



浙江大学  
ZHEJIANG UNIVERSITY

# 数值分析方法笔记

## 数值分析

姓名 \_\_\_\_\_ Soleil

学号 \_\_\_\_\_ lalalala~

学院 \_\_\_\_\_ 信电学院 (开课学院)

2024 年 10 月 17 日

# 数值分析

## 摘要

## 1 絮论

数值分析方法，侧重的是方法，更偏向于计算

### 1.1 数值分析在做什么

1. 近似解 vs 精确解
2. 迭代算法
3. 收敛性
4. 误差分析

### 1.2 重大科学问题

1. 人工智能低能耗问题
2. 非线性与尺度的关系

### 1.3 数值分析

1. Numerical approaches for
  - (a) linear equations
  - (b) nonlinear equations
  - (c) differentiation and integration
  - (d) partial differential equations

## Motivation

- Q1: Find  $\mathbf{x}$ ?

$$\mathbf{Ax} = \mathbf{b}$$

- Q2: Find  $x$ ?

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

- Q3: Wireless Channel Capacity

$$C = \int_0^{\infty} \log(1+x) \exp(-x) dx$$

- Q4: Guided depth upsampling



$$\frac{\partial D(\mathbf{p}; t)}{\partial t} = \operatorname{div}(\mathbf{W}(\mathbf{p}; t) \nabla D(\mathbf{p}; t)) \quad s.t. \quad D(\mathbf{p}; 0) = D_0(\mathbf{p})$$

3 / 19

图 1: motivation

## 2. Error analysis

# 2 Computer Arithmetic

**Basics:** 1 Byte(字节) = 8 Bits(比特)

## 2.1 数据类型

1. short real format (4 Bytes)
2. long real format (8 Bytes)
  - (a) Default data type in MATLAB, double in C language

- Long real format

- Default data type in MATLAB, 'double' in C
- Base: 2

1 bit	11 bits	52 bits
s	c	f

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

- Ex: 0 10000000011 101110...0
- Maximum /  $2^{1023} \cdot (2 - 2^{-52}) \approx 0.17977 \times 10^{309}$
- Minimum /  $2^{-1022} \cdot (1 + 0) \approx 0.22251 \times 10^{-307}$
- Overflow / underflow
- [http://en.wikipedia.org/wiki/IEEE\\_floating\\_point](http://en.wikipedia.org/wiki/IEEE_floating_point)

图 2: Long-real-format

- (b) 会发现，能表示的最大位数并不是想象中的  $2^{1024}$  是因为占用了最高位和最低位 Inf and NaN
- (c) 在有效位数截断时，有两种方式，一种直接丢弃 (way of chopping)，另一种则是在对应位加上 5 再舍去 (way of Roundoff)

- Base: 10
- $k$ -digit decimal machine number:

$$\pm 0.d_1d_2\dots d_k \times 10^n, \quad 1 \leq d_1 \leq 9, \quad 0 \leq d_i \leq 9$$

- Any positive number within the numerical range can be written as

$$y = 0.d_1d_2\dots d_k d_{k+1} d_{k+2} \dots \times 10^n$$

- Two ways to represent  $y$  with  $k$  digits:

- ① Chopping:

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

- ② Roundoff: Add  $5 \times 10^{n-(k+1)}$  and chop:

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

- Roundoff error

图 3: ERROR-ANALYSIS

## 2.2 ERRORS and significant digits

### 2.2.1 Definitions

**定义 2.1.** If  $p^*$  is an approximation to  $p$ , the **absolute error** is  $|p - p^*|$ , and the **relative error** is  $\frac{|p - p^*|}{|p|}$ , provided that  $p \neq 0$ .

#### Significant digits:

The number  $p^*$  is said to approximate  $p$  to  $t$  **significant digits** if  $t$  is the largest nonnegative integer for which

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

$$\begin{aligned} \left| \frac{y - fl(y)}{y} \right| &= \left| \frac{0.d_1d_2\dots d_kd_{k+1}\dots \times 10^n - 0.d_1d_2\dots d_k \times 10^n}{0.d_1d_2\dots \times 10^n} \right| \\ &= \left| \frac{0.d_{k+1}d_{k+2}\dots \times 10^{n-k}}{0.d_1d_2\dots \times 10^n} \right| = \left| \frac{0.d_{k+1}d_{k+2}\dots}{0.d_1d_2\dots} \right| \times 10^{-k} \\ &\leq \frac{1}{0.1} \times 10^{-k} = 10^{-k+1} \end{aligned}$$

图 4: ErrorAnalysis.png

疑问, 这里为什么是 5 | 解答: 这里的 5 是 define, 作为误差判定的标准

- Floating Point Operations

- Machine addition:  $x \oplus y = fl(fl(x) + fl(y))$
- Subtraction:  $x \ominus y = fl(fl(x) - fl(y))$
- Multiplication:  $x \otimes y = fl(fl(x) \times fl(y))$
- division:  $x \oslash y = fl(fl(x) \div fl(y))$

- Cancellation

- Common problem: Subtraction of nearly equal numbers:

$$fl(x) = 0.d_1d_2\dots d_p\alpha_{p+1}\alpha_{p+2}\dots\alpha_k \times 10^n$$

$$fl(y) = 0.d_1d_2\dots d_p\beta_{p+1}\beta_{p+2}\dots\beta_k \times 10^n$$

gives fewer digits for significance:

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

图 5: 减法消除有效位数

Given  $x = 5/7$ ,  $u = 0.714251$ ,  $v = 98765.9$ , and  $w = 0.111111 \times 10^{-4}$ .  
Determine the five-digit chopping values of

Operation	Result	Actual value	Absolute error	Relative error
$x \ominus u$	$0.30000 \times 10^{-4}$	$0.34714 \times 10^{-4}$	$0.471 \times 10^{-5}$	0.136
$(x \ominus u) \oplus w$	$0.27000 \times 10^1$	$0.31242 \times 10^1$	0.424	0.136
$(x \ominus u) \otimes v$	$0.29629 \times 10^1$	$0.34285 \times 10^1$	0.465	0.136
$u \oplus v$	$0.98765 \times 10^5$	$0.98766 \times 10^5$	$0.161 \times 10^1$	$0.163 \times 10^{-4}$

图 6: 不同运算误差差别很大

考虑一个 equation  $x^2 + 62.10x + 1 = 0$ , 如果我们采用四舍五入算法保留一定的有效位数 (四位), 最后的误差达到了 24%, 非常高; 但是但是, 如果将无理数放在分母上, 计算误差就会降低到  $6.2 \times 10^{-4}$

### 2.3 Algorithms(收敛)

a Algorithms is a set if operation to solve a problem or approximate a solution to ze problem (迭代的收敛性)

