数值分析课程笔记

Tan Dongliang
Changchun University of Science and Technology

更新：2026 年 1 月 17 日

1 绪论

算法源码

1.1 学习数值分析的重要性

数值分析的核心是**算法构造**和**误差分析**。数值分析能根据近似的数据，采用近似的方法获得**满足要求**的近似结果。

怎样使计算机的计算结果可信和尽量减少出错的情况，在科学计算中是非常重要的，因为错误的计算结果会产生错误的结论或否定原来正确的数学模型。

计算机解决实际问题的四个步骤：

1. 建立数学模型
2. **选择数值方法**
3. 编写程序
4. 上机计算

1.2 计算机中的数系与运算

1.2.1 机器数系的定义

数学上的十进制浮点数可表示为：

$$x = \pm 10^c \times 0.a_1a_2\cdots,\ a_i \in \{0,1,2,\cdots,9\}$$

类似的，β 进制的浮点数表示为：

$$x = \pm \beta^c \times 0.a_1a_2\cdots,\ a_i \in \{0,1,2,\cdots,\beta-1\}$$

$$x = \pm \beta^c \times 0.a_1a_2\cdots a_t,\ a_i \in \{0,1,2,\cdots,\beta-1\}$$

其中，t 为正整数，表示计算机的**字长**；c 为整数，表示**阶码**，满足 $L \le c \le U$，L 和 U 为固定整数。$0.a_1a_2\cdots a_t$ 称为**尾数**。

注 当且仅当 $a_1 \neq 0$ 时，称为规格化浮点数。

因此，在计算机系统中，实数系 $\mathbb{R}$ 是用所谓的浮点数系统 $\mathbb{F}$（机器数系）来近似表示的。

定义 1.1 (机器数系) 进制为 β 的浮点数系统表示为：

$$\mathbb{F}(\beta, t, L, U) = \{\pm\beta^c \times 0.a_1a_2\cdots a_t | a_i \in \{0, 1, 2, \cdots, \beta-1\}, L \leq c \leq U\}$$

机器数系是有限的离散集，其机器精度为 β^{-t}. 该数集整体在实数轴上的分布是不均匀的，但相同阶数的数又等距分布在实数轴的某一段上。

机器数系中有绝对值最大和最小的非零数 M 和 m，十进制浮点数系 $\mathbb{F}(10, 4, -99, 99)$，绝对值最大的非零数 $M = 10^{99} \times 0.9999$，绝对值最小的非零数 $m = 10^{-99} \times 0.0001$. 若数绝对值大于 M，产生**上溢错误**，小于 m，则产生**下溢错误**。上溢时，中断程序; 下溢时，用零表示该数继续程序；无论是上溢，还是下溢，都称为**溢出错误**。

表1列出了 IEEE（Institute of Electrical and Electronics Engineers）标准中规定的最常见的几种浮点数系统。

表 1: IEEE 浮点数系统

系统	进制 β	t	L	U
IEEE 单精度	2	24	-126	127
IEEE 双精度	2	53	-1022	1023
Cray	2	48	-16383	16384
HP calculator	2	12	-499	499
IBM mainframe	2	6	-64	63

1.2.2 机器数系的运算

计算机对输入数据 x（非零实数）的处理采用以下方式：

1. 若 $x \in \mathbb{F}(\beta, t, L, U)$，则原样接收 x
2. 若 $x \notin \mathbb{F}(\beta, t, L, U)$，但 $m \leq |x| \leq M$，则用 $\mathbb{F}(\beta, t, L, U)$ 中最接近 x 的数 $fl(x)$ 表示

机器数的运算规则如下：

- 加减法：**先对阶，后运算，**再舍入
- 乘除法：**先运算**，再舍入

案例 1 某计算机数系 $\mathbb{F}(10, 4, -99, 99)$ 的两个数 $x_1 = 0.2337 \times 10^{-1}$ 和 $x_2 = 0.3364 \times 10^2$，求这两个数相加、相乘的运算结果。

