[-1]) < eps:
23             return (i, _p)
24         _q = f(_p)
25         if _q*q[-1] < 0:
26             p[0] = p[-1]
27             q[0] = q[-1]
28         p[-1] = _p
29         q[-1] = _q
30     return False

```

## 2.2 Error Analysis for Iterative Methods

**定义 2.2.1** *Order of Convergence*  $\{p_n\}_{n=0}^{\infty}$  是一个收敛到  $p$  的序列, 但是对于所有的  $n$  都有  $p_n \neq p$ , 如果存在正数  $\lambda, \alpha$  满足以下条件:

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

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

如果  $\alpha = 1$  (and  $\lambda < 1$ ), 则称序列为线性收敛 (*linearly convergent*).

如果  $\alpha = 2$ , 则序列为二次收敛 (*quadratically convergent*).

线性收敛:  $|p_n - 0| \approx (0.5)^n |p_0|$ .

二次收敛:  $|\tilde{p}_n - 0| \approx (0.5)^{2^n - 1} |\tilde{p}_0|$ .

**定理 2.2.1** Let  $g \in C[a, b]$  be such that  $g(x) \in [a, b]$ , for all  $x \in [a, b]$ . Suppose that  $g'$  is continuous on  $(a, b)$  and a positive constant  $k < 1$  exists with

$$|g'(x)| \leq k, \quad \text{for all } x \in (a, b).$$

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

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

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

[Proof]

$$\begin{aligned} p_{n+1} - p &= g(p_n) - g(p) = g'(\xi_n)(p_n - p), \\ \Rightarrow \lim_{n \rightarrow \infty} \frac{p_{n+1} - p}{p_n - p} &= \lim_{n \rightarrow \infty} g'(\xi_n) = g'(p) \end{aligned}$$

$\Rightarrow$  If  $g'(p) \neq 0$ , fixed-point iteration exhibits linear convergence with asymptotic error constant  $|g'(p)|$ . □



**定理 2.2.2** *Let  $p$  be a solution of the equation  $x = g(x)$ . Suppose that  $g'(p) = 0$  and  $g''$  is continuous with  $|g''(x)| < M$  on an open interval  $I$  containing  $p$ . Then there exists a  $\delta > 0$  such that, for  $p_0 \in [p - \delta, p + \delta]$ , the sequence defined by  $p_n = g(p_{n-1})$ , when  $n \geq 1$ , converges at least quadratically to  $p$ . Moreover, for sufficiently large values of  $n$ ,*

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

**[Proof]**

$$g(x) = g(p) + g'(p)(x - p) + \frac{g''(\xi)}{2}(x - p)^2,$$

□

## 第三章 Interpolation and Polynomial Approximation

### 3.1 Interpolation and the Lagrange Polynomial

**定理 3.1.1 (Weierstrass Approximation Theorem)**  $f \in C[a, b]$ ,  
 $\forall$  polynomial  $P(x)$ , s.t.  $|f(x) - P(x)| < \epsilon$  for all  $x \in [a, b]$ .

**定理 3.1.2 (nth Lagrange Interpolating Polynomial)**  $x_0, \dots, x_n$  are  $n+1$  distinct numbers, then a unique polynomial  $P(x)$  of degree at most  $n$  exists with

$$\begin{cases} f(x_k) = P(x_k) \quad \text{for } k = 0 : n \\ P(x) = \sum_{k=0}^n f(x_k) L_{n,k}(x) \\ L_{n,k} = \prod_{\substack{i=0 \\ i \neq k}}^n \frac{(x-x_i)}{x_k-x_i} \\ L_{n,k}(x_j) = \delta_{k,j} \end{cases}$$

If  $f \in C^{n+1}[a, b]$ ,  $\exists \xi(x) \in (a, b)$ , s.t.

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

[Proof] let

$$g(t) = f(t) - P(t) - [f(x) - g(x)] \prod_{i=0}^n \frac{(t - x_i)}{x - x_i},$$

$g(t)$  satisfies

$$\begin{cases} g(x_k) = 0 & k = 0 : n \\ g(x) = 0 \\ g \in C^{n+1}[a, b] \end{cases}$$

By *Generalized Rolle's Theorem*,  $\exists \xi \in (a, b)$ , s.t.  $g^{(n+1)}(\xi) = 0$ , then we have

$$\begin{aligned} 0 &= g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - P^{(n+1)}(\xi) - [f(x) - P(x)] \frac{d^{n+1}}{dt^{n+1}} \left[ \prod_{i=0}^n \frac{(t - x_i)}{x - x_i} \right]_{t=\xi} \\ \Rightarrow 0 &= f^{(n+1)}(\xi) - [f(x) - P(x)] \frac{(n+1)!}{\prod_{i=0}^n (x - x_i)} \\ \Rightarrow f(x) &= P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^n (x - x_i). \end{aligned}$$

□

## 3.2 Data Approximation and Neville's Method

**定义 3.2.1** The Lagrange Polynomial that agrees with  $f(x)$  at the  $k$  distinct points  $x_{m_1}, x_{m_2}, \dots, x_{m_k}$  is denoted  $P_{m_1, m_2, \dots, m_k}(x)$ .

**定理 3.2.1** Let  $f$  be defined at  $x_0, x_1, \dots, x_k$ , then

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

is the  $k$ th Lagrange polynomial that interpolates  $f$  at the  $k+1$  points.

### 3.2.1 Neville's Method

To avoid the multiple subscripts, we let  $Q_{i,j}$  ( $0 \leq j \leq i$ ) denote the interpolating polynomial of degree  $j$  on the  $(j+1)$  numbers  $x_{i-j}, \dots, x_i$ .

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

then for  $i = 1 : n$ ,  $j = 1 : i$ ,

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

## 3.3 Divided Differences

### 3.3.1 Divided Differences Notation

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

$$\begin{aligned} f[x_i] &= f(x_i) \\ f[x_i, \dots, x_{i+k}] &= \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i} \end{aligned}$$

for each  $k = 0, 1, \dots, n$ ,  $P_n(x)$  can be rewritten in a form called *Newton's Divided-Difference*:

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

**定理 3.3.1**  $f \in C^n[a, b]$ ,  $x_i \in [a, b]$  for  $i = 0 : n$ ,  $\exists \xi \in (a, b)$ , s.t.

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

**[Proof]** Let  $g(x) = f(x) - P_n(x)$ , which has  $n+1$  distinct zeros in  $[a, b]$ . According to *Generalized Rolle's Theorem*,  $\exists \xi \in (a, b)$ , s.t.  $g^{(n)}(\xi) = 0$ .