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

收敛速度迭代

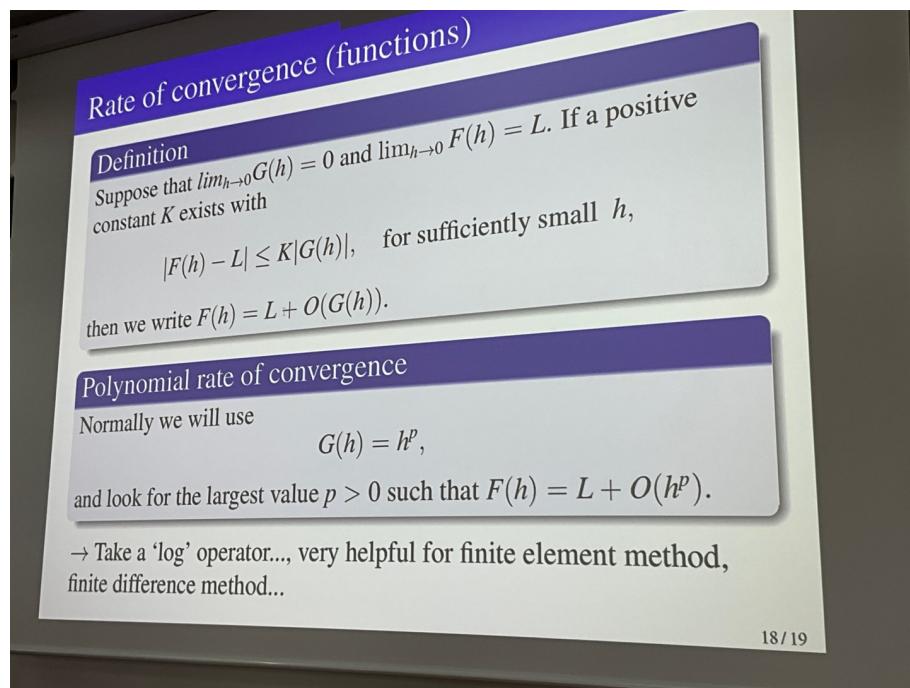


图 7: 收敛速度 2.jpg

### 3 Solutions of Equations in One Variable(单变量求根)

参考文章: 知乎

**定理 3.1.** *Rolle's Theorem*(罗尔中值定理)

**定理 3.2.** *Mean Value Theorem* (又叫拉格朗日中值定理)

**定理 3.3.** *Intermedi Value Theorem* (介值定理)

#### 3.1 The Way of Root Finding Problem

Forward problem

Inverse problem

1. 二分法 (The Bisection Method)(Because of IVT)
2. 不动点法
3. 牛顿法

#### 3.2 The Bisection Method

##### 3.2.1 Method

在区间  $(a, b)$  中有零点, 求出区间中点  $\frac{a+b}{2}$  的函数值, 记为  $f(x_1)$ , 那么当  $f(x_1) = 0$ , 证明找到零点了, 迭代结束, 不是的话, 假设  $f(a)f(x_1) < 0$ , 则在  $(a, x_1)$  中迭代求解\*, 当然也可能  $f(b)f(x_1) < 0$ , 那么就在  $(x_1, b)$  中迭代求解。

##### 3.2.2 Convergence

If the root lies at  $r$ , and after  $n$  iterations the interval is  $[a_n, b_n]$ , the length of the interval after each iteration is halved, i.e.,

$$|b_n - a_n| = \frac{|b_0 - a_0|}{2^n}.$$

Let  $\varepsilon_n$  denote the error at the  $n$ -th iteration, which is the distance between the midpoint and the actual root  $r$ . After  $n$  iterations, the error can be bounded by:

$$\varepsilon_n = |c_n - r| \leq \frac{|b_0 - a_0|}{2^n}.$$

Therefore, after  $n$  iterations, the maximum possible error decreases exponentially by a factor of  $\frac{1}{2}$  in each step, meaning the method converges \*\*linearly\*\*, with a convergence rate  $1/2$ .

$$x_n = x + O\left(\frac{1}{2^n}\right)$$

,the rate of Convergence is(收敛速度) 为

$$O\left(\frac{1}{2^n}\right)$$

### 3.3 The Fixed-Point Problem(不动点法)

For function  $f(p) = 0$ , convert it to  $f(p) = g(p) - p$ , then the problem convert to The Fixed-Point Problem

### 3.4 Well-Posed(适定性)

适定性存在、唯一且连续

#### 3.4.1 Conception (概念)

##### 1.2 不动点迭代

首先要明白什么是不动点，不动点的定义为：

若  $f(r) = r$ ，则称  $r$  为  $f(r)$  的一个不动点。

那问题来了，明明是函数求零点为什么要扯到不动点上去？

上面可以看出，二分法迭代速度慢，而不动点的优势恰恰是快速迭代求解。

对于不动点有下面的描述成立：

$f \in [a, b]$  且在  $[a, b]$  上为自身映射\* ( $f$  在  $[a, b]$  上的值域包含于  $[a, b]$ ，同济高数第一章介绍过)，则其在  $[a, b]$  中存在不动点。特别地，当  $f$  可微且  $\exists 0 < k < 1, \forall x \in [a, b], |f'(x)| \leq k$  时，不动点的存在性是唯一的。

对于不动点迭代：选择初值  $x_0$ ，构造数列  $x_n = f(x_{n-1})$ ，则  $\lim_{n \rightarrow \infty} x_n = x$  为不动点。

不动点迭代\*的几何意义相当于从  $(x_0, 0)$  作竖直直线交  $f$  于  $(x_0, x_1)$ ，再由  $(x_0, x_1)$  作水平直线交  $y = x$  于  $(x_1, x_1)$ ，再由  $(x_1, x_1)$  作竖直直线交  $f$  于  $(x_1, x_2)$ ，不断重复，直至收敛。

强调，只有是稳定不动点（对于  $k$  的要求在  $(0,1)$  范围内）的时候，  
不动点迭代才会收敛（压缩映射原理）

### 3.4.2 不动点收敛速度

In the condition of  $|g'(x)| \leq k < 1$ , for  $\forall x \in [a, b]$ , we have

$$\begin{aligned} |p_n - p| &= |g(p_{n-1}) - p| = |g'(\xi)||p_{n-1} - p| \\ &\leq k|p_{n-1} - p| \\ &\leq k^2|p_{n-2} - p| \\ &\leq k^n|p_1 - p| \end{aligned}$$

and it's obviously that the  $p_n$  is astringent (收敛的), and this Theorem is called Contraction Mapping (压缩映射原理)

思考：为什么逆问题这么困难——> 因为逆问题的时间复杂度要比正问题高得多  
(举例：求矩阵的逆)

## 3.5 Newton's Method

### 3.5.1 Conception

考虑 Taylor 公式，在根附近展开，我们有

$$0 = f(p_0) + f'(p_0)(p - p_0)$$

由此得到

$$p = p_0 - \frac{f(p_0)}{f'(p_0)}$$

迭代公式转化为

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

记

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

则不难发现 Netwon's Method 其实也是一种特殊的不动点法

### 3.5.2 收敛性判断

对于一般的不动点法来说，要求  $|g'(x)| \leq k < 1$ ，那我们来看 Netwon's Method, 有

$$g'(x) = \dots = \frac{f(x)f''(x)}{(f'(x))^2}$$

与此同时，由于  $f(x_0) = 0$ , 那么我们就有  $g'(x_0) = 0$ , 那么

$$\exists \delta > 0, \forall x \in [x_0 - \delta, x_0 + \delta], \text{ 我们总有 } g'(x) = k < 1$$

这时就会发现在零点附近，Netwon's Method 一定是收敛的不动点而不是发散的，而且收敛速度也会很快

### 3.5.3 Adjoint Method (伴随方法 or BT (即神经网络中的反向传播梯度算法))

Netwon's Method 方法收敛速度很快，但是他也给计算机计算带来了问题——导数的计算也是存在一定的困难，那么这时，引入 BT 算法就会在一定程度上解决这个问题 (BT 算法也是很不容易) 另：Hessian(求解二阶梯度) 方法

## 3.6 Order of Convergence

### Order of Convergence

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

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

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

- If  $\alpha = 1$ , the sequence is linearly convergent.
- If  $\alpha = 2$ , the sequence is quadratically convergent.

图 8: Order of Convergence

### 3.6.1 Error Analysis for Fix-point Methods

### Proof

- Since  $g'$  exists on  $(a, b)$ , applying the MVT, we have

$$p_{n+1} - p = g(p_n) - g(p) = g'(\xi_n)(p_n - p)$$

- Since  $\{p_n\}_{n=0}^{\infty}$  converges to  $p$ ,  $\{\xi_n\}_{n=0}^{\infty}$  also converges to  $p$ .

- Thus,

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

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

图 9: Error Analysis for Fix-point Methods

### Proof (2/2)

- When  $x = p_n$ , we have

$$p_{n+1} = g(p_n) = p + \frac{g''(\xi)}{2}(p_n - p)^2$$

- Thus,

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^2} = \frac{|g''(p)|}{2} < \frac{M}{2}$$

图 10: Netwon's Methods

#### 3.6.2 Error Analysis for Netwon's Methods

## 4 Fixed Points for Functions of Several Variables

### 4.1 Vector Norm(向量范数)

Tips: 此处所指向量不加说明默认为列向量

#### 4.1.1 Defination

范数的一些基本性质 (向量空间中范数的定义以及在线性空间中体现出来的性质)

#### Definition: Vector Norm

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

- ①  $\|\mathbf{x}\| \geq 0$  for all  $\mathbf{x} \in \mathbb{R}^n$
- ②  $\|\mathbf{x}\| = 0$  iff  $\mathbf{x} = 0$
- ③  $\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\|$  for all  $\alpha \in \mathbb{R}$  and  $\mathbf{x} \in \mathbb{R}^n$
- ④  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$  for all  $\|\mathbf{x}\|, \|\mathbf{y}\| \in \mathbb{R}^n$ .