解：相加运算

$$\begin{aligned} fl(x_1 + x_2) &= fl(0.2337 \times 10^{-1} + 0.3364 \times 10^2) \\ &= fl(\underbrace{0.0002337 \times 10^2 + 0.3364 \times 10^2}_{\text{对阶}}) \\ &= fl(0.3366337 \times 10^2) \\ &= 0.3366 \times 10^2 \end{aligned}$$

相乘运算

$$\begin{aligned} fl(x_1 \times x_2) &= fl(0.2337 \times 10^{-1} \times 0.3364 \times 10^{2}) \\ &= fl(\underbrace{0.7861667 \times 10^{0}}_{\text{直接运算}}) \\ &= fl(\underbrace{0.7862 \times 10^{0}}_{\text{舍入}}) \\ &= 0.7862 \end{aligned}$$

注 计算机的运算一般在中央处理器（CPU）中运行，而 CPU 中的运算器一般是多倍字长存储，因此，参加运算的数据允许有超出原机器数系字长的数出现。计算结果需要保存在内存或硬盘，因此需要严格按照机器数系的大小进行截断或舍入处理。

案例 2 在字长为 8 的十进制浮点数系统下，给定 $x = 0.23371258 \times 10^{-4}, y = 0.33678429 \times 10^{2}, z = -0.33677811 \times 10^{2}$，分别计算 $(x+y)+z$ 和 $x+(y+z)$ 的结果。

解：

$$\begin{aligned} (x+y)+z &= fl((0.23371258 \times 10^{-4} + 0.33678429 \times 10^{2}) - 0.33677811 \times 10^{2}) \\ &= fl(\underbrace{0.00000023371258 \times 10^{2} + 0.33678429 \times 10^{2} - 0.33677811 \times 10^{2}}_{\text{对阶}}) \\ &= fl(0.00000641371258 \times 10^{2}) \\ &= 0.641 \times 10^{-3} \end{aligned}$$

$$\begin{aligned} x+(y+z) &= fl(0.23371258 \times 10^{-4} + (0.33678429 \times 10^{2} - 0.33677811 \times 10^{2})) \\ &= fl(0.23371258 \times 10^{-4} + \underbrace{(0.33678429 \times 10^{2} - 0.33677811 \times 10^{2})}_{\text{对阶}}) \\ &= fl(0.23371258 \times 10^{-4} + 0.61800000 \times 10^{-3}) \\ &= fl(\underbrace{0.023371258 \times 10^{-3} + 0.61800000 \times 10^{-3}}_{\text{对阶}}) \\ &= fl(0.641371258 \times 10^{-3}) \\ &= 0.64137126 \times 10^{-3} \end{aligned}$$

而 $x+y+z$ 的精确值为 $0.641371258 \times 10^{-3}$. 比较可以发现，$(x+y)+z \neq x+(y+z)$，而且 $x+(y+z)$ 要比 $(x+y)+z$ 精确的多。这是由于计算机上实行对位及有限字长导致的“大数吃小数”现象。因此，设计算法时，要尽可能避免一个很大的数和一个很小的数相加的问题。多个数相加时，应从绝对值较小的数依次加起，以避免有效数字的损失。

1.3 误差

1.3.1 误差的定义

精确值与近似值的差异就是误差。

定义 1.2 (绝对误差) 设 x 是准确值，x^* 是 x 的一个近似值，称差 x^*-x 为近似值 x^* 的绝对误差，简称误差，记为 e^* 或 $e(x^*)$，即 $e(x^*)=x^*-x$.

定义 1.3 (绝对误差限) 称满足 $|e^*|=|x^*-x|\leq\varepsilon^*$ 的正数 ε^* 或 $\varepsilon(x^*)$ 为近似值 x^* 的绝对误差限.

