Numerical Analysis

Iydon *de* SUSTeX

2018 年 11 月 15 日

目录

第零章 Preface **5**

0.1 Preface 5

第一章 Mathematical Preliminaries and Error Analysis **6**

1.1 Mathematical Preliminaries and Error Analysis 6

1.1.1 Round-off Errors and Computer Arithmetic 6

第二章 Solutions of Equations in One Variable **10**

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

第三章 Interpolation and Polynomial Approximation **18**

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

第四章 Numerical Differentiation and Integration 27

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

第五章 Initial-Value Problems for Ordinary Differential Equations 31

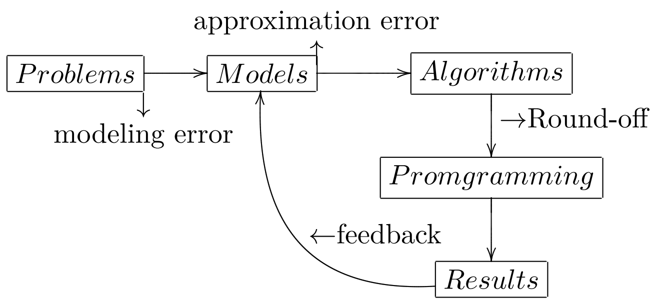
5.1 The Elementary Theory of Initial-Value Problems 31
5.1.1 Well-Posed Problems 32
5.2 Euler's Method 32
5.2.1 Errors Bounds for Euler's Method 33
5.3 Higher-Order Taylor Method 34
5.3.1 Taylor Method of Order n 35
5.4 Runge-Kutta Methods 35
5.4.1 Runge-Kutta Methods of Order Two 36
5.4.2 Midpoint Method 37
5.4.3 Modified Euler Method 37
5.4.4 Higher-Order Runge-Kutta Methods 37
5.4.5 Computational Comparisons 38
5.5 Error Control and the Runge-Kutta-Fehlberg Method 38
5.5.1 收敛性与相容性 38
5.6 Multistep Method 40

第六章 Direct Methods for Solving Linear Systems 42
6.1 Linear Systems of Equations and Pivoting Strategies 42
6.1.1 Gaussian Elimination with Backward Substitution . . 42
6.1.2 Operation Counts 42
6.1.3 Backward substitution *&/ 43
6.1.4 Backward substitution +&- 43
6.1.5 Gaussian Elimination with Partial Pivoting 43
6.1.6 Gaussian Elimination with Scaled Partial Pivoting . . 44
6.2 Matrix Factorization 45

第七章 Iterative Techniques in Matrix Algebra 46
7.1 Norms of Vectors and Matrices 46
7.1.1 Matrix Norms and Distances 47
7.2 Eigenvalues and Eigenvectors 47
7.2.1 Spectral Radius 48
7.2.2 Convergent Matrices 48
7.3 The Jacobi and Gauss-Siedel Iterative Techniques 48

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：

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

$$\begin{cases} S:\ 0/1 & signpart \\ c:\ 11\,digits & exponential\ part. \\ f:\ 52\,digits & mantissa\ part \end{cases}$$

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 $1\le d_1\le 9$，$0\le d_i\le 9$，$\forall i\ge 2$。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: $fl(y)=0.31416\times 10^1$.

π

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

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

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

$$\frac{|p-p^\star|}{p}\le 5\times 10^{-t}$$

.

Chopping floating:

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

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

Chopping:（其有效位数至少为 k-1）

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

Rounding:（其有效位数至少为 k）

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

Machine Operators

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

$$x \,\text{op}\, y = fl(fl(x) \,\text{op}\, fl(y))$$

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 : $a_0, a_1, \cdots, a_n(given); x$
output: $P_n(x)$

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

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

```

```python
    Args:
        poly: List, store the coefficient of the polynomial.
        x: Float, specify the variable in the polynomial.

    Returns:
        Float, result.