图 11: DefVectorNorm

范数不同的定义

### Definition: $L_1$ , $L_2$ , and $L_\infty$ Norms

The norms for the vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)^t$  are defined by:

- $L_1$  Norm

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$$

- $L_2$  Norm

$$\|\mathbf{x}\|_2 = \left\{ \sum_{i=1}^n x_i^2 \right\}^{1/2}$$

- $L_\infty$  Norm

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

Note that each of these norms reduces to the absolute value in the case  $n = 1$ .

图 12: Vector Norm

## 4.2 A Limit of a Sequence of Vectors in $\mathbb{R}^n$

**定理 4.1.** For each  $\mathbf{x} \in \mathbb{R}^n$ , we have  $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty$

This Theorem is very easy, we will less introduce the proof of it, here is a picture to show the answer:

### Proof

Let  $x_j$  be a coordinate of  $\mathbf{x}$  such that  $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i| = |x_j|$ .  
Then

$$\|\mathbf{x}\|_\infty^2 = |x_j|^2 \leq \sum_{i=1}^n x_i^2 = \|\mathbf{x}\|_2^2$$

and

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2$$

so

$$\|\mathbf{x}\|_2^2 = \sum_{i=1}^n x_i^2 \leq \sum_{i=1}^n x_j^2 = nx_j^2 = n\|\mathbf{x}\|_\infty^2$$

and  $\|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty$ .

图 13: proof

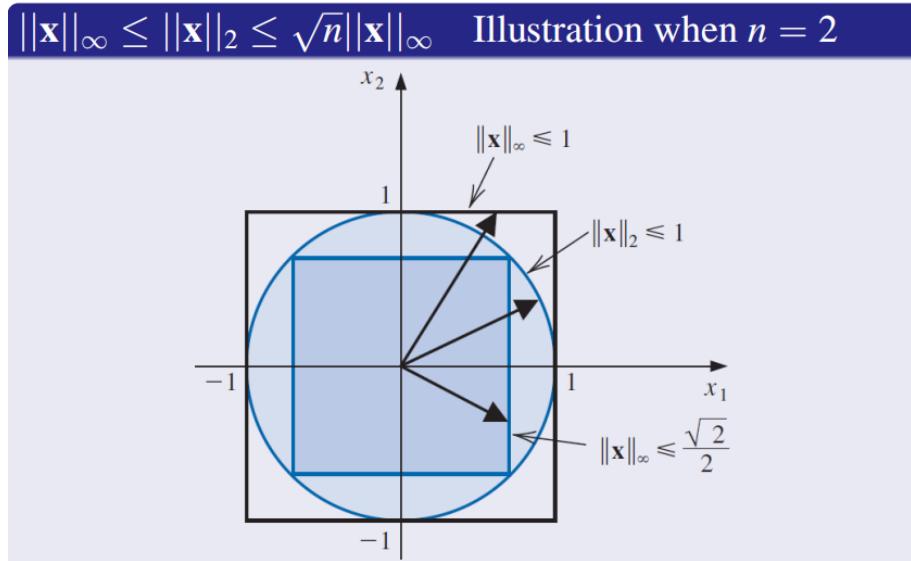


图 14: pictures to proof

**定理 4.2.** For each

### 4.3 Fixed Points for Functions of Several Variables

A system of nonlinear equations has the form:

$$f_1(x_1, x_2, x_3, \dots, x_n) = 0,$$

$$f_2(x_1, x_2, x_3, \dots, x_n) = 0,$$

$$f_3(x_1, x_2, x_3, \dots, x_n) = 0,$$

...

$$f_n(x_1, x_2, x_3, \dots, x_n) = 0,$$

which can be represented by

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}$$

The functions  $f_1, f_2, \dots, f_n$  are called the coordinate functions(坐标函数) of  $\mathbf{F}$ .

但是要注意,  $n$  个方程要求是相互独立的 (岂不是废话 bushi)

### 4.4 Fix-Point Method

A function  $\mathbf{G}$  from  $\mathbf{D} \in \mathbb{R}^n$  into  $\mathbf{R}^n$  has a fixed point at  $\mathbf{p} \in \mathbf{D}$  if  $\mathbf{G}(\mathbf{p}) = \mathbf{p}$ .

### Fixed Points in $\mathbb{R}^n$

A function  $\mathbf{G}$  from  $\mathbf{D} \in \mathbb{R}^n$  into  $\mathbb{R}^n$  has a **fixed point** at  $\mathbf{p} \in D$  if  $\mathbf{G}(\mathbf{p}) = \mathbf{p}$ .

### Theorem (Existence of Fixed Points)

Let  $\mathbf{D} = \{(x_1, x_2, \dots, x_n)^t | a_i \leq x_i \leq b_i, \text{for each } i = 1, 2, \dots, n\}$  for some collection of constants  $a_1, a_2, \dots, a_n$  and  $b_1, b_2, \dots, b_n$ .

Suppose  $\mathbf{G}$  is a continuous function from  $\mathbf{D} \in \mathbb{R}^n$  into  $\mathbb{R}^n$  with the property that  $\mathbf{G}(\mathbf{x}) \in \mathbf{D}$  whenever  $\mathbf{x} \in \mathbf{D}$ . Then  $\mathbf{G}$  has a fixed point in  $\mathbf{D}$ .

图 15: fixpoint

### Theorem (Fixed Point Theorem)

Let  $\mathbf{D} = \{(x_1, x_2, \dots, x_n)^t | a_i \leq x_i \leq b_i, \text{for each } i = 1, 2, \dots, n\}$  for some collection of constants  $a_1, a_2, \dots, a_n$  and  $b_1, b_2, \dots, b_n$ . Suppose  $\mathbf{G}$  is a continuous function from  $\mathbf{D} \in \mathbb{R}^n$  into  $\mathbb{R}^n$  with the property that  $\mathbf{G}(\mathbf{x}) \in \mathbf{D}$  whenever  $\mathbf{x} \in \mathbf{D}$ . Then  $\mathbf{G}$  has a fixed point in  $\mathbf{D}$ .

Moreover, suppose that all the component functions of  $\mathbf{G}$  have continuous partial derivatives and a constant  $K < 1$  exists with

$$\left| \frac{\partial g_i(\mathbf{x})}{\partial x_j} \right| \leq \frac{K}{n}, \quad \text{whenever } \mathbf{x} \in \mathbf{D},$$

for each  $j = 1, 2, \dots, n$  and each component function  $g_i$ . Then the sequence  $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$  defined by an arbitrarily selected  $\mathbf{x}^{(0)}$  in  $\mathbf{D}$  and generated by

$$\mathbf{x}^{(k)} = \mathbf{G}(\mathbf{x}^{(k-1)}), \quad \text{for each } k \geq 1$$

converges to the unique fixed point  $\mathbf{p} \in \mathbf{D}$  and

$$\|\mathbf{x}^{(k)} - \mathbf{p}\|_{\infty} \leq \frac{K^k}{1-K} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|_{\infty}$$

图 16: fix-point-define

**原理推导** :

$\mathbf{D}$  是一个  $n$  元列向量, 那对于从  $\mathbf{D}$  到  $\mathbf{D}$  映射的函数  $\mathbf{G}$  而言, 如果存在不动 “点”  $\mathbf{p}$ , 使得

$$\mathbf{p} = \mathbf{G}(\mathbf{p}) \text{ 亦即 } p_i = g_i(\mathbf{p}), \forall i = 1, 2, 3, \dots, n$$

那么我们接下来讨论其收敛特性: 类比一元函数的不动点收敛的充分条件, 仍采用类似压缩映射 +MVT (拉格朗日中值定理) 的思路, 使用多元的 MVT 公式:

$$f(\mathbf{x}) - f(\mathbf{y}) = \nabla f(\zeta) \cdot (\mathbf{x} - \mathbf{y})$$

将上式变形, 并使用  $g_i$  来代替  $f$ , 并使用柯西不等式可以得到:

$$|g_i(\mathbf{x}) - g_i(\mathbf{y})| = \|\nabla f(\zeta) \cdot (\mathbf{x} - \mathbf{y})\|_2 \leq \|\nabla f(\zeta)\|_2 \cdot \|\mathbf{x} - \mathbf{y}\|_2$$