定义 1.4 (相对误差) 设 x 是准确值，x^* 是 x 的一个近似值，称 $\frac{e^*}{x}=\frac{x^*-x}{x}$ 为近似值 x^* 的相对误差，记为 e_r^* 或 $e_r(x^*)$，即

$$e_r(x^*)=\frac{e^*}{x}=\frac{x^*-x}{x}$$

相对误差的绝对值越小，近似程度越高。

定义 1.5 (相对误差限) 称满足 $|e_r^*|=|\frac{x^*-x}{x}|\leq\varepsilon_r^*$ 的正数 ε_r^* 或 $\varepsilon_r(x^*)$ 为近似值 x^* 的相对误差限.

1.3.2 数值计算的误差

定理 1.1 (四则运算的误差) 假设 x^* 和 y^* 分别是准确值 x 和 y 的一个近似值，则有

1. 四则运算的绝对误差估计：

$$\begin{aligned}
e\left(x^*\pm y^*\right)&=e\left(x^*\right)\pm e\left(y^*\right)\\
e\left(x^*y^*\right)&\approx y^*e\left(x^*\right)+x^*e\left(y^*\right)\\
e\left(\frac{x^*}{y^*}\right)&\approx\frac{y^*e\left(x^*\right)-x^*e\left(y^*\right)}{\left(y^*\right)^2}
\end{aligned}$$

2. 四则运算的相对误差估计：

$$\begin{aligned}
e_r\left(x^*\pm y^*\right)&\approx\frac{x^*e_r\left(x^*\right)\pm y^*e_r\left(y^*\right)}{x^*\pm y^*}\\
e_r\left(x^*y^*\right)&\approx e_r\left(x^*\right)+e_r\left(y^*\right)\\
e_r\left(\frac{x^*}{y^*}\right)&\approx e_r\left(x^*\right)-e_r\left(y^*\right)
\end{aligned}$$

定理 1.2 (多元函数的误差估计) 设多元函数 $u=f(x_1,x_2,\cdots,x_n)$，自变量 $(x_1,x_2,\cdots,x_n)$ 的近似值为 $(x_1^*,x_2^*,\cdots,x_n^*)$，则有多元函数 $f(x_1,x_2,\cdots,x_n)$ 的误差估计

$$\begin{aligned}
&(1)\ e(f(x_1^*,x_2^*,\cdots,x_n^*))\approx\sum_{i=1}^n\frac{\partial f(x_1^*,x_2^*,\cdots,x_n^*)}{\partial x_i}e(x_i^*)\\
&(2)\ \varepsilon(f(x_1^*,x_2^*,\cdots,x_n^*))\approx\sum_{i=1}^n\left|\frac{\partial f(x_1^*,x_2^*,\cdots,x_n^*)}{\partial x_i}\right|\varepsilon(x_i^*)\\
&(3)\ \varepsilon_r(f(x_1^*,x_2^*,\cdots,x_n^*))\approx\sum_{i=1}^n\left|\frac{\partial f(x_1^*,x_2^*,\cdots,x_n^*)}{\partial x_i}\right|\frac{\varepsilon(x_i^*)}{|f(x_1^*,x_2^*,\cdots,x_n^*)|}
\end{aligned}$$

1.3.3 计算机的计算误差

设计算机的数系为 $\mathbb{F}(\beta,t,L,U)$，m 及 M 是其中绝对值最小及最大的正数，某数 $x=\pm\beta^c\times0.a_1a_2\cdots,a_1\neq0$，满足 $m<|x|<M,x\notin\mathbb{F}(\beta,t,L,U)$，则计算机经舍入处理后以数 $fl(x)$ 接收，即

$$fl(x)=\pm\beta^c\times\tilde{a},\tilde{a}=\begin{cases}0.a_1a_2\cdots a_t & ,\quad 0\leqslant a_{t+1}<\beta/2\\ 0.a_1a_2\cdots a_t+\beta^{-t}, & a_{t+1}\geqslant\beta/2\end{cases}$$

