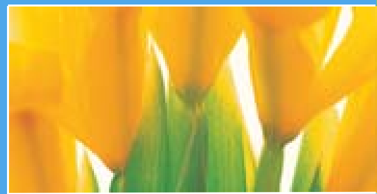
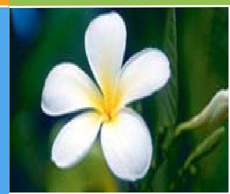
第二章 非线性方程的数值解法

Solutions of Nonlinear Equations





龚怡

2167570874@qq.com

计算机学院

线性方程 和 非线性方程

线性方程 指未知数都是一次的方程，也称为一次方程 。因为在笛卡尔坐标系上任何一个一次方程的表示都是一条直线。

其一般的形式是$ax + by + ... + cz + d = 0$。

组成一次方程的每个项必须是 常数 或者是一个常数和一个变量的乘积，且方程中必须包含一个变量。

非线性方程 就是因变量与自变量之间的关系不是线性的关系。

这类方程很多，例如平方关系、对数关系、指数关系、三角函数关系等等。

此类方程往往很难得到精确解，经常需要求近似解问题。

问题的提出

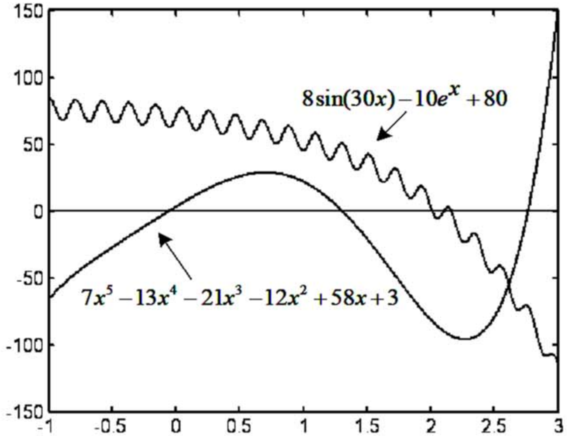
高次（$n \geqslant 2$）的代数方程，或者含有三角函数、指数函数等超越函数的超越方程，都统称为非线性方程。非线性方程一般没有现成的求根公式，对于超越方程来说，就连有没有根、有几个根也难以判断。