$$\begin{aligned} 0 &= g^{(n)}(\xi) = f^{(n)}(\xi) - P_n^{(n)}(\xi) = f^{(n)}(\xi) - n!f[x_0, \dots, x_n] \\ \Rightarrow f[x_0, x_1, \dots, x_n] &= \frac{f^{(n)}(\xi)}{n!} \end{aligned}$$

□

When the nodes are arranged consecutively with equal spacing, then we use  $h = x_{i+1} - x_i$  and  $x = s \cdot h + x_0$ , the equation will become

$$\begin{aligned} P_n(x) &= P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^n s(s-1)\cdots(s-k+1)h^k f[x_0, \dots, x_k] \\ &= f[x_0] + \sum_{k=1}^n \binom{s}{k} k! h^k f[x_0, \dots, x_k]. \end{aligned}$$

### 3.3.2 Forward Differences

$$\begin{aligned} f[x_0, x_1] &= \frac{1}{h} (f(x_1) - f(x_0)) = \frac{1}{h} \Delta f(x_0) \\ f[x_0, x_1, x_2] &= \frac{1}{2h} \left( \frac{\Delta f(x_1) - \Delta f(x_0)}{h} \right) = \frac{1}{2h^2} \Delta^2 f(x_0) \end{aligned}$$

In general,

$$\begin{aligned} f[x_0, x_1, \dots, x_k] &= \frac{1}{k! h^k} \Delta^k f(x_0) \\ \Rightarrow P_n(x) &= f[x_0] + \sum_{k=1}^n \binom{s}{k} \Delta^k f(x_0) \end{aligned}$$

### 3.3.3 Backward Differences

$$\begin{aligned} f[x_n, x_{n-1}] &= \frac{1}{h} \nabla f(x_n) \\ f[x_n, x_{n-1}, x_{n-2}] &= \frac{1}{2h^2} \nabla^2 f(x_n) \end{aligned}$$

In general,

$$\begin{aligned} f[x_n, x_{n-1}, \dots, x_{n-k}] &= \frac{1}{k! h^k} \nabla^k f(x_n) \\ \Rightarrow P_n(x) &= f[x_n] + \sum_{k=1}^n \frac{s(s+1)\cdots(s+k-1)}{k!} \nabla^k f(x_n) \end{aligned}$$

Also, we have

$$\binom{-s}{k} = \frac{-s(-s-1)\cdots(-s-k+1)}{k!} = (-1)^k \frac{s(s+1)\cdots(s+k-1)}{k!}$$

$$\Rightarrow P_n(x) = f[x_n] + \sum_{k=1}^n (-1)^k \binom{-s}{k} \nabla^k f(x_n)$$

### 3.3.4 Centered Differences

$$\begin{aligned} P_n(x) = P_{2m+1}(x) &= f[x_0] + \frac{sh}{2} (f[x_{-1}, x_0] + f[x_0, x_1]) + (sh)^2 f[x_{-1}, x_0, x_1] \\ &+ \cdots \\ &+ s^2(s^2-1)\cdots(s^2-(m-1)^2)h^{2m}f[x_{-m}, \cdots, x_{m+1}] \\ &+ \frac{s^2(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2} (f[x_{-m-1}, \cdots, x_m] + f[x_{-m}, \cdots, x_{m+1}]) \end{aligned}$$

If  $n = 2m + 1$  is odd, we use the above formula, if  $n = 2m$  is even, we delete the last line and then use the above formula.

x	f(x)	1st	2nd	3rd	4th divided differences
$x_{-2}$	$f[x_{-2}]$	$f[x_{-2}, x_{-1}]$	$f[x_{-2}, x_{-1}, x_0]$	$f[x_{-2}, x_{-1}, x_0, x_1]$	$f[x_{-2}, x_{-1}, x_0, x_1, x_2]$
$x_{-1}$	$f[x_{-1}]$	$f[x_{-1}, x_0]$	$f[x_{-1}, x_0, x_1]$	$f[x_{-1}, x_0, x_1, x_2]$	
$x_0$	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$		
$x_1$	$f[x_1]$	$f[x_1, x_2]$			
$x_2$	$f[x_2]$				

## 3.4 Hermite Interpolation

The osculating polynomial approximating a function  $f \in C^m[a, b]$  at  $x_i$  for each  $i = 0 : n$ , of which the derivatives of order less than or equal to  $m_i$ , then the degree of this osculating polynomial is at most  $M = \sum_{i=0}^n m_i + n$ .

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}, \quad \text{for each } i = 0 : n, k = 0 : m_i.$$

$$\begin{cases} n = 0 & m_0 \text{ Taylor polynomial for } f \text{ at } x_0 \\ m_i = 0 (\text{each } i) & \text{nth Lagrange polynomial} \end{cases}$$

### 3.4.1 Hermite Polynomials

**定理 3.4.1**  $f \in C'[a, b]$  and  $x_0, \dots, x_n \in [a, b]$ , the unique polynomial of least degree agreeing with  $f$  and  $f'$  at  $x_0, \dots, x_n$  is the Hermite polynomial of degree at most  $2n + 1$ .

$$H_{2n+1}(x) = \sum_{j=0}^n f(x_j) H_{n,j}(x) + \sum_{j=0}^n f'(x_j) \hat{H}_{n,j}(x)$$

$$\begin{cases} H_{n,j}(x) &= [1 - 2(x - x_j)L'_{n,j}(x_j)] L_{n,j}^2(x) \\ \hat{H}_{n,j}(x) &= (x - x_j)L_{n,j}^2(x) \end{cases}$$

Moreover, if  $f \in C^{2n+2}[a, b]$ , then

$$f(x) = H_{2n+1}(x) + \frac{(x - x_0)^2 \cdots (x - x_n)^2}{(2n + 2)!} f^{(2n+2)}(\xi(x)).$$

[Proof]

$$\begin{aligned} H_{n,j}(x_i) &= \delta_{i,j} & \hat{H}_{n,j}(x_i) &= 0 \\ H'_{n,j}(x_i) &= 0 & \hat{H}'_{n,j}(x_i) &= \delta_{i,j} \end{aligned}$$

□

### 3.4.2 Hermite Polynomials Using Divided Differences

Suppose that the distinct numbers  $x_0, \dots, x_n$  are given together with values of  $f$  and  $f'$ . Define a new sequence  $z_0, \dots, z_{2n+1}$  by

$$z_{2i} = z_{2i+1} = x_i \quad \text{for } i = 0 : n.$$

We have  $H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \dots, z_k](x - z_0) \cdots (x - z_{k-1})$ .

$z$	$f(z)$	First divided differences	$\cdots$
$z_0 = x_0$	$f[z_0] = f(x_0)$	$f[z_0, z_1] = f'(x_0)$	$\vdots$
$z_1 = x_0$	$f[z_1] = f(x_0)$	$f[z_1, z_2] = \frac{f[z_2] - f[z_1]}{z_2 - z_1}$	$\vdots$
$z_2 = x_1$	$f[z_2] = f(x_1)$	$f[z_2, z_3] = f'(x_1)$	$\vdots$
$z_3 = x_1$	$f[z_3] = f(x_1)$	$f[z_3, z_4] = \frac{f[z_4] - f[z_3]}{z_4 - z_3}$	$\vdots$
$z_4 = x_2$	$f[z_4] = f(x_2)$	$f[z_4, z_5] = f'(x_2)$	
$z_5 = x_2$	$f[z_5] = f(x_2)$		

### 3.5 Cubic Spline Interpolation

**定义 3.5.1** Given a function  $f$  defined on  $[a, b]$ ,  $a = x_0 < x_1 < \cdots < x_n = b$ , a cubic spline interpolation  $S$  for  $f$  is a function that satisfies the following conditions.

1.  $S_j(x)$  is a cubic polynomial, on the subinterval  $[x_j, x_{j+1}]$  for  $j = 0 : n - 1$ .
2.  $S_j(x_j) = f(x_j)$ ,  $S_j(x_{j+1}) = f(x_{j+1})$  for  $j = 0 : n - 2$ .
3.  $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$  for  $j = 0 : n - 2$ .
4.  $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$  for  $j = 0 : n - 2$ .
5.  $\begin{cases} \text{natural boundary:} & S''(x_0) = S''(x_n) = 0. \\ \text{clamped boundary:} & S'(x_0) = f'(x_0), S'(x_n) = f'(x_n). \end{cases}$

#### 3.5.1 Construction of a Cubic Spline

Let  $h_j = x_{j+1} - x_j$  (forward):



(1)

$$\begin{aligned}
S_j(x) &= a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3 \quad \text{for } j = 0 : n - 1 \\
\Rightarrow a_{j+1} &= S_{j+1}(x_{j+1}) = S_j(x_{j+1}) \\
&= a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = f(x_{j+1}) \quad \text{for } j = 0 : n - 1
\end{aligned}$$

(2)

$$\begin{aligned}
S'_j(x) &= b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2 \\
\Rightarrow b_{j+1} &= S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}) \\
&= b_j + 2c_j h_j + 3d_j h_j^2 \quad \text{for } j = 0 : n - 1
\end{aligned}$$

(3)

$$\begin{aligned}
S''_j(x) &= 2c_j + 6d_j(x - x_j) \\
\Rightarrow 2c_{j+1} &= S''_{j+1}(x_{j+1}) = S''_j(x_{j+1}) \\
&= 2c_j + 6d_j h_j \quad \text{for } j = 0 : n - 1
\end{aligned}$$

Above all, the linear system to be solved is:

$$Ax = b$$

$$\begin{cases}
A = \text{diag}([1, 2(h_0 + h_1), \dots, 2(h_{n-2} + h_{n-1}), 1]) \\
\quad + \text{diag}([0, h_1, \dots, h_{n-1}], 1) + \text{diag}([h_0, \dots, h_{n-2}, 0], -1) \\
x = [c_0; c_1; \dots; c_n] \\
b = \left[0; \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0); \dots; \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}); 0\right]
\end{cases}$$

Then we will get  $b_j, d_j$  by

$$\begin{cases}
b_j &= \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \\
d_j &= \frac{1}{3h_j}(c_{j+1} - c_j)
\end{cases}$$

### 3.5.2 Clamped Splines

$$Ax = b$$

$$\left\{ \begin{array}{l} A = \begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & \vdots \\ 0 & h_1 & 2(h_1 + h_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & h_{n-1} \\ 0 & \cdots & \cdots & h_{n-1} & 2h_{n-1} \end{pmatrix} \\ x = (c_0 \quad c_1 \quad \cdots \quad c_n)^T \\ b = \begin{pmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{pmatrix} \end{array} \right.$$

# 第四章 Numerical Differentiation and Integration

## 4.1 Numerical Differentiation

To approximate  $f'(x)$  ( $x_0 \in (a, b)$ ,  $f \in C^2[a, b]$ ),  $x_1 = x_0 + h \in [a, b]$ .