    Raises:
        If `poly' is empty, raise IndexError.
        If type(args) does not correspond, raise TypeError.
    """
    result = poly[0]
    for i in range(1, poly.__len__()):
        result = x*result + poly[i]
    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 $n \to \infty$，$\alpha_n \to \alpha$，$\beta_n \to 0$，其中 $|\alpha_n - \alpha| \le k\,|\beta_n|$（与 n 的取值无关），则称 α_n 是以 β_n 的速度收敛到 α 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* $f \in [a,b], \forall k \in f([a,b]), \exists c \in [a,b],\ s.t.\ f(c) = k$。

```
def Bisection(fun, a:float, b:float, max_step:int=128, ...
    eps:float=1e-6)->float:
    mid_last = a
    if fun(a)*fun(b) < 0:
        for i in range(0, max_step):
            mid = (a+b) / 2
            if abs(mid-mid_last)<eps or abs(fun(mid))<eps:
                print("Step: %d\nZero: %fc"%(i, mid))
                return mid
            else:
                if fun(mid)*fun(a)<0:
                    b = mid
                else:
                    a = mid
            mid_last = mid
        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 $\exists$ *root* $P\in [a,b]$，则有 $|P_n - P| \le \frac{b-a}{x^n}$。

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

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

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 $g'(x) \le \kappa < 1$。

[Proof]
存在性：

$$\begin{cases} h(a) = & g(a) - a \ge 0 \\ h(b) = & g(b) - b \le 0 \end{cases}$$

于是有 $h(a)h(b) \le 0$，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 $P_1 \neq P_2$。

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

□

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 $P_i=g(P_{i-1}),\quad i=1,2,\cdots$，则 $\lim_{n\to\infty}P_n=P$（P 为不动点）。

[Proof]

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

□

其中，寻找不动点的代码如下：

```
def fixed_point(fun, start:float=0, max_step:int=128, ...
    eps:float=1e-6)->float:
    new_val = fun(start)
    for i in range(0, max_step):
        old_val = new_val
        new_val = fun(old_val)
        if -eps<old_val-new_val<eps:
            print(i)
            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

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

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 $\{p_n\}_{n=0}^{\infty}$,

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

```python
def Newton_method(f, df, start:float=0.0, max_step:int=32,...
    sign_dig:int=6)->float:
    fun = lambda x: x - f(x)/df(x)
    return fixed_point(fun, start, max_step, sign_dig)

def fixed_point(fun, start:float, max_step:int,...
    sign_dig:int)->float:
    fl = lambda x: round(x, 100)
    eps = 10**(-sign_dig)
    new_val = fun(start)
    for i in range(0, max_step):
        old_val = fl(new_val)
```

```
        new_val = fl(fun(old_val))
        if abs(old_val-new_val)<=2*eps:
            return (i, new_val)
    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.

```python
def false_position(f, start:list, max_step:int=32, ...
    eps:float=1e-6) -> float:
    """
    False position.
    ----------------------------
    Args:
        f: Function.
        start: List of float, the first iteration point.
        max_step: Integer, max number of iteration.

    Returns:
        Float zero.
        zero | fun(zero)\\sim 0.

    Raises:
        None.
    """
    p = [i    for i in start]
    q = [f(i) for i in start]
    for i in range(max_step):
        _p = p[-1] - q[-1]*(p[-1]-p[0])/(q[-1]-q[0])
        if abs(_p-p[-1]) < eps:
            return (i, _p)
        _q = f(_p)
        if _q*q[-1] < 0:
            p[0] = p[-1]
            q[0] = q[-1]
        p[-1] = _p
        q[-1] = _q
    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 λ, α 满足以下条件：

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

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

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*

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

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

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

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

[Proof]

$$p_{n+1} - p = g(p_n) - g(p) = g'(\xi_n)(p_n - p),$$
$$\Rightarrow \lim_{n\to\infty} \frac{p_{n+1}-p}{p_n-p} = \lim_{n\to\infty} g'(\xi_n) = g'(p)$$

$\Rightarrow$ 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) $x_0, \ldots, x_n$ *are* $n+1$ *distinct numbers, then a unique polynomial* $P(x)$ *of degree at most* n *exists with*

$$f(x_k) = P(x_k) \quad for k = 0 : n$$

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

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

$$0=g^{(n+1)}(\xi)=f^{(n+1)}(\xi)-P^{(n+1)}(\xi)-[f(x)-P(x)]\frac{\mathrm{d}^{n+1}}{\mathrm{d}t^{n+1}}\left[\prod_{i=0}^{n}\frac{(t-x_i)}{x-x_i}\right]_{t=\xi}$$