此时我们将  $\mathbf{y} =$  不动点  $\mathbf{p}$  带入得到:

$$|g_i(\mathbf{x}) - g_i(\mathbf{p})| \leq \|\nabla f(\zeta)\|_2 \cdot \|\mathbf{x} - \mathbf{p}\|_2$$

由于  $g_i(\mathbf{p}) = \mathbf{p}$  进而有

$$|g_i(\mathbf{x}) - \mathbf{p}|_2 \leq \|\nabla f(\zeta)\|_2 \cdot \|\mathbf{x} - \mathbf{p}\|_2$$

即

$$\frac{\|\mathbf{G}(\mathbf{x}) - \mathbf{p}\|_2}{\|\mathbf{x} - \mathbf{p}\|_2} \leq \|\nabla f(\zeta)\|_2 \triangleq K$$

我们让  $K$  小于 1, 那么就不难推出这样的迭代是收敛的 (公式写的有点问题, 意会一下, 大致思路是这样)

## 4.5 Newton's Method

# 5 The solution of Linear Systems

(线性方程组的寻根问题)

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = 0, \quad (1)$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = 0, \quad (2)$$

$$\vdots \quad (3)$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = 0. \quad (4)$$

which can be represented as

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

## A More Precise Description

Gaussian elimination procedure is described more precisely, although more intricately, by forming a sequence of augmented matrices  $\tilde{A}^{(1)}, \tilde{A}^{(2)}, \dots, \tilde{A}^{(n)}$ , where  $\tilde{A}^{(1)}$  is the matrix  $\tilde{A}$  given earlier and  $\tilde{A}^{(k)}$ , for each  $k = 2, 3, \dots, n$ , has entries  $a_{ij}^{(k)}$ , where:

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)}, & \text{when } i = 1, 2, \dots, k-1 \text{ and } j = 1, 2, \dots, n+1 \\ 0, & \text{when } i = k, k+1, \dots, n \text{ and } j = 1, 2, \dots, k-1 \\ a_{ij}^{(k-1)} - \frac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} a_{k-1,j}^{(k-1)}, & \text{when } i = k, k+1, \dots, n \text{ and } j = k, k+1, \dots, n+1 \end{cases}$$

图 17: details-of-gauss

## 5.1 Gaussian elimination with backward substitution

### 5.1.1 算法内容

解方程组  $\mathbf{A}\vec{x} = \vec{b}$ , 记  $\mathbf{A}^{(1)} = \mathbf{A} = a_{ij}^{(1)}$ ,  $\vec{b}^{(1)} = \vec{b} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$ ,  $\vec{x}^{(1)} = \vec{x} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_n^{(1)} \end{pmatrix}$ , 则

增广矩阵  $\tilde{\mathbf{A}}$  为:

$$\tilde{\mathbf{A}} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}$$

通过高斯消元我们可以得到新的增广矩阵  $\tilde{\mathbf{A}}^{(k)}$ :

$$\tilde{\mathbf{A}}^{(k)} = \begin{bmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \cdots & a_{1n}^{(k)} & b_1^{(k)} \\ 0 & a_{22}^{(k)} & \cdots & a_{2n}^{(k)} & b_2^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{bmatrix}$$

注意, 对于得到的新的增广矩阵中, 我们要求  $a_{ii}^k \neq 0$ (即要保证方程有唯一解) 但是在高斯消元的过程, 会发现, 有可能某一项是 0

### Remarks

The procedure will fail if one of the elements  $a_{kk}^{(k)}$  is zero because

- either the following step can not be formed

$$\left( E_i - \frac{a_{i,k}}{a_{kk}^{(k)}} E_k \right) \rightarrow (E_i)$$

- or the backward substitution cannot be accomplished in the case  $a_{nn}^{(n)}$ .

The system may still have a solution, but the technique for finding it must be altered.

图 18: problems

#### 5.1.2 主元 (Pivot) 的定义:

- **主元**: 在一个系数矩阵 ( $A$ ) 中, 用于消去其他行中的相应元素的元素 ( $a_k$ ) 被称为第 ( $k$ ) 个主元。具体来说, 当我们进行高斯消去法 (Gaussian elimination) 时, 主元是那个用来将 ( $k+1$ )、( $k+2$ ) 到 ( $n$ ) 行的元素消去 (即让它们变成零) 的系数。
- **主元行**: 含有主元 ( $a_{-k}$ ) 的第 ( $k$ ) 行称为第 ( $k$ ) 个主元行。

#### 5.1.3 主元策略 (Pivoting Strategy):

主元策略是指在进行高斯消去法时如何选择主元, 以提高算法的稳定性和减少数值误差。常见的主元策略包括以下两种:

1. **避免主元为零**: 如果在消去过程中选到的主元 ( $a_{-k}$ ) 为零 (或接近于零), 消去过程将无法进行 (或者可能会产生非常大的误差)。因此, 在进行消去时, 可以通过交换行 (或者列) 来选取一个非零的元素作为主元。这称为**部分主元法** (Partial Pivoting) 或**完全主元法** (Complete Pivoting)。
2. **减少误差**: 即使主元不为零, 为了减少计算误差, 我们通常希望选择一个数值较大的元素作为主元。这也是一种常见的主元策略, 目的是减少由于计算中的舍入误差而引入的不准确性。

另:

### Scaled Pivoting Strategies

- The effect of scaling is to ensure that the largest element in each row has a relative magnitude of 1 before the comparison for row interchange is performed.

#### 5.1.4 总结

总的来说，这段话描述了在高斯消去法中如何选择主元，以及选择主元的策略。通过合理选择主元，可以避免消去失败或数值不稳定的情况，提高计算的准确性。

### Gaussian Elimination Algorithm

INPUT number of unknowns and equations  $n$ ; augmented matrix  $A = [a_{ij}]$ , where  $1 \leq i \leq n$  and  $1 \leq j \leq n + 1$ .

OUTPUT solution  $x_1, x_2, \dots, x_n$  or message that the linear system has no unique solution.

Step 1 For  $i = 1, \dots, n - 1$  do Steps 2–4. (Elimination process.)

Step 2 Let  $p$  be the smallest integer with  $i \leq p \leq n$  and  $a_{pi} \neq 0$ .  
If no integer  $p$  can be found  
then OUTPUT ('no unique solution exists');  
STOP.

$O(n)$

Step 3 If  $p \neq i$  then perform  $(E_p) \leftrightarrow (E_i)$ .

$O(n)$

Step 4 For  $j = i + 1, \dots, n$  do Steps 5 and 6.

Step 5 Set  $m_{ji} = a_{ji}/a_{ii}$ .

$O(n)$

Step 6 Perform  $(E_j - m_{ji}E_i) \rightarrow (E_j)$ ;

$O(n)$

Step 7 If  $a_{nn} = 0$  then OUTPUT ('no unique solution exists');  
STOP.

$O(n^3)$

Step 8 Set  $x_n = a_{n,n+1}/a_{nn}$ . (Start backward substitution.)

Step 9 For  $i = n - 1, \dots, 1$  set  $x_i = \left[ a_{i,n+1} - \sum_{j=i+1}^n a_{ij}x_j \right] / a_{ii}$ .

$O(n)$

Step 10 OUTPUT  $(x_1, \dots, x_n)$ ; (Procedure completed successfully.)

$O(n)$

$O(n^2)$

32/1

图 19: 算法流程及时间复杂度

时间复杂度主要考虑两方面——乘法计算的次数和 loop 循环的次数，亦即说明 Gauss 消元法的时间复杂度是  $O(n^3)$

## 5.2 LU Factorization(LU 分解)

高斯消元法实际上是一种特殊的 LU 分解

## Permutation Matrices

### Permutation Matrices

An  $n \times n$  permutation matrix  $P = [p_{ij}]$  is a matrix obtained by rearranging the rows of  $I_n$ , the identity matrix.

### Example

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$PA = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

图 20: 置换矩阵

### 5.2.1 主对角占优

#### Definition (Diagonally Dominant Matrices)

The  $n \times n$  matrix  $A$  is said to be **diagonally dominant** when

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^n |a_{ij}|$$

holds for each  $i = 1, 2, \dots, n$ .