n次代数方程

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$$

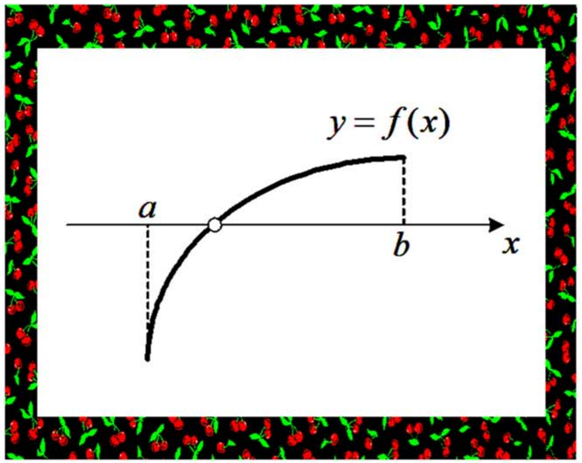
超越方程

$$e^{-x} + \ln x - \sin \frac{\pi x}{2} = 0$$



如果一些根据实际问题列出的方程是非线性方程，有什么方法可以求解呢？又或者怎样能够得到比较精确的近似解？

有多种常用的数值计算方法可以逼近解，如二分法、迭代法、牛顿法……



主要内容

非线性方程的数值解法

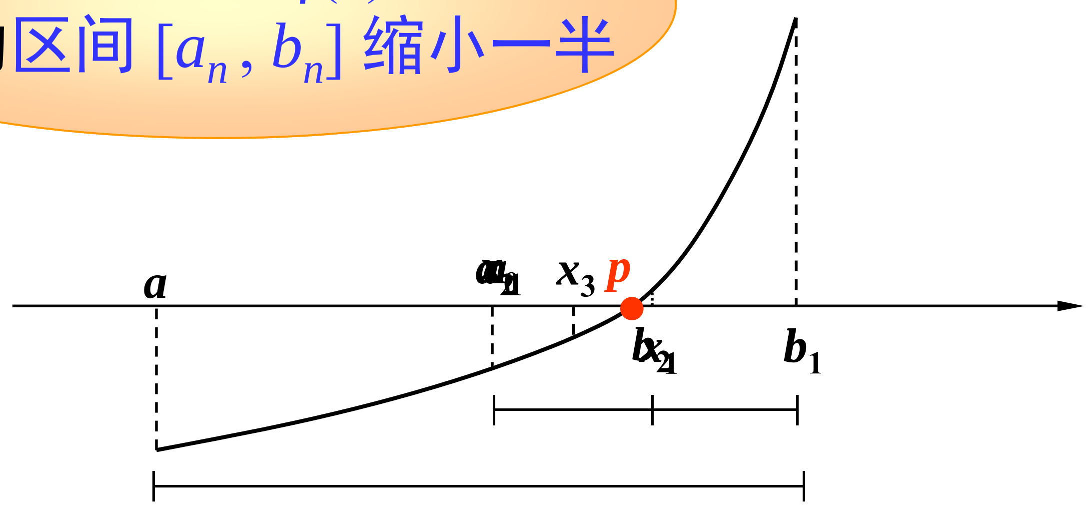
- 二分算法
- 迭代法
  - 一般迭代法
  - 迭代算法理论
  - 加速收敛迭代法
- 牛顿切线法
- 弦截法
  - 定端点弦截法
  - 动端点弦截法

2.1.1二分法（Bisection or Binary-search method）

❖ 原理：若 $f \in C[a, b]$，且 $f(a) \cdot f(b) < 0$，则 f 在 (a, b) 上必有一根。

每次迭代都使得 $f(x)=0$ 的解所在的区间 $[a_n, b_n]$ 缩小一半

什么时候停止？



2.1.1二分法（Bisection or Binary-search method）

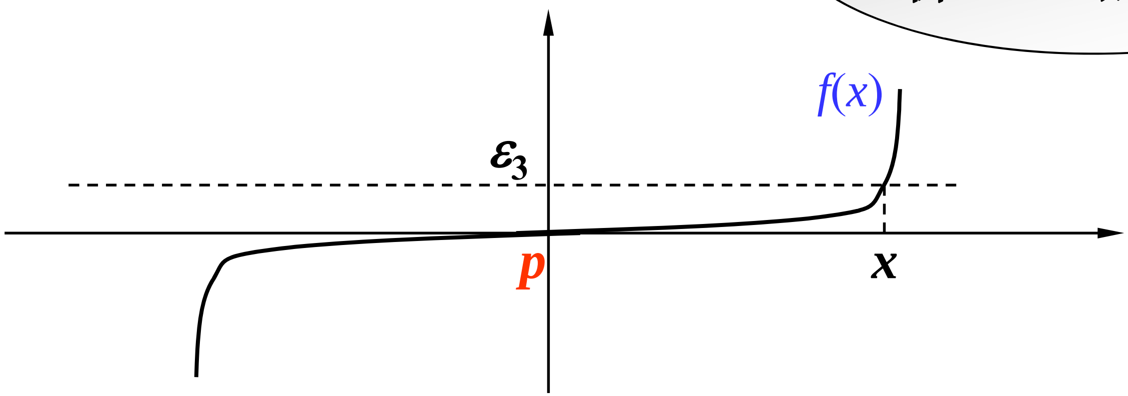
❖ 停止条件有 3 种常用的方式：

（1） $|x_n - x_{n-1}| < \varepsilon_1$ 　　相邻两个**x**的绝对误差

（2） $\dfrac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon_2$ 　　$x_n \neq 0$ 　　相邻两个**x**的相对误差