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

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

□

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 $x_{m_1},x_{m_2},\cdots,x_{m_k}$ is denoted $P_{m_1,m_2,\cdots m_k}(x)$.*

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

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

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

3.2.1 Neville's Method

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

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

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 kth divided difference relative to $x_i, \cdots, x_{i+k}$ is

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

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

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

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.*

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

[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. $g^{(n)}(\xi) = 0$.

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

□

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

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

3.3.2 Forward Differences

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

In general,

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

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,

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

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

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

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}	$f[x_{-2}]$	$f[x_{-2}, x_{-1}]$	$f[x_{-2}, x_{-1}, x_0]$	$\underline{f[x_{-2}, x_{-1}, x_0, x_1]}$	$\underline{f[x_{-2}, x_{-1}, x_0, x_1, x_2]}$
x_{-1}	$f[x_{-1}]$	$\underline{f[x_{-1}, x_0]}$	$\underline{f[x_{-1}, x_0, x_1]}$	$\underline{f[x_{-1}, x_0, x_1, x_2]}$	
x_0	$\underline{f[x_0]}$	$\underline{f[x_0, x_1]}$	$f[x_0, x_1, x_2]$		
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 $M = \sum_{i=0}^{n} m_i + n$.

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

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

3.4.1 Hermite Polynomials

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

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

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

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

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

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

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

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. $\begin{cases} \textit{natural boundary:} & S''(x_0) = S''(x_n) = 0. \\ \textit{clamped boundary:} & S'(x_0) = f'(x_0),\ S'(x_n) = f'(x_n). \end{cases}$

3.5.1 Construction of a Cubic Spline

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

(1)

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

(2)

$$\begin{aligned}S_j'(x) = b_j + 2c_j(x-x_j) + 3d_j(x-x_j)^2 \\ \Rightarrow b_{j+1} = S_{j+1}'(x_{j+1}) = S_j'(x_{j+1}) \\ = b_j + 2c_jh_j + 3d_jh_j^2 \quad \text{for } j = 0:n-1\end{aligned}$$

(3)

$$\begin{aligned}S_j''(x) = 2c_j + 6d_j(x-x_j) \\ \Rightarrow 2c_{j+1} = S_{j+1}''(x_{j+1}) = S_j''(x_{j+1}) \\ = 2c_j + 6d_jh_j \quad \text{for } j = 0:n-1\end{aligned}$$

Above all, the linear system to be solved is:

$Ax = b$

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

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

$$
Ax = b
$$

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

第四章 Numerical Differentiation and Integration

4.1 Numerical Differentiation

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

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

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) $\{x_0, x_1, \ldots, x_n\}$ *are* $(n+1)$ *dis-*

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

$$f(x) = \sum_{k=0}^{n} f(x_k)L_k(x) + \prod_{k=0}^{n}\left(\frac{x-x_k}{k+1}\right) f^{(n+1)}(\xi(x)).$$

$$\Rightarrow f'(x) = \sum_{k=0}^{n} f(x_k)L_k'(x) + D_x\left[\prod_{k=0}^{n}\left(\frac{x-x_k}{k+1}\right)\right] f^{(n+1)}(\xi(x))$$

$$+ \prod_{k=0}^{n}\left(\frac{x-x_k}{k+1}\right) D_x\left[f^{(n+1)}(\xi(x))\right].$$

$$\Rightarrow f'(x_j) = \sum_{k=0}^{n} f(x_k)L_k'(x_j) + \frac{f^{(n+1)}(\xi(x_j))}{(n+1)!}\prod_{\substack{k=0\\k\neq j}}^{n}(x_j - x_k).$$

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
$$f'(x_0) = \frac{1}{12h}\left[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h)\right.$$
$$\left.-f(x_0+2h)\right] + \frac{h^4}{30}f^{(5)}(\xi).$$