$$\begin{aligned}
 f(x) &= P_{0,1}(x) + \frac{(x-x_0)(x-x_1)}{2!} f''(\xi(x)) \\
 &= \frac{f(x_0)(x-x_0-h)}{-h} + \frac{f(x_0)(x-x_0)}{h} + \frac{(x-x_0)(x-x_1)}{2} f''(\xi(x)) \\
 \Rightarrow f'(x) &= \frac{f(x_0+h) - f(x_0)}{h} + \frac{2(x-x_0)-h}{2} f''(\xi(x)) + \frac{(x-x_0)(x-x_0-h)}{2} D_x(f''(\xi(x))) \\
 \Rightarrow f'(x_0) &= \frac{f(x_0+h) - f(x_0)}{h} - \frac{h}{2} f''(\xi)
 \end{aligned}$$

The above formula is known as the forward-difference formula if  $h > 0$ , and the backward-difference formula if  $h < 0$ .

**定理 4.1.1 ((n+1)-point Formula)**  $\{x_0, x_1, \dots, x_n\}$  are  $(n+1)$  dis-

*tinct numbers in interval  $I$ ,  $f \in C^{n+1}(I)$ .*

$$\begin{aligned}
 f(x) &= \sum_{k=0}^n f(x_k) L_k(x) + \prod_{k=0}^n \left( \frac{x - x_k}{k+1} \right) f^{(n+1)}(\xi(x)). \\
 \Rightarrow f'(x) &= \sum_{k=0}^n f(x_k) L'_k(x) + D_x \left[ \prod_{k=0}^n \left( \frac{x - x_k}{k+1} \right) \right] f^{(n+1)}(\xi(x)) \\
 &\quad + \prod_{k=0}^n \left( \frac{x - x_k}{k+1} \right) D_x [f^{(n+1)}(\xi(x))]. \\
 \Rightarrow f'(x_j) &= \sum_{k=0}^n f(x_k) L'_k(x_j) + \frac{f^{(n+1)}(\xi(x_j))}{(n+1)!} \prod_{\substack{k=0 \\ k \neq j}}^n (x_j - x_k).
 \end{aligned}$$

#### 4.1.1 Three-Point Formulas

If the nodes are equally spaced,  $x_1 = x_0 + h$ ,  $x_2 = x_0 + 2h$ , then

- Three-Point Formula

$$f'(x_0) = \frac{1}{h} \left[ \frac{1}{2} f(x_0) - 2f(x_0 + h) + \frac{3}{2} f(x_0 + 2h) \right] + \frac{h^2}{3} f^{(3)}(\xi_0).$$

- Three-Point Endpoint Formula

$$f'(x_0) = \frac{1}{h} \left[ -\frac{3}{2} f(x_0) + 2f(x_0 + h) - \frac{1}{2} f(x_0 + 2h) \right] + \frac{h^2}{3} f^{(3)}(\xi_1).$$

- Three-Point Midpoint Formula

$$f'(x_0) = \frac{1}{h} \left[ -\frac{1}{2} f(x_0) + \frac{1}{2} f(x_0 + 2h) \right] - \frac{h^2}{6} f^{(3)}(\xi_2).$$

#### 4.1.2 Five-Point Formulas

- Five-Point Midpoint Formula

$$\begin{aligned}
 f'(x_0) &= \frac{1}{12h} [f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) \\
 &\quad - f(x_0 + 2h)] + \frac{h^4}{30} f^{(5)}(\xi).
 \end{aligned}$$

- Five-Point Endpoint Formula

$$\begin{aligned}
 f'(x_0) &= \frac{1}{12h} [-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) \\
 &\quad + 16f(x_0 + 3h) - 3f(x_0 + 4h)] + \frac{h^4}{5} f^{(5)}(\xi).
 \end{aligned}$$

### 4.1.3 Second Derivative Midpoint Formula

$$f''(x_0) = \frac{1}{h} [f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{12} f^{(4)}(\xi)$$

If  $f^{(4)}$  is continuous on  $[x_0 - h, x_0 + h]$ , it is also bounded, and the approximation is  $O(h^2)$ .

### 4.1.4 Round-Off Error Instability

$$f'(x_0) = \frac{1}{2h} [f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6} f^{(3)}(\xi_1)$$

Suppose that in evaluating  $f(x_0 + h)$  and  $f(x_0 - h)$  we encounter round-off errors  $e(x_0 + h)$  and  $e(x_0 - h)$ .

$$f(x_0 + h) = \tilde{f}(x_0 + h) + e(x_0 + h) \quad \text{and} \quad f(x_0 - h) = \tilde{f}(x_0 - h) + e(x_0 - h)$$

The total error in the approximation

$$\begin{aligned} \left| f'(x_0) - \frac{\tilde{f}(x_0 + h) - \tilde{f}(x_0 - h)}{2h} \right| &= \left| \frac{e(x_0 + h) - e(x_0 - h)}{2h} - \frac{h^2}{6} f^{(3)}(\xi_1) \right| \\ &\leq \frac{\varepsilon}{h} + \frac{h^2}{6} M, \end{aligned}$$

where  $e(x_0 \pm h)$  are bounded by  $\varepsilon > 0$  and  $f^{(3)}$  are bounded by  $M > 0$ . There is an optimal  $h$  such that the bound is small.

## 4.2 Richardson's Extrapolation

Suppose that for each number  $h \neq 0$ , we have a formula  $N_1(h)$  that approximates an unknown constant  $M$  with truncation error  $O(h)$

$$\begin{aligned} M - N_1(h) &= K_1 h + K_2 h^2 + K_3 h^3 + \dots \\ M - N_1\left(\frac{h}{2}\right) &= K_1 \frac{h}{2} + K_2 \frac{h^2}{4} + K_3 \frac{h^3}{8} + \dots \end{aligned}$$

If we subtract the first equation from the second equation, then we'll get

$$\begin{aligned} M &= N_1 \left( \frac{h}{2} \right) + \left[ N_1 \left( \frac{h}{2} - N_1(h) \right) \right] + K_2 \left( \frac{h^2}{2} - h^2 \right) + K_3 \left( \frac{h^3}{4} - h^3 \right) + \dots \\ \Rightarrow N_2(h) &= N_1 \left( \frac{h}{2} \right) + \left[ N_1 \left( \frac{h}{2} - N_1(h) \right) \right] \\ \Rightarrow M &= N_2(h) - \frac{K_2}{2} h^2 - \frac{3K_3}{4} h^3 + \dots \quad \text{with truncation error } O(h^2) \end{aligned}$$

$$\begin{aligned} M &= N_2 \left( \frac{h}{2} \right) + \left[ N_2 \left( \frac{h}{2} - N_2(h) \right) \right] / 3 + \frac{K_3}{8} h^3 + \dots \\ \Rightarrow N_3(h) &= N_2 \left( \frac{h}{2} \right) + \left[ N_2 \left( \frac{h}{2} - N_2(h) \right) \right] / 3 \\ \Rightarrow M &= N_3(h) - \frac{K_3}{8} h^3 + \frac{7K_3}{48} h^4 + \dots \quad \text{with truncation error } O(h^3) \end{aligned}$$

### 4.3 Elements of Numerical Integration

$$\begin{aligned} \int_a^b f(x) dx &= \int_a^b \sum_{i=0}^n f(x_i) L_i(x) dx + \int_a^b \prod_{i=0}^n (x - x_i) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} dx \\ &= \int_a^b a_i f(x_i) dx + \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^n (x - x_i) f^{(n+1)}(\xi(x)) dx, \end{aligned}$$

where  $a_i = \int_a^b L_i(x) dx$  for each  $i = 0, 1, \dots, n$ .