#### Definition (Strictly Diagonally Dominant)

A diagonally dominant matrix is said to be **strictly diagonally dominant** when

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|$$

holds for each  $i = 1, 2, \dots, n$ .

图 21: 主对角占优

iff——当且仅当 0

## 6 Iterative solution of Linear Systems

**定义 6.1.** 矩阵范数：对于  $n \times n$  复矩阵空间  $\mathbb{C}^{n \times n}$ , 我们也希望定义一个长度衡量矩阵的大小，定义距离比较两个矩阵之间的接近程度，由此我们引进了矩阵范数  
设  $\mathbb{C}^{n \times n} \rightarrow \mathbb{R}$  的函数  $\|\bullet\|$  满足

1. 正定性： $\forall A \in \mathbb{C}^{n \times n}, \|A\| \geq 0$  其中  $\|A\| = 0$  当且仅当  $A = 0$
2. 齐次性： $f(\alpha A) = |\alpha| \cdot f(A), A \in \mathbb{R}^{n \times n}, \alpha \in \mathbb{R}$
3. 相容性：

$$\|AB\| \leq \|A\| \|B\|, \forall A, B \in \mathbb{R}^{n \times n}, \|Ax\| \leq \|A\| \|x\|, \forall A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^n$$

**定义2：**设  $\mathbb{C}^{n \times n} \rightarrow \mathbb{R}$  的函数  $\|\bullet\|$  满足

- (1) 正定性： $\forall A \in \mathbb{C}^{n \times n}, \|A\| \geq 0$ , 其中  $\|A\| = 0$  当且仅当  $A = 0$
- (2) 齐次性<sup>\*</sup>： $\forall A \in \mathbb{C}^{n \times n}, \forall \alpha \in \mathbb{C}$ , 有  $\|\alpha A\| = |\alpha| \cdot \|A\|$
- (3) 三角不等式： $\forall A, B \in \mathbb{C}^{n \times n}$ , 有  $\|A + B\| \leq \|A\| + \|B\|$
- (4) 乘法相容性： $\forall A, B \in \mathbb{C}^{n \times n}$ , 有  $\|AB\| \leq \|A\| \|B\|$

则称函数  $\|\bullet\|$  为  $\mathbb{C}^{n \times n}$  上的矩阵范数

图 22: 矩阵范数定义

(当然了, 由于矩阵并没有长度的概念, 因此将矩阵范数看作是对矩阵长度的度量并不可行)

### 6.1 诱导范数

#### 6.1.1 定义

**定义 6.2.** 矩阵的诱导范数定义如下：

$$\|A\|_{(m,n)} = \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|Ax\|_{(m)}}{\|x\|_{(n)}} = \sup_{\substack{x \in \mathbb{C}^n \\ \|x\|_n=1}} \|Ax\|_{(m)}.$$

### 6.1.2 相关性质

**命题 6.3.** If  $\mathbf{A} = (a_{ij})$  is a  $n \times n$  matrix, then

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

证明. 设  $x = (x_1, x_2, \dots, x_n)^T \neq 0$ , 我们有

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

记

$$\mu = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

于是我们有

$$\|Ax\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| |x_j| \leq \|x\|_\infty \cdot \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| = \|x\|_\infty \mu$$

则

$$\|\mathbf{A}\|_\infty = \max_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} \leq \mu$$

设  $\mu = \sum_{j=1}^n |a_{i_0 j}|$ , 令

$$x_0 = (sgn(a_{i_0 1}), sgn(a_{i_0 2}), \dots, sgn(a_{i_0 n}))^T$$

则  $Ax_0$  的第  $i_0$  个分量为  $\sum_{j=1}^n a_{i_0 j} sgn(a_{i_0 j}) = \sum_{j=1}^n |a_{i_0 j}| = \mu$ , 且

$$\|x_0\|_\infty = 1$$

从而

$$\frac{\|Ax_0\|_\infty}{\|x_0\|_\infty} = \mu$$

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

□

## 6.2 特征值和特征向量

**定义 6.4.** 谱半径: 定义为矩阵  $\mathbf{A}$  的特征值的绝对值的最大值

$$\rho(\mathbf{A}) = \max\{|\lambda|\}$$

**推论 6.5.** 相应的, 谱范数有如下两个性质

$$1. (\|\mathbf{A}\|_2)^2 = \rho(\mathbf{A}^T \mathbf{A})$$

$$2. \rho(\mathbf{A}) \leq \|\mathbf{A}\|$$

证明. 对于第一个式子,

□

### 6.3 Convergent Matrix(收敛矩阵)

**定义 6.6.** We call an  $n \times n$  matrix  $A$  **convergent** if

$$\lim_{k \rightarrow \infty} (A^{(k)})_{ij} = A_0$$

**定义2:** 设  $\mathbb{C}^{n \times n}$  上的矩阵序列  $\{A_k\}$  以及矩阵  $A_0$  满足: 对给定的矩阵范数  $\|\bullet\|_v$ ,  $\lim_{k \rightarrow \infty} \|A_k - A_0\|_v = 0$ , 则称序列  $\{A_k\}$  依范数  $\|\bullet\|_v$  收敛于  $A_0$ , 并记为  $\lim_{k \rightarrow \infty} A_k = A_0$

对于依范数收敛, 我们有如下定理

**定理3:** 对于  $\mathbb{C}^{n \times n}$  上的矩阵序列  $\{A_k\}$  以及矩阵  $A_0$ , 其中  $A_k = (a_{ij})^{n \times n}$ ,  $A_0 = (a_{ij})^{n \times n}$ , 则下面两个命题等价

- (1)  $\{A_k\}$  依矩阵范数  $\|\bullet\|_v$  收敛于  $A_0$
- (2) 对  $\forall 1 \leq i, j \leq n$ , 有  $\lim_{k \rightarrow \infty} a_{ij}^{(k)} = a_{ij}$  (即按坐标收敛或者按元素收敛)

以后我们将矩阵序列依范数收敛简称为矩阵序列收敛

图 23: 依范数收敛

#### 6.3.1 The Jacobi Iterative Method

#### 6.3.2 The Gauss-Seidel Method

#### 6.3.3 收敛性分析

计算谱半径比较复杂, 直接给出收敛性判定的一般结论:

1. 如果  $A$  严格对角占优:

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|, \quad i = 1, 2, \dots, n$$

则 Jacobi 迭代和 Gauss-Seidel 迭代收敛;

2. 如果  $A$  对称正定, Gauss-Seidel 迭代收敛, 以及:

3. 如果  $A$  对称正定且  $(2D - A)$  对称正定, Jacobi 迭代也收敛。

直观解释: 严格对角占优, 对角阵就记录了  $A$  大部分信息;  $A$  对称正定, Gauss-Seidel 利用下三角和对角来近似  $A$ , 实际上已经包含了  $A$  的所有信息; 如果  $A$  对称正定且  $(2D - A)$  对称正定也是为了保证对角线元素比重更大。

图 24: 收敛性分析

**Theorem (Stein-Rosenberg)**

If  $a_{ij} \leq 0$ , for each  $i \neq j$  and  $a_{ii} > 0$ , for each  $i = 1, 2, \dots, n$ , then one and only one of the following statements holds:

- ①  $0 \leq \rho(T_g) < \rho(T_j) < 1$ ;
- ②  $1 < \rho(T_j) < \rho(T_g)$ ;
- ③  $\rho(T_j) = \rho(T_g) = 0$ ;
- ④  $\rho(T_j) = \rho(T_g) = 1$ ;

图 25: Stein-Rosenberg

**6.3.4 Condition Number(条件数)****Residual Vector**

Suppose  $\tilde{\mathbf{x}} \in \mathbb{R}^n$  is an approximation to the solution of the linear system defined by  $A\mathbf{x} = \mathbf{b}$ . The **residual vector** for  $\tilde{\mathbf{x}}$  with respect to this system is  $\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{x}}$ .

**Condition Number**

The **condition number** of a nonsingular matrix  $A$  relative to a norm  $\|\cdot\|$  is

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

图 26: Condition Number

抓住最主要的特征——> 找最大特征值

**6.4 特征值的估计——圆盘定理 (Gerschgorin(戈氏) 圆盘第一定理)****6.5 The Power Method****Overview**

CFL 与最大特征值的关系 (思考?)