- Five-Point Endpoint Formula
$$f'(x_0) = \frac{1}{12h}\left[-25f(x_0) + 48f(x_0+h) - 36f(x_0+2h)\right.$$
$$\left.+16f(x_0+3h - 3f(x_0+4h))\right] + \frac{h^4}{5}f^{(5)}(\xi).$$

4.1.3 Second Derivative Midpoint Formula

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

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

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

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

$$f(x_0+h) = \tilde{f}(x_0+h) + e(x_0+h) \qquad \textit{and} \qquad f(x_0-h) = \tilde{f}(x_0-h) + e(x_0-h)$$

The total error in the approximation

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

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 $O(h)$

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

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

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

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

4.3 Elements of Numerical Integration

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

where $a_i = \int_a^b L_i(x)\mathrm{d}x$ for each $i = 0, 1, \ldots, n$.

4.3.1 The Trapezoidal Rule

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

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* Lispschitz 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*

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

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

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

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

5.1.1 Well-Posed Problems

定理 5.1.3 (Well-Posed) *Suppose that* $D = \{(t, y)|a \le t \le b, y \in \mathbb{R}\}$ *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*

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha$$

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

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha$$

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

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

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

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

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

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

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

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

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

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

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

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

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

[Proof]

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

□

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

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

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*

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

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 $i=0,1,\cdots,N-1$ *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

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

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

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

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha$$

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

[Proof]

$$y_{i+1} = y_i + hf(t_i, y_i) + \cdots + \frac{h^n}{n!} f^{(n-1)}(t_i, y_i) + \frac{h^{n+1}}{(n+1)!} f^{(n)}(\xi_i, y(\xi_i))$$

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

for each $i = 0, 1, \cdots, N-1$. 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 $i = 1, 2, \cdots, N$. □

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 $D = \{(t, y) | a \le t \le b, c \le y \le d\}$ ($D = [a, b] \times [c, d]$) and let $(t_0, y_0) \in D$. For every $(t, y) \in D$, there exists ξ between t and t_0 and μ between y and y_0 with*

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

where

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

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

The function $P_n(t,y)$ *is called the* nth 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

$$\begin{cases}
y_{n+1} = y_n + h(c_1k_1 + c_2k_2) \\
k1 = f(x_n,y_n) \\
k1 = f(x_n + \lambda_2 h, y_n + \mu_{21}hk_1)
\end{cases}$$

$$\begin{aligned}
T_{n+1} &= y(x_{n+1}) - y(x_n) - h\left[c_1f(x_n,y_n) + c_2f(x_n+\lambda_2h, y_n+\mu_{21}hf_n)\right] \\
&= hf_n + \frac{h^2}{2}\left[f'_x(x_n,y_n) + f'_y(x_n,y_n)f_n\right] \\
&\quad - h\left[c_1f_n + c_2\left(f_n + \lambda_2f'_x(x_n,y_n)h + \mu_{21}f'_y(x_n,y_n)f_nh\right)\right] + O(h^3) \\
&= (1-c_1-c_2)f_nh + \left(\frac{1}{2} - c_2\lambda_2\right)f'_x(x_n,y_n)h^2 \\
&\quad + \left(\frac{1}{2} - c_2\mu_{21}\right)f'_y(x_n,y_n)f_nh^2 + O(h^3) \\
\Rightarrow y_{n+1} &= y_n + hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f(x_n,y_n)\right)
\end{aligned}$$

5.4.2 Midpoint Method

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

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

5.4.3 Modified Euler Method

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

5.4.4 Higher-Order Runge-Kutta Methods

Runge-Kutta Order Three:

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

Runge-Kutta Order Four:

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

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 $h \to 0$ 时有 $y_n \to y(x_n)$，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

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

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)

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

所以有

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

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

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

再注意到当 $x_n-x_0=nh\leq T$ 时，

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

最终有

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

由此可以断定，如果初值准确，即 $e_0=0$，证毕。 □

定义 5.5.2 相容 若单步法的增量函数 φ 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 δ 的扰动，于以后各节点值 $y_m(m>n)$ 上产生的偏差不超过 δ，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 λ 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*