#### 4.3.1 The Trapezoidal Rule

To derive the Trapezoidal rule for approximating  $\int_a^b f(x) dx$ , let  $x_0 = a$ ,  $x_1 = b$ ,  $h = b - a$ .

$$\begin{aligned} \int_a^b f(x) dx &= \int_a^b \left[ \frac{(x - x_0)}{(x_1 - x_0)f(x_1)} + \frac{(x - x_1)}{(x_0 - x_1)f(x_0)} \right] dx + \frac{1}{2} \int_a^b (x - x_0)(x - x_1) f''(\xi(x)) dx \\ &= \int_{x_0}^{x_1} \frac{(x - x_0)f(x_1) - (x - x_1)f(x_0)}{x_1 - x_0} dx + \frac{f''(\xi)}{2} \left[ \frac{x^3}{3} - \frac{(x_0 + x_1)}{2} x^2 + x_0 x_1 x \right]_{x_0}^{x_1} \\ &= \left[ \frac{(x - x_0)^2 f(x_1) - (x - x_1)^2 f(x_0)}{2(x_0 - x_1)} \right]_{x_0}^{x_1} - \frac{h^3}{12} f''(\xi) \\ &= \frac{h}{2} [f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi). \end{aligned}$$

### 4.3.2 Simpson's Rule

$$\begin{aligned}
 \int_{x_0}^{x_2} f(x)dx &= \left[ f(x_1)(x - x_1) + \frac{f'(x_1)}{2}(x - x_1)^2 + \frac{f''(x_1)}{6}(x - x_1)^3 + \frac{f^{(3)}(x_1)}{24}(x - x_1)^4 \right]_{x_0}^{x_2} \\
 &\quad + \frac{1}{24} \int_{x_0}^{x_2} f^{(4)}(\xi(x))(x - x_1)^4 dx \\
 &= 2hf(x_1) + \frac{h^3}{3}f''(x_1) + \frac{f^{(4)}(\xi_1)}{60}h^5 \\
 &= 2hf(x_1) + \frac{h^3}{3} \left\{ \frac{1}{h^2} [f(x_0) - 2f(x_1) + f(x_2)] - \frac{h^2}{12}f^{(4)}(\xi_2) \right\} + \frac{f^{(4)}(\xi_1)}{60}h^5 \\
 &= \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{12} \left[ \frac{1}{3}f^{(4)}(\xi_2) - \frac{1}{5}f^{(4)}(\xi_1) \right] \\
 &= \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90}f^{(4)}(\xi)
 \end{aligned}$$

### 4.3.3 Measuring Precision

**定义 4.3.1 (The degree of accuracy or precision)**

*The largest positive integer  $n$  such that the formula is exact for  $x^k$  for  $k = 0, 1, \dots, n$ .*

### 4.3.4 Newton-Cotes Formulas

The Trapezoidal and Simpson's rules are examples of a class of methods known as Newton-Cotes formulas. There are two types of Newton-Cotes formulas, open and closed.

#### Closed Newton-Cotes Formulas

The  $(n + 1)$ -point closed Newton-Cotes uses nodes  $x_i = x_0 + ih$ , for  $i = 0, 1, \dots, n$ , where  $x_0 = a$ ,  $x_n = b$  and  $h = (b - a)/n$ .

**定理 4.3.1** *If  $n$  is even and  $f \in C^{n+2}[a, b]$*

$$\int_a^b f(x)dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1) \dots (t-n) dt.$$

*If  $n$  is odd and  $f \in C^{n+1}[a, b]$*

$$\int_a^b f(x)dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+2} f^{(n+1)}(\xi)}{(n+1)!} \int_0^n t(t-1) \dots (t-n) dt.$$

$$\xi \in (a, b).$$

n=1 Trapezoidal rule

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2} [f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi)$$

n=2 Simpson's rule

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90} f^{(4)}(\xi)$$

n=3 Simpson's Three-Eighths rule

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] - \frac{3h^5}{80} f^{(4)}(\xi)$$

n=4

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] - \frac{8h^7}{945} f^{(6)}(\xi)$$

### Open Newton-Cotes Formulas

The open Newton-Cotes formulas do not include the endpoints of  $[a, b]$  as nodes. They use the nodes  $x_i = x_0 + ih$ , for  $i = 0, 1, \dots, n$ , where  $h = (b - a)/(n + 2)$  and  $x_0 = a + h$ ,  $x_n = b - h$ .



**定理 4.3.2** *If  $n$  is even and  $f \in C^{n+2}[a, b]$*

$$\int_a^b f(x)dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_{-1}^{n+1} t^2(t-1)\dots(t-n)dt.$$

*If  $n$  is odd and  $f \in C^{n+1}[a, b]$*

$$\int_a^b f(x)dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+2} f^{(n+1)}(\xi)}{(n+1)!} \int_{-1}^{n+1} t(t-1)\dots(t-n)dt.$$

$$\xi \in (a, b).$$

n=0 Midpoint rule

$$\int_{x_{-1}}^{x_1} f(x)dx = 2hf(x_0) + \frac{h^3}{3} f''(\xi)$$

n=1

$$\int_{x_{-1}}^{x_2} f(x)dx = \frac{3h}{2} [f(x_0) + f(x_1)] - \frac{3h^3}{4} f''(\xi)$$

n=2

$$\int_{x_{-1}}^{x_3} f(x)dx = \frac{4h}{3} [2f(x_0) - f(x_1) + 2f(x_2)] + \frac{14h^5}{45} f^{(4)}(\xi)$$

n=3

$$\int_{x_{-1}}^{x_4} f(x)dx = \frac{5h}{24} [11f(x_0) + f(x_1) + f(x_2) + 11f(x_3)] + \frac{95h^7}{144} f^{(4)}(\xi)$$

## 4.4 Composite Numerical Integration

To calculate an arbitrary integral  $\int_a^b f(x)dx$ , choose an even integer  $n$ , subdivide the interval  $[a, b]$  into  $n$  subinterval, and apply Simpson's rule on

each consecutive pair of subintervals. With  $h = (b - a)/h$  and  $x_j = a + jh$ , for  $j = 0, 1, \dots, n$ , we have

$$\begin{aligned}
 \int_a^b f(x)dx &= \sum_{j=1}^{n/2} \int_{x_{2j-2}}^{x_{2j}} f(x)dx \\
 &= \sum_{j=1}^{n/2} \left\{ \frac{h}{3} [f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})] - \frac{h^5}{90} f^{(4)}(\xi_j) \right\} \\
 &= \frac{h}{3} \left[ f(x_0) + 2 \sum_{j=1}^{n/2-1} f(x_{2j}) + 4 \sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n) \right] - \frac{h^5}{90} \sum_{j=1}^{n/2} f^{(4)}(\xi_j) \\
 &= \frac{h}{3} [\dots] - \frac{h^5}{90} \left( \frac{n}{2} \right) f^{(4)}(\mu) \\
 &= \frac{h}{3} [\dots] - \frac{(b-a)}{180} h^4 f^{(4)}(\mu)
 \end{aligned}$$

**定理 4.4.1 (Composite Simpson's Rule)** Let  $f \in C^4[a, b]$ ,  $n$  be even,  $h = \frac{b-a}{n}$ , and  $x_j = a + jh$  for  $j = 0, 1, \dots, n$ . There exists a  $\mu \in (a, b)$  s.t. with  $n$  subintervals

$$\begin{aligned}
 \int_a^b f(x)dx &= \frac{h}{3} \left[ f(a) + 2 \sum_{j=1}^{n/2-1} f(x_{2j}) + 4 \sum_{j=1}^{n/2} f(x_{2j-1}) + f(b) \right] \\
 &\quad - \frac{b-a}{180} h^4 f^{(4)}(\mu).