考慮到一个  $n \times n$  的矩阵  $A$  的特征值最大值没有重根, 特征值按序号、模值大小排序, 亦即

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n| \geq 0$$

## 7 插值与多项式近似

对于有限元分析，三角形网格是属于一阶精度（三角形边界以切线贴合曲线边界），对于矩形网格来说，是属于 0 阶精度（矩形边界无法很好贴合曲线边界），并且对于三角形边界获得的雅可比矩阵是常数，而对于四边形来说雅可比矩阵是随位置变化，计算更加困难。

### 7.1 泰勒插值

### 7.2 拉格朗日插值 (Lagrange Polynomials)

拉格朗日插值就是构造一个次数至多为  $n$  次的多项式使它通过  $n+1$  个给定的点，这个多项式就是拉格朗日多项式。

构造  $P(x) = a_0 + a_1x$ ，使得  $P(x_0) = y_0$ ,  $P(x_1) = y_1$ 。

则  $P(x) = y_0 + \frac{y_1-y_0}{x_1-x_0}(x-x_0) = \frac{x-x_1}{x_0-x_1}y_0 + \frac{x-x_0}{x_1-x_0}y_1$ 。

其中  $\frac{x-x_1}{x_0-x_1}$  和  $\frac{x-x_0}{x_1-x_0}$  分别记作  $L_{1,0}(x)$  和  $L_{1,1}(x)$ （第一个下标即为  $n$  的值，第二个下标为样本点的序号），这称为拉格朗日基函数 (Lagrange Basis)。

那我们不难发现，对于  $L_{n,i}(x_j)$  是满足  $\delta_{ij}$  函数的性质的：

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

推广到  $n$  次插值，构造  $P(x) = a_0 + a_1x + \cdots + a_nx^n$ ，使得  $P(x_i) = y_i$ ,  $i = 0, 1, \dots, n$ 。就是要找到  $L_{n,i}(x)$  使得  $L_{n,i}(x_j) = \delta_{ij}$

分析可知，这里的  $L_{n,i}(x)$  有  $n$  个根，分别为  $x_0, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ 。所以可以构造出

$$L_{n,i}(x) = C(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)$$

又因为  $L_{n,i}(x_i) = 1$ ，所以

$$L_{n,i}(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

即

$$L_{n,i}(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

于是我们根据拉格朗日基函数构造出了  $n$  次拉格朗日插值多项式

$$P_n(x) = \sum_{i=0}^n L_{n,i}(x)y_i$$

### 拉格朗日插值具有唯一性

对  $n$  个不同的点,  $n$  次拉格朗日插值多项式是唯一的

证明. 如果不唯一, 假设存在另一个多项式  $Q_n(x)$ , 使得  $Q_n(x_i) = y_i$ ,  $i = 0, 1, \dots, n$ , 且  $Q_n(x) \neq P_n(x)$ 。

则  $R_n(x) = P_n(x) - Q_n(x)$  是一个次数不超过  $n$  的多项式, 且  $R_n(x_i) = 0$ ,  $i = 0, 1, \dots, n$ 。

由于  $R_n(x)$  的次数不超过  $n$ ,  $n$  次多项式不可能有  $n+1$  个解, 所以  $R_n(x) = 0$ , 即  $P_n(x) = Q_n(x)$ , 与假设矛盾。  $\square$

### 拉格朗日插值的余项

假定  $a \leq x_0 < x_1 < \dots < x_n \leq b$ ,  $f \in C[a, b]$ ,  $P_n(x)$  是  $f(x)$  在  $x_0, x_1, \dots, x_n$  上的拉格朗日插值多项式, 则对任意  $x \in [a, b]$ , 存在  $\xi(x) \in (a, b)$ , 使得

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

证明. 我们构造函数

$$h(t) = f(t) - P_n(t) + (f(x) - P_n(x)) \left( \prod_{i=0}^n \frac{t - x_i}{x - x_i} \right)$$

不难发现,  $h(t)$  有  $n+1$  个零点, 反复使用罗尔中值定义, 我们可以得到

$$h^{(n+1)}(\xi(x)) = 0$$

亦即

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

由于  $\left( \prod_{i=0}^n \frac{t - x_i}{x - x_i} \right)^{(n)}|_{t=\xi(x)} = \frac{(n+1)!}{\prod_{i=0}^n (x - x_i)}$ , 同时有  $P^{(n+1)}(t) = 0$

接着有

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

显然, 将式子整理, 不难发现这个式子就是所求

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

$\square$

### 7.3 Neville 迭代插值法 (Neville's Method)

**命题 7.1.** 设  $f$  在  $x_0, x_1, \dots, x_n$  上有定义,  $m_1, m_2, \dots, m_k$  是  $k$  个不同的整数,  $0 \leq m_i \leq n$ ,  $i = 1, 2, \dots, k$ 。记在这  $k$  个点上与  $f(x)$  对应的拉格朗日多项式为  $P_{m_1, m_2, \dots, m_k}(x)$ 。

**定理 7.2.** 设  $f$  在  $x_0, x_1, \dots, x_n$  上有定义, 让  $x_i$  和  $x_j$  是这个集合中的两个不同的数。则

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

描述了对  $f$  在  $x_0, x_1, \dots, x_k$  这  $k + 1$  个点上的  $k$  次插值多项式。

证明. 对于任意  $0 \leq r \leq k$ ,  $r \neq i$  和  $r \neq j$ , 分子上的两个插值多项式在  $x_r$  处都等于  $f(x_r)$ , 所以  $P(x_r) = f(x_r)$ 。

分子上的第一个多项式在  $x_i$  处等于  $f(x_i)$ , 而第二个多项式在  $x_i$  处为 0, 所以  $P(x_i) = f(x_i)$ 。同理  $P(x_j) = f(x_j)$ 。

所以  $P(x)$  在  $x_0, x_1, \dots, x_k$  上与  $f(x)$  相同, 因为  $P(x)$  是  $k$  次多项式, 所以  $P(x) = P_{0,1,\dots,k}(x)$

□

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

图 27: Method

#### Comments

- This result implies that the interpolating polynomials can be generated recursively.
- For example we have

$$\begin{aligned} P_{0,1} &= \frac{1}{x_1 - x_0} [(x - x_0)P_1 - (x - x_1)P_0] \\ P_{1,2} &= \frac{1}{x_2 - x_1} [(x - x_1)P_2 - (x - x_2)P_1] \\ P_{0,1,2} &= \frac{1}{x_2 - x_0} [(x - x_0)P_{1,2} - (x - x_2)P_{0,1}] \end{aligned}$$

and so on.

图 28: example

## 7.4 Newton's Divided Difference Interpolating Polynomial (牛顿插值)

上节讲到 Lagrange 插值时候，其实它有一个缺点，就是如果我们增删一个节点的话，我们需要更改所有的插值基函数，这样就太不方便了，所以我们提出牛顿插值。

$f(x)$  在  $x_0, x_1, \dots, x_n$  处取值为  $f(x_0), f(x_1), \dots, f(x_n)$ ,

称  $f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$  为  $f(x)$  在  $x_i, x_j$  处的一阶差商,

称  $f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}$  为  $f(x)$  在  $x_i, x_j, x_k$  处的二阶差商,

称  $f[x_0, x_1, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$  为  $f(x)$  在  $x_0, x_1, \dots, x_k$  处的  $k$  阶差商.