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

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

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

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.

$$\begin{cases} \textit{When } b_m = 0, \textit{ the method is called explicit, or open.} \\ \textit{When } b_m \neq 0, \textit{ the method is called implicit, or closed.} \end{cases}$$

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

$$
\begin{aligned}
E1 :& a_{11}x_1 + a12x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\
E2 :& a_{21}x_1 + a22x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\
& \vdots \\
En :& a_{n1}x_1 + an2x_2 + \cdots + a_{nn}x_n = a_{n,n+1}
\end{aligned}
$$

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

```
def gaussian_elimination_partial_pivoting(A, b):
    np_result = A**-1 * b
    print(np_result.T)
    n = A.shape[0]
    x = np.zeros((n,1))
    tmp = 0
    for k in range(n-1):
        M = k
        for m in range(k+1,n):
            if A[m,k] > A[M,k]: M = m
        A[[k,M]] = A[[M,k]]
        b[[k,M]] = b[[M,k]]
        for i in range(k+1,n):
            m = A[i,k] / A[k,k]
            for j in range(k,n):
                A[i,j] = A[i,j] - m*A[k,j]
            b[i,0] = b[i,0] - m*b[k,0]
```

```
    x[-1,0] = b[-1,0] / A[-1,-1]
    for i in range(n-2,-1,-1):
        for j in range(i+1,n):
            tmp += A[i,j] * x[j,0]
        x[i,0] = (b[i,0] - tmp) / A[i,i]
        tmp = 0
    return x
```

6.1.6 Gaussian Elimination with Scaled Partial Pivoting

```
def gaussian_elimination_scaled_partial_pivoting(A, b):
    np_result = A**-1 * b
    print(np_result.T)
    n = A.shape[0]
    x = np.zeros((n,1))
    tmp = 0
    for k in range(n):
        M = np.max(A[k,:])
        A[k,:] /= M
        b[k,0] /= M
    for k in range(n-1):
        for i in range(k+1,n):
            m = A[i,k] / A[k,k]
            for j in range(k,n):
                A[i,j] = A[i,j] - m*A[k,j]
            b[i,0] = b[i,0] - m*b[k,0]
    x[-1,0] = b[-1,0] / A[-1,-1]
    for i in range(n-2,-1,-1):
        for j in range(i+1,n):
            tmp += A[i,j] * x[j,0]
        x[i,0] = (b[i,0] - tmp) / A[i,i]
        tmp = 0
    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 maxtrix L and an upper-triangular matrix U, that is $A = LU$, where $m_{ji} = a_{ji}^{(i)}/a_{ii}^{(i)}$*

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

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) $\|\mathbf{x}\| \geq \mathbf{0}$ *for all* $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$.

(ii) $\|\mathbf{x}\| = \mathbf{0}$ *if and only if* $\mathbf{x} = \mathbf{0}$.

(iii) $\|\alpha\mathbf{x}\| = |\alpha|\,\|\mathbf{x}\|$ *for all* $\alpha \in \mathbb{R}$ *and* $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$.

(iv) $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *for all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathbf{n}}$.

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

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

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

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

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 s a real-valued function, $\|\cdot\|$, defined on this ser, satisfying for all $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ and all real numbers α.*

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

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

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

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

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

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

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

is a matrix norm.

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

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

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*

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

(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) $\|\mathbf{A}\|_{\mathbf{2}} = [\rho(\mathbf{A^t A})]^{\mathbf{1/2}}$,

(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*

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

定理 7.2.2 *The following statements are equivalent*

(i) $\mathbf{A}$ *is a convergent matrix.*

(ii) $\lim_{n\to\infty} \|\mathbf{A^n}\| = \mathbf{0}$*, for some natural norm.*

(iii) $\lim_{n\to\infty} \|\mathbf{A^n}\| = \mathbf{0}$*, for all natural norms.*

(iv) $\rho(\mathbf{A}) < \mathbf{1}$.

(v) $\lim_{n\to\infty} \mathbf{A^n x} = \mathbf{0}$*, for every* $\mathbf{x}$.

7.3 The Jacobi and Gauss-Siedel Iterative Techniques