 \end{aligned}$$

**定理 4.4.2 (Composite Trapezoidal Rule)** Let  $f \in C^2[a, b]$ ,  $n$  be even,  $h = \frac{b-a}{n}$ , and  $x_j = a + jh$  for  $j = 0, 1, \dots, n$ . There exists a  $\mu \in (a, b)$  s.t. with  $n$  subintervals

$$\int_a^b f(x)dx = \frac{h}{2} \left[ f(a) + 2 \sum_{j=1}^{n-1} f(x_j) + f(b) \right] - \frac{b-a}{12} h^2 f''(\mu).$$

**定理 4.4.3 (Composite Midpoint Rule)** Let  $f \in C^2[a, b]$ ,  $n$  be even,  $h = \frac{b-a}{n+2}$ , and  $x_j = a + (j+1)h$  for  $j = -1, 0, \dots, n+1$ . There exists  $a \mu \in (a, b)$  s.t. with  $n+2$  subintervals

$$\int_a^b f(x) dx = 2h \sum_{j=0}^{n/2} f(x_{2j}) + \frac{b-a}{6} h^2 f''(\mu).$$

#### 4.4.1 Round-off Error Stability

$$\begin{aligned} e(h) &= \left| \frac{h}{3} \left[ e_0 + 2 \sum_{j=1}^{n/2-1} e_{2j} + 4 \sum_{j=1}^{n/2} e_{2j-1} + e_n \right] \right| \\ &\leq \frac{h}{3} \left[ |e_0| + 2 \sum_{j=1}^{n/2-1} |e_{2j}| + 4 \sum_{j=1}^{n/2} |e_{2j-1}| + |e_n| \right] \\ &\leq \frac{h}{3} \left[ \varepsilon + 2\left(\frac{n}{2} - 1\right)\varepsilon + 4\left(\frac{n}{2}\right)\varepsilon + \varepsilon \right] = \frac{h}{3} 3h\varepsilon = nh\varepsilon \\ &= (b-a)\varepsilon. \end{aligned}$$

If the round-off errors are uniformly bounded by  $\varepsilon$ .

### 4.5 Romberg Integration

#### Romberg

To approximate the integral  $I = \int_a^b f(x) dx$ , select an integer  $n > 0$ .

**INPUT** endpoints  $a, b$ ; integer  $n$ .

**OUTPUT** an array  $R$ . (Compute  $R$  by rows; only the last 2 rows are saved in storage.)

**Step 1** Set  $h = b - a$ ;  
 $R_{1,1} = \frac{h}{2}(f(a) + f(b)).$

**Step 2** OUTPUT  $(R_{1,1})$ .

**Step 3** For  $i = 2, \dots, n$  do Steps 4–8.

$$\textbf{Step 4} \quad \text{Set } R_{2,1} = \frac{1}{2} \left[ R_{1,1} + h \sum_{k=1}^{2^{i-2}} f(a + (k - 0.5)h) \right].$$

(Approximation from Trapezoidal method.)

$$\textbf{Step 5} \quad \text{For } j = 2, \dots, i \\ \text{set } R_{2,j} = R_{2,j-1} + \frac{R_{2,j-1} - R_{1,j-1}}{4^{j-1} - 1}. \quad (\text{Extrapolation.})$$

**Step 6** OUTPUT ( $R_{2,j}$  for  $j = 1, 2, \dots, i$ ).

**Step 7** Set  $h = h/2$ .

**Step 8** For  $j = 1, 2, \dots, i$  set  $R_{1,j} = R_{2,j}$ . (Update row 1 of  $R$ .)

**Step 9** STOP.

## 4.6 Gaussian Quadrature

### 4.6.1 Legendre Polynomial

**定理 4.6.1** Suppose  $x_1, x_2, \dots, x_n$  are the roots of the  $n$ th Legendre polynomial  $P_n(x)$  and that for  $i = 1, 2, \dots, n$  the number  $C_i$  are defined by

$$C_i = \int_{-1}^1 \prod_{\substack{j=1 \\ j \neq i}}^n \left( \frac{x - x_j}{x_i - x_j} \right) dx.$$

If  $P(x)$  is any polynomial of degree less than  $2n$ , then

$$\int_{-1}^1 P(x) dx = \sum_{i=1}^n C_i P(x_i)$$

[Proof]

(1)  $P(x)$  is of degree less than  $n$ .

$$\begin{aligned} \int_{-1}^1 P(x) dx &= \int_{-1}^1 \sum_{i=1}^n P(x_i) L_i(x) dx = \int_{-1}^1 \sum_{i=1}^n \prod_{\substack{j=1 \\ j \neq i}}^n \left( \frac{x - x_j}{x_i - x_j} \right) P(x_i) dx \\ &= \sum_{i=1}^n \left[ \int_{-1}^1 \prod_{\substack{j=1 \\ j \neq i}}^n \left( \frac{x - x_j}{x_i - x_j} \right) dx \right] P(x_i) = \sum_{i=1}^n C_i P(x_i). \end{aligned}$$

(2)  $P(x)$  is of degree at least  $n$  but less than  $2n$ .

$$P(x_i) = Q(x_i)P_n(x_i) + R(x_i) = R(x_i) \quad (\text{degree less than } n).$$

□

#### 4.6.2 Gaussian Quadrature on Arbitrary Intervals

An integral  $\int_a^b f(x)dx$  over an arbitrary  $[a, b]$  can be transformed into an integral over  $[-1, 1]$

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

# 第五章 Initial-Value Problems for Ordinary Differential Equations

## 5.1 The Elementary Theory of Initial-Value Problems

**定义 5.1.1 (Lipschitz Condition)** *A function  $f(t, y)$  is said to satisfy a Lipschitz condition in the variable  $Y$  on a set  $D \subset \mathbb{R}^2$  if a constant  $L > 0$  exists with*

$$|f(t, y_1) - f(t, y_2)| \leq L |y_1 - y_2|$$

*whenever  $(t_1, y_1), (t_2, y_2)$  are in  $D$ . The constant  $L$  is called a Lipschitz constant for  $f$ .*

**定义 5.1.2 (Convex)** *A set  $D \subset \mathbb{R}^2$  is said to be convex if whenever  $(t_1, y_1), (t_2, y_2) \in D$ , then for every  $\lambda \in [0, 1]$ ,*

$$((1 - \lambda)t_1 + \lambda t_2, (1 - \lambda)y_1 + \lambda y_2) \in D.$$

**定理 5.1.1** *Suppose  $f(t, y)$  is defined on a convex set  $D \subset \mathbb{R}^2$ , if a*

constant  $L > 0$  exists with

$$\left| \frac{\partial f}{\partial y}(t, y) \right| \leq L$$

for all  $(t, y) \in D$ , then  $f$  satisfies a Lipschitz condition on  $D$  in the variable  $y$  with Lipschitz constant  $L$ .

**定理 5.1.2** Suppose that  $D = \{(t, y) | a \leq t \leq b, y \in \mathbb{R}\}$  and  $f(t, y)$  is continuous on  $D$ . If  $f$  satisfies a Lipschitz condition on  $D$  in the variable  $y$ , then the initial-value problem

$$\begin{cases} y'(t) = f(t, y) & a \leq t \leq b \\ y(a) = \alpha \end{cases}$$

has a unique solution  $y(t)$  for  $a \leq t \leq b$ .