## 7.5 样条插值方法 (spline)

样条多项式插值是一种将插值问题分解为多个低次多项式区间段来进行逼近的方法。最常见的是三次样条插值 (Cubic Spline Interpolant)。

在每个区间  $[x_i, x_{i+1}]$  上，使用一个三次多项式  $S_i(x)$ :

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

这些多项式段需要满足以下条件:

1. 插值点处的函数值连续性条件:  $S_i(x_i) = f(x_i)$
2. 一阶导数的连续性条件:  $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$
3. 二阶导数的连续性条件:  $S''_i(x_{i+1}) = S''_{i+1}(x_{i+1})$

这时候我们发现方程是欠定方程，即缺少条件，这时候我们补充边界条件:

4. 边界条件
  - (a) 自由边界条件  $S''_0(x_0) = 0$  和  $S''_n(x_n) = 0$
  - (b) 紧束缚边界条件  $S'_0(x_0) = f'(x_0)$  和  $S'_n(x_n) = f'(x_n)$

我们发现，总共有  $2n + 2n - 2 + 2 = 4n$  个方程，而我们刚刚好有  $n \times 4 = 4n$  个未知数

## 参数计算

记  $h_j = x_j - x_{j-1}$ , 由条件 3, 可得

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3, \quad j = 0, 1, \dots, n-1$$

又由条件 4, 因为  $S'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$ , 所以

$$b_{j+1} = S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}) = b_j + 2c_j h_j + 3d_j h_j^2, \quad j = 0, 1, \dots, n-1$$

又由条件 5, 因为  $S''(x) = 2c_j + 6d_j(x - x_j)$ , 所以

$$c_{j+1} = \frac{S''_{j+1}(x_{j+1})}{2} = c_j + 3d_j h_j, \quad j = 0, 1, \dots, n-1$$

所以

$$\begin{cases} a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \\ b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2 \\ c_{j+1} = c_j + 3d_j h_j \end{cases}, \quad j = 0, 1, \dots, n-1$$

把最后一个式子代入前两个式子, 消去  $d_j$ , 得到

$$\begin{cases} a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \\ b_{j+1} = b_j + h_j(c_j + c_{j+1}) \end{cases}, \quad j = 0, 1, \dots, n-1$$

为了减少未知数, 我们有

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \Rightarrow \begin{cases} b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \\ b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) \end{cases}$$

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \Rightarrow b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j)$$

所以

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

$$\Rightarrow \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) + h_{j-1}(c_{j-1} + c_j)$$

$$\Rightarrow h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_jc_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \quad (j = 1, 2, \dots, n-1)$$

因为  $a_j, h_j$  已知, 所以上式未知量仅为  $c_j$ , 而且求出  $c_j$  后,  $b_j$  也就求出了。

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

所以我们有  $n-1$  个方程,  $n+1$  个未知数, 所以我们需要额外的两个方程。

### 7.5.1 自由边界条件

书上给的是  $S''(a) = S''(b) = 0$ , 实际上, 我们在做题中扩展到了  $S''(a) = s_0$ ,  $S''(b) = s_n$ , 此时

$$c_0 = \frac{S''(a)}{2} = \frac{s_0}{2}, \quad c_n = \frac{S''(b)}{2} = \frac{s_n}{2}$$

所以, 我们可以将上面的方程组写成  $\mathbf{Ax} = \mathbf{b}$  的形式, 其中  $\mathbf{A}$  为  $(n+1) \times (n+1)$  的矩阵

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \cdots & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1 + h_2) & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{b}$  和  $\mathbf{x}$  为  $(n+1) \times 1$  的向量

$$\mathbf{b} = \begin{bmatrix} \frac{s_0}{2} \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ \frac{s_n}{2} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$$

因为矩阵  $\mathbf{A}$  是严格对角占优的, 所以该方程组有唯一解。

### 7.5.2 固定边界

固定边界要求  $S'(a) = f'(a)$ ,  $S'(b) = f'(b)$ 。

因为  $f'(a) = S'(a) = S'(x_0) = b_0$ ,  $f'(b) = S'(b) = S'(x_n) = b_n$ , 所以

$$\begin{aligned} & \begin{cases} f'(a) = b_0 = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1) \\ f'(b) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n) = \frac{1}{h_{n-1}}(a_n - a_{n-1}) + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n) \end{cases} \\ \Rightarrow & \begin{cases} 2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{cases} \end{aligned}$$

所以, 我们可以将上面的方程组写成  $\mathbf{Ax} = \mathbf{b}$  的形式, 其中  $\mathbf{A}$  为  $(n+1) \times (n+1)$  的矩

阵

$$\mathbf{A} = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & 0 & 0 & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \cdots & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1 + h_2) & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

$\mathbf{b}$  和  $\mathbf{x}$  为  $(n+1) \times 1$  的向量

$$\mathbf{b} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$$

因为矩阵  $\mathbf{A}$  是严格对角占优的，所以该方程组有唯一解。

### 7.5.3 应用举例

## 8 逼近论 (Approximation Theory)

逼近和插值的区别在于，插值是要求通过所有的数据点，而逼近没有这个限制，而是要求逼近的函数和原函数的误差尽可能小——尽可能接近每个点。

### 8.1 Discrete Least Squares Approximation

#### 8.1.1 误差表达

设  $p(x)$  是逼近函数， $y_i$  是给定的  $n$  个数据点，那么逼近误差的三种表达方式如下：

#### Minimax problem

$$E_\infty(p) = \max\{|y_i - f(x)|\}$$

这用初等技术是解决不了的。

## Absolute deviation

$$E_1(p) = \sum_{i=1}^n |y_i - f(x)|$$

困难在于绝对值函数在零点不可微，可能无法求解多元函数的最小值。

## Least squares

$$E_2(p) = \sum_{i=1}^n (y_i - f(x))^2$$

此即为**最小二乘**的误差表达，也是最常用的逼近方法。

我们的目标是找到一个  $p(x)$ ，使得  $E_2(p)$  最小。

### 离散最小二乘逼近

定义： $P_n(x)$  是  $m$  个数据点的**离散最小二乘逼近**，如果  $P_n(x)$  是  $n$  次多项式，且满足

$$p = \arg \min_{p \in \mathbb{P}_n} \sum_{i=1}^m (y_i - p(x_i))^2$$

其中  $\mathbb{P}_n$  是  $n$  次多项式的集合， $n$  应远远小于  $m$ ，如果  $n = m - 1$ ，其即为 Lagrange 插值。

### 离散最小二乘逼近的解

设  $P_n(x) = a_0 + a_1x + \cdots + a_nx^n = \sum_{i=0}^n a_i x^i$ 。

$$E_2 = \sum_{i=1}^m (y_i - P_n(x_i))^2$$

为了使  $E_2$  最小，则其必要条件是

$$\frac{\partial E_2}{\partial a_k} = 0, \quad k = 0, 1, \dots, n$$

即

$$\begin{aligned}
\frac{\partial E_2}{\partial a_k} &= 2 \sum_{i=1}^m (P_n(x_i) - y_i) \frac{\partial P_n(x_i)}{\partial a_k} \\
&= 2 \sum_{i=1}^m \left( \sum_{j=0}^n a_j x_i^j - y_i \right) x_i^k \\
&= 2 \left( \sum_{j=0}^n \left( a_j \sum_{i=1}^m x_i^{j+k} \right) - \sum_{i=1}^m y_i x_i^k \right) = 0
\end{aligned}$$

即

$$\sum_{j=0}^n \left( a_j \sum_{i=1}^m x_i^{j+k} \right) = \sum_{i=1}^m y_i x_i^k, \quad k = 0, 1, \dots, n$$

也就是

$$\begin{bmatrix} \sum_{i=1}^m x_i^0 & \sum_{i=1}^m x_i^1 & \cdots & \sum_{i=1}^m x_i^n \\ \sum_{i=1}^m x_i^1 & \sum_{i=1}^m x_i^2 & \cdots & \sum_{i=1}^m x_i^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^m x_i^n & \sum_{i=1}^m x_i^{n+1} & \cdots & \sum_{i=1}^m x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m y_i x_i^0 \\ \sum_{i=1}^m y_i x_i^1 \\ \vdots \\ \sum_{i=1}^m y_i x_i^n \end{bmatrix}$$

## 8.2 对于直线拟合 7 更加紧凑的形式

### Linear Least Squares Problem

$$\{a_0^*, a_1^*\} = \underset{a_0, a_1}{\operatorname{argmin}} \sum_{i=1}^m [y_i - (a_1 x_i + a_0)]^2$$

$$\begin{aligned}
\mathbf{a}^* &= \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 \\
0 &= -\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{a})
\end{aligned}$$

Normal equation:

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \mathbf{a} &= \mathbf{X}^T \mathbf{y} \\
\Rightarrow \mathbf{a} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}
\end{aligned}$$

## 8.3 Orthogonal Polynomials and Least Squares Approximation (正交多项式与最小二乘逼近)

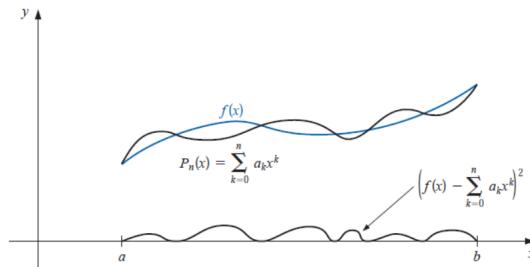
(实在是讲的太快了跟不上 www, 后面贴了不少图)