因此计算机对 x 的计算绝对误差和计算相对误差有如下估计

$$(1)\ |e(\mathrm{fl}(x))| = |x - \mathrm{fl}(x)| \leqslant 0.5 \times \beta^{c-t}$$

$$(2)\ |e_r(\mathrm{fl}(x))| = \frac{|x - \mathrm{fl}(x)|}{|x|} \leq \frac{0.5 \times \beta^{c-t}}{0.1 \times \beta^{c}} = 0.5 \times \beta^{1-t}$$

由此可知，**计算机对任何实数的计算相对误差限与实数本身无关，只与计算机字长 t 有关**，其值为 $0.5 \times \beta^{1-t}$. 因此通常定义数 $eps = 0.5 \times \beta^{1-t}$ 为计算机的精度.

1.4 有效数字

1.5 数值分析中的常用概念

数值分析的研究内容有如下两个方面：

- **连续系统的离散化**，如求连续函数的定积分
- **离散型方程的数值求解**，主要体现在怎样将来自实际问题的各种微分方程转化为合适的离散型方程，以便能通过该离散化方程来求出原来微分方程的近似解

考察数值分析中的求解公式，会看到其中大部分内容都是在这两大方面所做的研究得出的。一个算法所需要的乘法和除法总次数称为计算量，常用 N 表示。**计算量**的单位为 flop，表示完成一次浮点数乘法或除法所需要的时间。初始数据的微小变化，导致计算结果的剧烈变化的问题称为**病态问题**。初始数据的微小变化只引起计算结果的微小变化的计算问题称为良态问题。

如果一个算法进行计算的初始数据有误差，而在**计算过程中**产生的误差不增长，则称该算法为数值稳定算法，否则称为数值不稳定算法。

科学计算中值得注意的四个方面：

- 避免相近二数相减
- 避免用接近零的数做除数
- 控制误差的积累和传播，避免大数吃小数，用数值稳定的算法
- 简化计算过程，化简公式或想办法减少运算次数

2 非线性方程的求根方法

一般情况下，绝大部分非线性方程没有求根公式。数值分析中所采用的求根方法一般分为**区间法**（如二分法）和**迭代法**（如简单迭代法、牛顿迭代法等）两类，其共同点都是**构造收敛于根的数列** $\{x_k\}$，不同的是产生 $\{x_k\}$ 的方法不一样。

定义 2.1 (非线性方程) *设 $f(x)$ 为一元连续函数，称方程 $f(x) = 0$ 为函数方程；当 $f(x)$ 不是 x 的线性函数时，称对应的函数方程为非线性方程。*

衡量一种求根方法的优劣，可由其产生的数列 $\{x_k\}$ 收敛于根 x^* 的快慢决定，为此引入收敛阶的概念。

定义 2.2 (收敛阶) *设数列 $\{x_k\}$ 收敛于 x^*，若存在正数 p 和 C，满足*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} = C$$

则称 $\{x_k\}$ 的收敛阶为 p 或 $\{x_k\}$ 收敛于 x^* 的速度是 p 阶的；同时称产生 $\{x_k\}$ 的方法具有 p 阶敛速。

注 本节所讨论的求根方法一次只能求一个根，对求方程多个根的问题，可利用函数 $f(x)$ 的特点先将根进行隔离，使每个区间只含有一个根，再利用本节求根方法分别求解。

2.1 二分法

基本思想：利用连续函数零点定理，将含根区间逐次减半缩小，构造出收敛点列来逼近根.