### 5.1.1 Well-Posed Problems

**定理 5.1.3 (Well-Posed)** Suppose that  $D = \{(t, y) | a \leq t \leq b, y \in \mathbb{R}\}$  and  $f(t, y)$ , if  $f$  is continuous and satisfies a Lipschitz condition in the variable  $y$  on the set  $D$ , then the initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

is well-posed.

## 5.2 Euler's Method

The object of *Euler's method* is to obtain approximations to the well-posed initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

We will use Taylor's Theorem to derive Euler's method. Suppose that  $y(t)$ , the unique solution has two continuous derivations on  $[a, b]$ , so that for each

$i = 0, 1, \dots, N - 1$

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2}y''(\xi_i)$$

for some number  $\xi_i$  in  $t_i, t_{i+1}$ . Because  $h = t_{i+1} - t_i$ , we have

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(\xi_i)$$

Euler's method constructs  $\omega_i \approx y(t_i)$ , for each  $i = 1, 2, \dots, N$ , by deleting the remainder term, then Euler's method is

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + hf(t_i, \omega_i) \quad \text{for } i = 0, 1, \dots, N - 1 \end{cases}$$

### 5.2.1 Errors Bounds for Euler's Method

**定理 5.2.1** Suppose  $f$  is continuous and satisfies a Lipschitz condition with constant  $L$  on

$$D = \{(t, y) | a \leq t \leq b, y \in \mathbb{R}\}$$

and that a constant  $M$  exists with

$$|y''(t)| \leq M, \quad \text{for all } t \in [a, b]$$

where  $y(t)$  denotes the unique solution to the initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

Let  $\omega_0, \dots, \omega_N$  be the approximations generated by Euler's method for some positive integer  $N$ , then for each  $i = 0, 1, \dots, N$

$$|y(t_i) - \omega_i| \leq \frac{hM}{2L} [e^{L(t_i-a)} - 1].$$



[Proof]

$$\begin{aligned}
 |y_{i+1} - \omega_{i+1}| &\leq |y_i - \omega_i| + h |f(t_i, y_i) - f(t_i, \omega_i)| + \frac{h^2}{2} |y''(\xi_i)| \\
 &\leq (1 + hL) |y_i - \omega_i| + \frac{h^2 M}{2} \\
 &\leq e^{(i+1)hL} (|y_0 - \omega_0| + \frac{h^2 M}{2hL}) - \frac{h^2 M}{2hL} \\
 &= \frac{hM}{2L} (e^{(t_{i+1}-a)L} - 1).
 \end{aligned}$$

□

**定理 5.2.2** If  $u_0, u_1, \dots, u_N$  be the approximations and  $|\delta_i| < \delta$ , then

$$|y(t_i) - u_i| \leq \frac{1}{L} \left( \frac{hM}{2} + \frac{\delta}{h} \right) [e^{L(t_i-a)} - 1] + \delta e^{L(t_i-a)}$$

The minimal value of  $E(f)$  occurs when  $h = \sqrt{\frac{2\delta}{M}}$

### 5.3 Higher-Order Taylor Method

**定义 5.3.1 (Local Truncation Error)** The difference method

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + h\phi(t_i, \omega_i) \end{cases} \quad \text{for each } i = 0, 1, \dots, N-1$$

has local truncation error

$$\tau_{i+1}(x) = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i)$$

for each  $i = 0, 1, \dots, N-1$  where  $y_i$  and  $y_{i+1}$  denote the accuracy at a specific step, assuming that the method was exact at the previous step.

Euler's method has  $\tau_{i+1} = \frac{h}{2} y''(\xi_i)$ , so the local truncation error in Euler's method is  $O(h)$ .

### 5.3.1 Taylor Method of Order $n$

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + h\phi(t_i, \omega_i) \quad \text{for each } i = 0, 1, \dots, N-1 \end{cases}$$

where  $T^{(n)}(t_i, \omega_i) = f(t_i, \omega_i) + \frac{h}{2}f'(t_i, \omega_i) + \dots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, \omega_i)$ .

**定理 5.3.1** *If Taylor's method of order  $n$  is used to approximate the solution to*

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

*with step size  $h$  and if  $y \in C^{n+1}[a, b]$ , then the local truncation error is  $O(h^n)$ .*

[Proof]

$$\begin{aligned} y_{i+1} &= y_i + hf(t_i, y_i) + \dots + \frac{h^n}{n!}f^{(n-1)}(t_i, y_i) + \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i)) \\ \Rightarrow \tau_{i+1}(h) &= \frac{y_{i+1} - y_i}{h} - T^{(n)}(t_i, y_i) = \frac{h^n}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i)) \end{aligned}$$

for each  $i = 0, 1, \dots, N-1$ . Since  $y \in C^{n+1}[a, b]$ , we have  $y^{(n+1)}(t) = f^{(n)}(t, y(t))$  bounded on  $[a, b]$  and  $\tau_i(h) = O(h^n)$  for each  $i = 1, 2, \dots, N$ .

□

## 5.4 Runge-Kutta Methods

**定理 5.4.1** *Suppose that  $f(t, y)$  and all its partial derivatives of order less or equal to  $n+1$  are continuous on  $D = \{(t, y) | a \leq t \leq b, c \leq y \leq d\}$  ( $D = [a, b] \times [c, d]$ ) and let  $(t_0, y_0) \in D$ . For every  $(t, y) \in D$ , there exists  $\xi$  between  $t$  and  $t_0$  and  $\mu$  between  $y$  and  $y_0$  with*

$$f(t, y) = P_n(t, y) + R_n(t, y)$$

where

$$P_n(t, y) = f(t_0, y_0) + \left[ (t - t_0) \frac{\partial f}{\partial t}(t_0, y_0) + (y - y_0) \frac{\partial f}{\partial y}(t_0, y_0) \right] \\ + \left[ \frac{(t - t_0)^2}{2} \frac{\partial^2 f}{\partial t^2}(t_0, y_0) + (t - t_0)(y - y_0) \frac{\partial^2 f}{\partial t \partial y}(t_0, y_0) + \frac{(y - y_0)^2}{2} \frac{\partial^2 f}{\partial y^2}(t_0, y_0) \right] \\ + \left[ \frac{1}{n!} \sum_{j=0}^{n+1} \binom{n}{j} (t - t_0)^{n-j} (y - y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1} \partial y^j}(t_0, y_0) \right]$$

$$\text{and } R_n(t, y) = \frac{1}{(n+1)!} \sum_{j=0}^{n+1} \binom{n+1}{j} (t - t_0)^{n+1-j} (y - y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1} \partial y^j}(\xi, \mu)$$

The function  $P_n(t, y)$  is called the  $n$ th Taylor polynomial in two variables for the function  $f$  about  $(t_0, y_0)$ , and  $R_n(t, y)$  is the remainder term associated with  $P_n(t, y)$ .

#### 5.4.1 Runge-Kutta Methods of Order Two

$$\begin{cases} y_{n+1} = y_n + h(c_1 k_1 + c_2 k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \lambda_2 h, y_n + \mu_{21} h k_1) \end{cases}$$

$$\begin{aligned} T_{n+1} &= y(x_{n+1}) - y(x_n) - h[c_1 f(x_n, y_n) + c_2 f(x_n + \lambda_2 h, y_n + \mu_{21} h f_n)] \\ &= h f_n + \frac{h^2}{2} [f'_x(x_n, y_n) + f'_y(x_n, y_n) f_n] \\ &\quad - h [c_1 f_n + c_2 (f_n + \lambda_2 f'_x(x_n, y_n) h + \mu_{21} f'_y(x_n, y_n) f_n h)] + O(h^3) \\ &= (1 - c_1 - c_2) f_n h + \left( \frac{1}{2} - c_2 \lambda_2 \right) f'_x(x_n, y_n) h^2 \\ &\quad + \left( \frac{1}{2} - c_2 \mu_{21} \right) f'_y(x_n, y_n) f_n h^2 + O(h^3) \\ &\Rightarrow y_{n+1} = y_n + h f \left( x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n) \right) \end{aligned}$$

### 5.4.2 Midpoint Method

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + hf\left(t_i + \frac{h}{2}, \omega_i + \frac{h}{2}f(t_i, \omega_i)\right) \end{cases} \quad \text{for } i = 0, \dots, N-1$$

Local truncation error:  $O(h^2)$ .