刚刚是离散化的最小二乘逼近，现在是连续的最小二乘逼近。

## Approximation of Functions

Suppose  $f \in C[a, b]$  and that a polynomial  $P_n(x)$  of degree at most  $n$  is required that will minimize the error

$$E = \int_a^b [f(x) - P_n(x)]^2 dx = \int_a^b \left( f(x) - \sum_{k=0}^n a_k x^k \right)^2 dx$$



## Approximation of Functions

Since

$$E = \int_a^b [f(x)]^2 dx - 2 \sum_{k=0}^n a_k \int_a^b x^k f(x) dx + \int_a^b \left( \sum_{k=0}^n a_k x^k \right)^2 dx,$$

we have

$$\frac{\partial E}{\partial a_j} = -2 \int_a^b x^j f(x) dx + 2 \sum_{k=0}^n a_k \int_a^b x^{j+k} dx.$$

Hence, to find  $P_n(x)$ , the  $(n+1)$  linear **normal equations**

$$\sum_{k=0}^n a_k \int_a^b x^{j+k} dx = \int_a^b x^j f(x) dx, \text{ for each } j = 0, 1, \dots, n.$$

must be solved for the  $(n+1)$  unknowns  $a_j$ . The normal equations always have a **unique** solution provided that  $f \in C[a, b]$ .

## Disadvantages

- ① The coefficients  $a_0, a_1, \dots, a_n$  in the linear system are of the form

$$\int_a^b x^{j+k} dx = \frac{b^{j+k+1} - a^{j+k+1}}{j+k+1},$$

a linear system that does not have an easily computed numerical solution.

- ② The calculations that were performed in obtaining the best  $n$ -degree polynomial,  $P_n(x)$ , do not lessen the amount of work required to obtain  $P_{n+1}(x)$ .

### 8.3.1 Linearly Independent Functions

#### Definition: Linearly Independent

The set of functions  $\{\phi_0, \dots, \phi_n\}$  is said to be **linearly independent** on  $[a, b]$  if, whenever

$$c_0\phi_0(x) + c_1\phi_1(x) + \dots + c_n\phi_n(x) = 0, \quad \text{for all } x \in [a, b],$$

we have  $c_0 = c_1 = \dots = c_n = 0$ . Otherwise the set of functions is said to be **linearly dependent**.

#### Theorem

Suppose that  $\{\phi_0(x), \phi_1(x), \dots, \phi_n(x)\}$  is a collection of linearly independent polynomials in  $\prod_n$ . Then any polynomial in  $\prod_n$  can be written uniquely as a linear combination of  $\phi_0(x), \phi_1(x), \dots, \phi_n(x)$ .

#### Definition: Weight function

An integrable function  $w$  is called a **weight function** on the interval  $I$  if  $w(x) \geq 0$ , for all  $x$  in  $I$ , but  $w(x) \neq 0$  on any subinterval of  $I$ .

#### Example

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

#### Definition: Orthogonal Set of Functions

$\{\phi_0(x), \phi_1(x), \dots, \phi_n(x)\}$  is said to be an **orthogonal set of functions** for the interval  $[a, b]$  with respect to the weight function  $w$  if

$$\int_a^b w(x)\phi_k(x)\phi_j(x)dx = \begin{cases} 0, & \text{when } j \neq k, \\ \alpha_j > 0, & \text{when } j = k. \end{cases}$$

If, in addition,  $\alpha_j = 1$  for each  $j = 0, 1, \dots, n$ , the set is said to be **orthonormal**.

## 9 Numerical Differentiation(数值微分)

### 9.1 一阶导数

#### 9.1.1 两点法

最简单的方法：用两个点，取  $h > 0$

Forward (前向差分) :  $f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$

Backward (后向差分) :  $f'(x) = \frac{f(x)-f(x-h)}{h} + O(h)$

构造由  $x_0$  和  $x_0 + h$  确定的一次 Lagrange 插值多项式:

$$\begin{aligned} f(x) &= \frac{f(x_0)(x-x_0-h)}{x_0-x_0-h} + \frac{f(x_0+h)(x-x_0)}{x_0+h-x_0} + \frac{(x-x_0)(x-x_0-h)}{2!} f''(\xi_x) \\ f'(x) &= \frac{f(x_0+h)-f(x_0)}{h} + \frac{2(x-x_0)-h}{2} f''(\xi_x) + \frac{(x-x_0)(x-x_0-h)}{2!} \frac{d}{dx} f''(\xi_x) \\ f'(x_0) &= \frac{f(x_0+h)-f(x_0)}{h} - \frac{h}{2} f''(\xi_x) \end{aligned}$$

最后我们将  $f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} - \frac{h}{2} f''(\xi_x)$  用于误差估计,  $\delta = |\frac{h}{2} f''(\xi_x)|$

### 9.1.2 一般方法

对于  $n+1$  个点, 使用拉格朗日插值

$$\begin{aligned} f(x) &= \sum_{k=0}^n f(x_k) L_k(x) + \frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} f^{(n+1)}(\xi_x) \\ f'(x_j) &= \sum_{k=0}^n f(x_k) L'_k(x_j) + \frac{f^{(n+1)}(\xi_j)}{(n+1)!} \prod_{k=0, k \neq j}^n (x_j - x_k) \end{aligned}$$

总体而言, 更多的评估点会产生更高的准确性。另一方面, 功能评估的数量增加, 舍入误差也会增加。因此, 数值微分是不稳定的!

### 9.1.3 对于三个点的 Lagrange polynomial

$$\begin{aligned} f'(x_j) &= f(x_0) \frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)} + f(x_1) \frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)} \\ &\quad + f(x_2) \frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)} + \frac{f^{(3)}(\xi_j)}{3!} \prod_{k=0, k \neq j}^2 (x_j - x_k) \end{aligned}$$

如果三个点等间距, 即  $x_0, x_1, x_2$  等距, 即  $x_1 = x_0 + h, x_2 = x_0 + 2h$ , 则

$$\begin{aligned} f'(x_j) &= f(x_0) \frac{2x_j - 2x_0 - 3h}{2h^2} + f(x_1) \frac{2x_j - 2x_0 - 2h}{-h^2} \\ &\quad + f(x_2) \frac{2x_j - 2x_0 - h}{2h^2} + \frac{f^{(3)}(\xi_j)}{3!} \prod_{k=0, k \neq j}^2 (x_j - x_k) \end{aligned}$$

由此

$$\begin{aligned} f'(x_0) &= \frac{1}{h} \left( -\frac{3}{2} f(x_0) + 2f(x_0+h) - \frac{1}{2} f(x_0+2h) \right) + \frac{h^2}{3} f^{(3)}(\xi_0) \\ f'(x_1) &= \frac{1}{h} \left( -\frac{1}{2} f(x_0) + \frac{1}{2} f(x_0+2h) \right) - \frac{h^2}{6} f^{(3)}(\xi_1) \\ f'(x_2) &= \frac{1}{h} \left( \frac{1}{2} f(x_0) - 2f(x_0+h) + \frac{3}{2} f(x_0+2h) \right) + \frac{h^2}{3} f^{(3)}(\xi_2) \end{aligned}$$

可以看出，位于中间位置的导数，即  $f'(x_1) = f'(x_0 + h)$  的精度是相对来说比较好的点（而且还是只需要计算两次函数值就可以了），

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

#### 9.1.4 其他点数的数值微分计算

##### Five-Point Formulas

###### Five-Point Endpoint Formula

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

where  $\xi$  lies between  $x_0$  and  $x_0 + 4h$ .

###### Five-Point Midpoint Formula

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

where  $\xi$  lies between  $x_0 - 2h$  and  $x_0 + 2h$ .

## 10 示例

引理 10.1.

命题 10.2.  $adf$

定义 10.3.  $adsf$

例 10.4.  $asdfasdf$

在一个线性电路中，电流的积分与电压的积分之和等于零。

asdfas[1]

Code Listing 1: Python example

```
1 def hello_world():
2     print("Hello, world!")
```

## 参考文献

- [1] John Smith and Jane Doe. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2019:1–13, 2019.