2.1.1 构造原理

微积分中的零点定理指出：若 $f(x) \in C[a,b]$，且满足 $f(a)f(b) < 0$，则在区间 (a,b) 中至少有一点 ξ，使得 $f(\xi) = 0$. 将含根区间分为两个长度相等的子区间后，在这两个子区间上也可利用零点定理确定根在哪个子区间上，**在这一系列子区间中依次选择一个点就可以构造出数列 $\{x_k\}$，由于含根区间是以逐次减半的方式减小并最后趋于零，因此数列 $\{x_k\}$ 会收敛于根 x^*.**

二分法构造过程：
(1) 记含根区间 $I_0 = [a,b]$，取 I_0 中点 $x_0 = 0.5(a+b)$，并计算 $f(x_0)$;
(2) 判别 $f(x_0)$ 的值，
若 $f(x_0) = 0$，则 $x^* = x_0$，终止;
若 $f(a)f(x_0) < 0$，则 $x^* \in [a, x_0]$，取 $I_1 = [a, x_0]$;
若 $f(a)f(x_0) > 0$，则 $x^* \in [x_0, b]$，取 $I_1 = [x_0, b]$;
(3) 记 $I_1 = [a_1, b_1]$，取 I_1 中点 $x_1 = 0.5(a_1 + b_1)$，
若 x_1 满足根的精度要求，取 $x^* \approx x_1$，终止；否则用 I_1 代替 I_0, 转 (1).

2.1.2 收敛性

因为根 $x^* \in I_k$ $(k = 0, 1, \cdots)$，且 $I_0 \supset I_1 \supset I_2 \supset \cdots$，所以有 x_k 的误差满足

$$|x^* - x_k| \le \frac{1}{2}(b_k - a_k) = \frac{1}{2^2}(b_{k-1} - a_{k-1}) = \cdots = \frac{1}{2^{k+1}}(b-a).$$

于是当 $k \to \infty$ 时，由 $\frac{1}{2^{k+1}}(b-a) \to 0$ 得到 $x_k \to x^*$，即二分法产生的数列 $\{x_k\}$ 收敛于根 x^*. 给定精度 $\epsilon > 0$ 后，可以使用 $K = \lceil \frac{\ln(b-a) - \ln\epsilon}{\ln 2} - 1 \rceil$ 或 $|x_k - x_{k-1}| \le \epsilon$ 作为终止条件，此时，x_k 是满足精度的根，证明略。

2.2 简单迭代法

简单迭代法是非线性方程求根方法中各类迭代法的基础，也可以很方便的扩展应用到方程（组）求解问题中。

基本思想：将方程 $f(x) = 0$ 等价变形为 $x = \varphi(x)$，获得迭代公式 $x_{k+1} = \varphi(x_k)$，再由它计算出逼近根 x^* 的数列 $\{x_k\}$.

2.2.1 构造原理

若 x^* 为根，则有 $f(x^*)=0$，故 $x^*=\varphi(x^*)$ 成立，说明根 x^* 可以经过函数 $\varphi(x)$ 计算出来。

证明. 当 $\varphi(x)$ 产生的迭代数列 $\{x_k\}$ 收敛时，一定收敛到根 x^*. 设 $\lim_{k\to\infty} x_k=\bar{x}$, 且迭代函数 $\varphi(x)$ 一般是连续的，利用连续函数与极限的关系，有

$$\bar{x}=\lim_{k\to\infty} x_{k+1}=\lim_{k\to\infty}\varphi(x_k)=\varphi(\lim_{k\to\infty} x_k)=\varphi(\bar{x}),$$

这说明迭代数列 $\{x_k\}$ 的极限就是所求的根，即 $\bar{x}=x^*$.

简单迭代法构造过程：
(1) 将方程 $f(x)=0$ 改写为等价形式 $x=\varphi(x)$;
(2) 构造迭代公式 $x_{k+1}=\varphi(x_k)$;
(3) 取初值 x_0，由迭代公式计算数列 $x_1=\varphi(x_0), x_2=\varphi(x_1),\cdots$.