### 5.4.3 Modified Euler Method

$$\begin{cases} \omega_0 = \alpha \\ \omega_{i+1} = \omega_i + \frac{h}{2} [f(t_i, \omega_i), f(t_{i+1}, \omega_i + hf(t_i, \omega_i))] \end{cases} \quad \text{for } i = 0, \dots, N-1$$

### 5.4.4 Higher-Order Runge-Kutta Methods

Runge-Kutta Order Three:

$$\begin{cases} \omega_0 = \alpha \\ k_1 = hf(t_i, \omega_i) \\ k_2 = hf\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1\right) \\ k_3 = hf(t_i + h, \omega_i - k_1 + 2k_2) \\ \omega_{i+1} = \omega_i + \frac{1}{6}(k_1 + 4k_2 + k_3) \end{cases}$$

Runge-Kutta Order Four:

$$\begin{cases} \omega_0 = \alpha \\ k_1 = hf(t_i, \omega_i) \\ k_2 = hf\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1\right) \\ k_3 = hf\left(t_i + \frac{h}{2}, \omega_i - k_1 + \frac{1}{2}k_2\right) \\ k_4 = hf(t_i + h, \omega_i + k_3) \\ \omega_{i+1} = \omega_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

### 5.4.5 Computational Comparisons

Evaluations per step	$n \in [2, 4]$	$n \in [5, 7]$	$n \in [8, 9]$	$n \in [10, \infty]$
Best possible local truncation error	$O(h^n)$	$O(h^{n-1})$	$O(h^{n-2})$	$O(h^3)$

## 5.5 Error Control and the Runge-Kutta-Fehlberg Method

### 5.5.1 收敛性与相容性

**定义 5.5.1 (收敛)** 若一种数值方法, 对于固定的  $x_n = x_0 + nh$ , 当  $h \rightarrow 0$  时有  $y_n \rightarrow y(x_n)$ , 其中  $y(x)$  是初值问题的精确解, 则称该方法是收敛的

**定理 5.5.1 (整体截断误差)** 假设单步法具有  $p$  阶精度, 且增量函数  $\varphi(x, y, h)$  关于  $y$  满足 Lipschitz 条件

$$|\varphi(x, y, h) - \varphi(x, \bar{y}, h)| \leq L_\varphi |y - \bar{y}|.$$

又设初值  $y_0$  是准确的, 即  $y_0 = y(x_0)$ , 则其整体截断误差

$$y(x_n) - y_n = O(h^p).$$

**[Proof]** 设以  $\bar{y}_{n+1}$  表示取  $y_n = y(x_n)$  用公式求得的结果, 即

$$\bar{y}_{n+1} = y(x_n) + h\varphi(x_n, y(x_n), h),$$

则局部截断误差满足, 存在常数  $C$ , 使 ( $p$  阶精度)

$$|y(x_{n+1}) - \bar{y}_{n+1}| \leq Ch^{p+1}$$

所以有

$$\begin{aligned} |\bar{y}_{n+1} - y_{n+1}| &\leq |y(x_n) - y_n| + h |\varphi(x_n, y(x_n), h) - \varphi(x_n, y_n, h)| \\ &\leq (1 + hL_\varphi) |y(x_n) - y_n|, \end{aligned}$$

从而有

$$\begin{aligned} |y(x_{n+1}) - y_{n+1}| &\leq |\bar{y}_{n+1} - y_{n+1}| + |y(x_{n+1}) - \bar{y}_{n+1}| \\ &\leq (1 + hL_\varphi) |y(x_n) - y_n| + Ch^{p+1} \end{aligned}$$

即对整体截断误差  $e_n = y(x_n) - y_n$  成立下列递推关系

$$\begin{aligned} |e_n| &\leq (1 + hL_\varphi) |e_{n-1}| + Ch^{p+1} \\ &\leq (1 + hL_\varphi)^n |e_0| + \frac{Ch^p}{L_\varphi} [(1 + hL_\varphi)^n - 1] \end{aligned}$$

再注意到当  $x_n - x_0 = nh \leq T$  时,

$$(1 + hL_\varphi)^n \leq (e^{hL_\varphi})^n \leq e^{TL_\varphi}$$

最终有

$$|e_n| \leq |e_0| e^{TL_\varphi} + \frac{Ch^p}{L_\varphi} (e^{TL_\varphi} - 1)$$

由此可以断定, 如果初值准确, 即  $e_0 = 0$ , 证毕。  $\square$

**定义 5.5.2 相容** 若单步法的增量函数  $\varphi$  满足  $\varphi(x, y, 0) = f(x, y)$ , 则称单步法与初值问题是相容的。

**定义 5.5.3 稳定** 若一种数值方法在节点值  $y_n$  上大小为  $\delta$  的扰动, 于以后各节点值  $y_m (m > n)$  上产生的偏差不超过  $\delta$ , 则称该方法是稳定的。

为了只考虑数值方法本身, 通常只检验将数值方法用于解模型方程的稳定性, 模型方程为

$$y' = \lambda y.$$

其中  $\lambda$  为复数, 这个方程分析简单, 对一般方程可以通过局部线性优化转化为这种形式, 例如在  $\bar{x}, \bar{y}$  的邻域, 可展开为

$$y' = f(x, y) = f(\bar{x}, \bar{y}) + f'_x(\bar{x}, \bar{y})(x - \bar{x}) + f'_y(\bar{x}, \bar{y})(y - \bar{y}) + \cdots$$

**定义 5.5.4** 单步法对于解模型方程, 若得到的解  $y_{n+1} = E(h\lambda)y_n$ , 满足  $|E(h\lambda)| < 1$ , 则称该单步法是绝对稳定的, 在  $\mu = h\lambda$  的平面上, 使  $|E(h\lambda)| < 1$  的变量围成的区域, 称为绝对稳定域, 它与实轴的交称为绝对稳定区间。

欧拉法	$E(h\lambda) = 1 + h\lambda$
二阶 R-K 方法	$E(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2}$
三阶 R-K 方法	$E(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6}$
四阶 R-K 方法	$E(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + \frac{(h\lambda)^4}{24}$
后退欧拉法	$E(h\lambda) = \frac{1}{1-h\lambda}$
梯形法	$E(h\lambda) = \frac{2+h\lambda}{2-h\lambda}$

## 5.6 Multistep Method

**定义 5.6.1** An  $m$ -step multistep method for solving the initial-value problem

$$y' = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

] has a difference equation for finding the approximation  $\omega_{i+1}$  at the mesh point  $t_{i+1}$  represented by the following equation, where  $m$  is an integer greater than 1:

$$\begin{aligned} \omega_{i+1} = & a_{m-1}\omega_i + a_{m-2}\omega_{i-1} + \cdots + a_0\omega_{i+1-m} \\ & + h[b_m f(t_{i+1}, \omega_{i+1}) + b_{m-1}f(t_i, \omega_i) \\ & + \cdots + b_0 f(t_{i+1-m}, \omega_{i+1-m})] \end{aligned}$$

for  $i = m-1, m, \cdots, N-1$ , where  $h = \frac{b-a}{N}$ , the  $a_0, a_1, \cdots, a_{m-1}$  and  $b_0, b_1, \cdots, b_m$  are constant, and the starting values

$$\omega_0 = \alpha_0, \quad \omega_1 = \alpha_1, \cdots, \omega_{m-1} = \alpha_{m-1}$$

are specified.

$\left\{ \begin{array}{l} \text{When } b_m = 0, \text{ the method is called explicit, or open.} \\ \text{When } b_m \neq 0, \text{ the method is called implicit, or closed.} \end{array} \right.$



## 第六章 Direct Methods for Solving Linear Systems

### 6.1 Linear Systems of Equations and Pivoting Strategies

#### 6.1.1 Gaussian Elimination with Backward Substitution

$$\begin{aligned} E1 : & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\ E2 : & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\ & \vdots \\ En : & a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = a_{n,n+1} \end{aligned}$$

#### 6.1.2 Operation Counts

Multiplications / divisions

$$\sum_{i=1}^{n-1} (n-i) + (n-i)(n-i+1) = \frac{2n^3 + 3n^2 - 5n}{6}$$

**Additions / subtractions**

$$\sum_{i=1}^{n-1} (n-i)(n-i+1) = \frac{n^3 - n}{3}$$

**6.1.3 Backward substitution \*&/**

$$1 + \sum_{i=1}^{n-1} ((n-i) + 1) = \frac{n^2 + n}{2}$$

**6.1.4 Backward substitution +&-**

$$\sum_{i=1}^{n-1} ((n-i-1) + 1) = \frac{n^2 - n}{2}$$

**6.1.5 Gaussian Elimination with Partial Pivoting**

```

1 def gaussian_elimination_partial_pivoting(A, b):
2     np_result = A**-1 * b
3     print(np_result.T)
4     n = A.shape[0]
5     x = np.zeros((n,1))
6     tmp = 0
7     for k in range(n-1):
8         M = k
9         for m in range(k+1,n):
10             if A[m,k] > A[M,k]: M = m
11         A[[k,M]] = A[[M,k]]
12         b[[k,M]] = b[[M,k]]
13         for i in range(k+1,n):
14             m = A[i,k] / A[k,k]
15             for j in range(k,n):
16                 A[i,j] = A[i,j] - m*A[k,j]
17                 b[i,0] = b[i,0] - m*b[k,0]

```

```

18     x[-1,0] = b[-1,0] / A[-1,-1]
19     for i in range(n-2,-1,-1):
20         for j in range(i+1,n):
21             tmp += A[i,j] * x[j,0]
22             x[i,0] = (b[i,0] - tmp) / A[i,i]
23             tmp = 0
24     return x

```

### 6.1.6 Gaussian Elimination with Scaled Partial Pivoting

```

1 def gaussian_elimination_scaled_partial_pivoting(A, b):
2     np_result = A**-1 * b
3     print(np_result.T)
4     n = A.shape[0]
5     x = np.zeros((n,1))
6     tmp = 0
7     for k in range(n):
8         M = np.max(A[k,:])
9         A[k,:] /= M
10        b[k,0] /= M
11    for k in range(n-1):
12        for i in range(k+1,n):
13            m = A[i,k] / A[k,k]
14            for j in range(k,n):
15                A[i,j] = A[i,j] - m*A[k,j]
16            b[i,0] = b[i,0] - m*b[k,0]
17    x[-1,0] = b[-1,0] / A[-1,-1]
18    for i in range(n-2,-1,-1):
19        for j in range(i+1,n):
20            tmp += A[i,j] * x[j,0]
21            x[i,0] = (b[i,0] - tmp) / A[i,i]
22            tmp = 0
23    return x

```

## 6.2 Matrix Factorization

**定理 6.2.1** *If Gaussian elimination can be performed on the linear system  $Ax = b$  without row interchanges, then the matrix  $A$  can be factored into the product of a lower-triangular matrix  $L$  and an upper-triangular matrix  $U$ , that is  $A = LU$ , where  $m_{ji} = a_{ji}^{(i)} / a_{ii}^{(i)}$*

$$L = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{pmatrix} \quad U = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{n,n-1} & 1 \end{pmatrix}$$

## 第七章 Iterative Techniques in Matrix Algebra

### 7.1 Norms of Vectors and Matrices

**定义 7.1.1** A vector norm on  $\mathbb{R}^n$  is a function,  $\|\cdot\|$ , from  $\mathbb{R}^n$  to  $\mathbb{R}$  with the following properties.

- (i)  $\|\mathbf{x}\| \geq 0$  for all  $\mathbf{x} \in \mathbb{R}^n$ .
- (ii)  $\|\mathbf{x}\| = 0$  if and only if  $\mathbf{x} = \mathbf{0}$ .
- (iii)  $\|\alpha\mathbf{x}\| = |\alpha| \|\mathbf{x}\|$  for all  $\alpha \in \mathbb{R}$  and  $\mathbf{x} \in \mathbb{R}^n$ .
- (iv)  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$  for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ .

**定义 7.1.2** The  $l_1$ ,  $l_2$ ,  $l_\infty$  norms for the vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)^t$  are defined by

- $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$
- $\|\mathbf{x}\|_2 = [\sum_{i=1}^n x_i^2]^{1/2}$
- $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$

**定理 7.1.1** The sequence of vectors  $\mathbf{x}^{(k)}$  converges to  $\mathbf{x}$  in  $\mathbb{R}^n$  with respect to the  $l_\infty$  norm if and only if  $\lim_{k \rightarrow +\infty} x_i^{(k)} = x_i$ , for each

$$i = 1, 2, \dots, n.$$

### 7.1.1 Matrix Norms and Distances

**定义 7.1.3 (Matrix Norms)** A matrix norm on the set of all  $n \times n$  matrices is a real-valued function,  $\|\cdot\|$ , defined on this set, satisfying for all  $n \times n$  matrices  $\mathbf{A}$  and  $\mathbf{B}$  and all real numbers  $\alpha$ .

$$(i) \quad \|\mathbf{A}\| \geq 0.$$

$$(ii) \quad \|\mathbf{A}\| = 0 \text{ if and only if } \mathbf{A} \text{ is } \mathbf{0}, \text{ the matrix with all 0 entries.}$$

$$(iii) \quad \|\alpha\mathbf{A}\| = |\alpha| \|\mathbf{A}\|.$$

$$(iv) \quad \|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|.$$

$$(v) \quad \|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|.$$

**定理 7.1.2** If  $\|\cdot\|$  is a vector norm on  $\mathbb{R}$ , then

$$\|\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|$$

is a matrix norm.

**定理 7.1.3** If  $\mathbf{A} = (a_{ij})$  is an  $n \times n$  matrix, then

$$\|\mathbf{A}\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|.$$

## 7.2 Eigenvalues and Eigenvectors

If  $\mathbf{A}$  is a square matrix, the *characteristic polynomial* of  $\mathbf{A}$  is defined by  $p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$ , the zeros of  $p$  are *eigenvalues*, or characteristic values, of the matrix  $\mathbf{A}$ .

### 7.2.1 Spectral Radius

**定义 7.2.1 (Spectral Radius)** The spectral radius  $\rho(\mathbf{A})$  of a matrix  $\mathbf{A}$  is defined by

$$\rho(\mathbf{A}) = \max |\lambda|, \quad \text{where } \lambda \text{ is an eigen value of } \mathbf{A}.$$

(For complex  $\lambda = \alpha + \beta i$ , we define  $|\lambda| = (\alpha^2 + \beta^2)^{1/2}$ .)

**定理 7.2.1** If  $\mathbf{A}$  is an  $n \times n$  matrix, then

- (i)  $\|\mathbf{A}\|_2 = [\rho(\mathbf{A}^t \mathbf{A})]^{1/2}$ ,
- (ii)  $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ , for any natural norm  $\|\cdot\|$ .

### 7.2.2 Convergent Matrices

**定义 7.2.2 (Convergent)** We call an  $n \times n$  matrix  $\mathbf{A}$  convergent if

$$\lim_{k \rightarrow \infty} (\mathbf{A}^k)_{ij} = 0, \quad \text{for each } i = 1, 2, \dots, n \text{ and } j = 1, 2, \dots, n.$$

**定理 7.2.2** The following statements are equivalent

- (i)  $\mathbf{A}$  is a convergent matrix.
- (ii)  $\lim_{n \rightarrow \infty} \|\mathbf{A}^n\| = 0$ , for some natural norm.
- (iii)  $\lim_{n \rightarrow \infty} \|\mathbf{A}^n\| = 0$ , for all natural norms.
- (iv)  $\rho(\mathbf{A}) < 1$ .
- (v)  $\lim_{n \rightarrow \infty} \mathbf{A}^n \mathbf{x} = 0$ , for every  $\mathbf{x}$ .

## 7.3 The Jacobi and Gauss-Siedel Iterative Techniques