数列 $\{x_k\}$ 称为迭代数列，$\varphi(x)$ 为迭代函数。对于根 x^*，$x^*=\varphi(x^*)$，即 $\varphi(x)$ 变不动它，这样的点 (x^*) 形象地称为 $\varphi(x)$ 的不动点，方程 $x=\varphi(x)$ 为不动点方程。因此，有时也说非线性方程求根问题就是求对应的迭代函数的不动点问题。

实际中经常使用的是加速的 Aitken 迭代法，这里直接给出算法流程：

Require: 初值 x^0，根的精度 ϵ, 迭代函数 $\varphi(x)$
Ensure: 方程的根 x^*
1: $x_k \leftarrow x_0$
2: 计算 $x_{k+1}^{(1)}=\varphi(x_k)$，$x_{k+1}^{(2)}=\varphi(x_{k+1}^{(1)})$
3: 计算 $x_{k+1}=x_k-\frac{(x_{k+1}^{(1)}-x_k)^2}{x_{k+1}^{(2)}-2x_{k+1}^{(1)}+x_k}$
4: 判定 x_{k+1} 是否满足精度 ϵ，满足则输出根 $x^*=x_{k+1}$；否则，$x_k\leftarrow x_{k+1}$, 返回步骤 2.

2.2.2 收敛性

简单迭代法通常是线性收敛的，个别情况下可以达到超线性收敛或更高阶收敛。以下定理证明，当迭代函数 $\varphi(x)$ 满足一定条件时，其产生的迭代数列是收敛的，迭代公式可以收敛到根 x^*.

定理 2.1 (Lipschitz continuity) 设迭代函数 $\varphi(x)$ 满足下列两个条件：

(1) 当 $x\in[a,b]$ 时，有 $\varphi(x)\in[a,b]$;

(2) 任取 $x_1,x_2\in[a,b]$，存在与 x_1 和 x_2 无关的正常数 $L<1$，满足

$$|\varphi(x_1)-\varphi(x_2)|\le L|x_1-x_2| \tag{1}$$

则 $\varphi(x)$ 在 $[a,b]$ 中有唯一不动点 x^*，且迭代公式 $x_{k+1}=\varphi(x_k)$ 对任取 $x_0\in[a,b]$，产生的数列 $\{x_k\}$ 都收敛于 x^*.

证明. 作辅助函数 $\psi(x)=x-\varphi(x)$. **（存在性）**由条件（1）知 $\psi(a)\psi(b)\le 0$. 由零点定理，存在 $\xi\in[a,b]$，使 $\psi(\xi)=0$，由此得出 $\xi=\varphi(\xi)$，说明 $\varphi(x)$ 在 $[a,b]$ 上有不动点 ξ. **（唯一性）**假设存在另一不动点 η，有 $\eta=\varphi(\eta)$，利用条件（2）可得

$$|\xi-\eta|=|\varphi(\xi)-\varphi(\eta)|\le L|\xi-\eta|<|\xi-\eta|,$$

等式前后矛盾，因此满足定理条件下，$\varphi(x)$ 在 $[a,b]$ 有唯一不动点。**（收敛性）**利用 x^* 是不动点、x_k 的迭代公式及条件（2），有

$$|x_k-x^*|=|\varphi(x_{k-1})-\varphi(x^*)|\le L|x_{k-1}-x^*|\le L^2|x_{k-2}-x^*|\le L^k|x_0-x^*|,$$

注意到 $0<L<1$，因此，当 $k\to\infty$ 时，可得 $L^k\to 0$，因此有

$$\lim_{k\to\infty}|x_k-x^*|=0$$

故 $\lim_{k\to\infty}x_k=x^*$.

定理 2.2 (推论) *设迭代函数 $\varphi(x)$ 满足：*

(1) *当 $x\in[a,b]$ 时，有 $\varphi(x)\in[a,b]$;*

(2) *存在正常数 $L<1$，满足 $|\varphi'(x)|\le L$*

则 $\varphi(x)$ 在 $[a,b]$ 中有唯一不动点 x^*，对任取 $x_0\in[a,b]$，产生的数列 $\{x_k\}$ 都收敛于 x^*. 证明略。

2.2.3 误差估计

在给定精度 $\epsilon>0$ 后，要使 $|x^*-x_k|<\epsilon$，只要计算到 $\frac{L}{1-L}|x_k-x_{k-1}|<\epsilon$ 或 $\frac{L^k}{1-L}|x_1-x_0|<\epsilon$ 即可。也可以直接求出满足要求的迭代次数：

$$K=\left\lceil \ln\left|\frac{(1-L)\epsilon}{x_1-x_0}\right| / \ln L\right\rceil.$$

实际中都用 $|x_k-x_{k-1}|\le\epsilon$ 作为终止条件，通常也能求出满足精度要求的根。

证明略。

2.3 Newton 迭代法

牛顿迭代法（Newton's method）又称切线法或牛顿-拉夫逊方法（Newton-Raphson method），是牛顿在研究求根问题时借助大胆近似技术得出的目前求根方法中收敛最快的方法。

基本思想：将函数 $f(x)$ 做线性化处理（将 $f(x)$ 在某点展开为 Taylor 公式后，取其线性部分 $L(x)$ 代替 $f(x)$），把方程 $f(x)=0$ 转化为对应的近似方程 $L(x)=0$，再从 $L(x)=0$ 中构造迭代公式。

牛顿迭代法具有超线性收敛速度，但实际上它一般具有**平方收敛**速度。

2.3.1 构造原理

Newton 迭代法通过不断求解原函数的近似函数方程来构造迭代数列，直至收敛。

Newton 迭代法构造过程：

(1) 将方程 $f(x)$ 在 x_k 处做 Taylor 展开

$$f(x)=f(x_k)+f'(x_k)(x-x_k)+\frac{f''(x_k)}{2!}(x-x_k)^2+\cdots$$

(2) 取一阶线性部分 $L(x)=f(x_k)+f'(x_k)(x-x_k)$，求解近似方程 $L(x)=0$，有

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

(3) 从步骤（2）中求得解，记为 x_{k+1}，得到迭代关系式

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \tag{2}$$

式2称为 Newton 迭代公式，用此公式求方程 $f(x)=0$ 根的方法称为 Newton 迭代法。

2.3.2 收敛性

实际上 Newton 迭代法需要选择一个尽可能靠近 x^* 的初值才能保证收敛。

以下定理给出了牛顿迭代法收敛的充分条件：

定理 2.3 (Newton 迭代法收敛的充分条件) 设函数 $f(x)\in C^2[a,b]$，满足下列三个条件：

(1) $f(a)f(b)<0$;

(2) $f'(x)\neq 0$，$x\in(a,b)$;

(3) $f''(x), x\in(a,b)$ 存在且不变号.

则在 $[a,b]$ 内任取一点 x_0，只要 $f(x_0)f''(x_0)>0$，Newton 迭代法产生的数列一定收敛于 $[a,b]$ 上的唯一根 x^*.

2.3.3 Newton 迭代法的变形与推广

Newton 迭代法的变形：割线法 为了克服 Newton 迭代法需要求导数 $f'(x)$ 的不便，采用 $f'(x)$ 的近似值

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

将其代入牛顿迭代公式2可得割线法的迭代公式

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

割线法需要两个初值才能产生迭代数列，因此是多点（两点）迭代公式。割线法是超线性收敛的，收敛阶为 1.618.

Newton 迭代法的推广：求解非线性方程组 变元数为 n 的非线性方程组的向量形式为

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

其中，

$$\boldsymbol{F}(\boldsymbol{x})=\begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{bmatrix}=\begin{bmatrix} f_1\left(x_1,x_2,\cdots,x_n\right) \\ f_2\left(x_1,x_2,\cdots,x_n\right) \\ \vdots \\ f_n\left(x_1,x_2,\cdots,x_n\right) \end{bmatrix},\boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

非线性方程组的 Newton 迭代公式为

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\left(\nabla_{\boldsymbol{x}}\boldsymbol{F}\right)^{-1}\boldsymbol{F}(\boldsymbol{x}_k)$$

3 线性方程组的解法