（3） $|f(x_n)| < \varepsilon_3$ 　　函数值误差

第3种不能保证 ***x*** 的精度



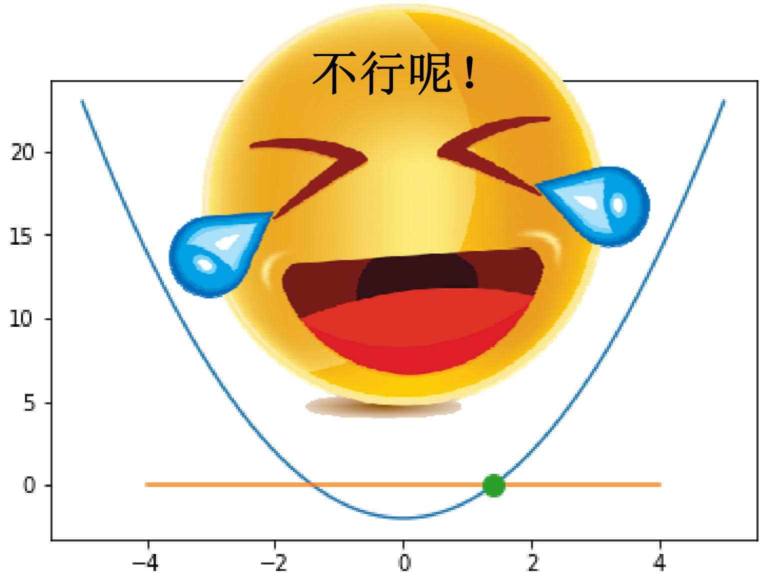
2.1.1二分法（Bisection or Binary-search method）

❖ 原理：若 $f \in C[a, b]$，且 $f(a) \cdot f(b) < 0$，则 f 在 (a, b) 上必有一根。

- **例：** 用二分法求解方程在[1.3, 1.5]上的解。

$$f(x) = x^2 - 2 = 0$$

```python
iter = 0        #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT:
    xmid = (xlow + xup) / 2       #计算新的中
    iter += 1                     #迭代计数加
    if f(xmid) > 0:               #中点函数值
        xup = xmid                #更新xup
    else:                         #中点函数值
        xlow = xmid               #更新xlow
```



❖ 请问：第1次上机课的二分法程序，可以直接用于求解区间 [-2,0] 上的解吗？

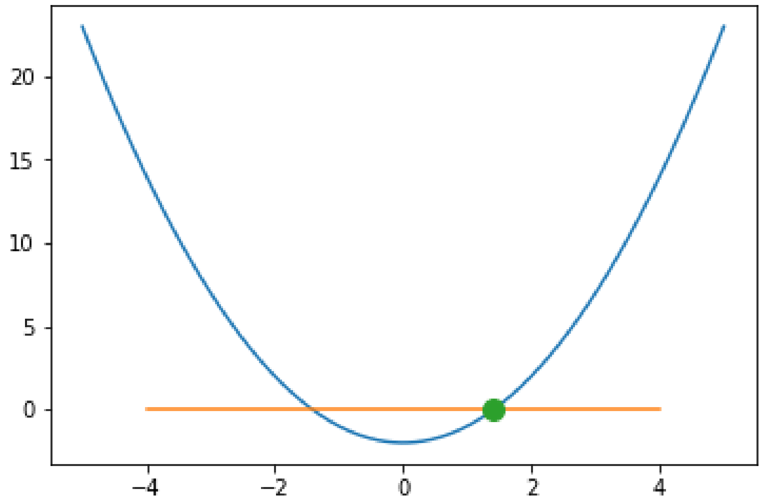
2.1.1二分法（Bisection or Binary-search method）

❖ 原理：若 $f \in C[a, b]$，且 $f(a) \cdot f(b) < 0$，则 f 在 (a, b) 上必有一根。

■ **原因：程序中的中点函数值>0，则区间新上限值置为中点值。这只在[0, 2]区间上的解成立。**

在[-2, 0]区间，需要对调赋值语句

```
iter = 0        #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT:
    xmid = (xlow + xup) / 2       #计算新的中
    iter += 1                     #迭代计数加
    if f(xmid) > 0:               #中点函数值
        xup = xmid                #更新xup
    else:                         #中点函数值
        xlow = xmid               #更新xlow
```



2.1.1二分法（Bisection or Binary-search method）

- 原理：若 $f \in C[a, b]$，且 $f(a) \cdot f(b) < 0$，则 f 在 (a, b) 上必有一根。
- 在编写程序的时候还需要注意计算的机器精度
  - 尽量减少误差

1. 判断区间 $[a, b]$ 内是否有根

直接使用 $f(a) \cdot f(b) < 0$ 不好

应该用 $\mathrm{sign}(f(a)) \cdot \mathrm{sign}(f(b)) < 0$

符号函数：

$$\mathrm{sign}(x) = \begin{cases} -1, & \text{if} \quad x < 0 \\ 0, & \text{if} \quad x = 0 \\ 1, & \text{if} \quad x > 0 \end{cases}$$

原因是当避免乘积越界导致错误

2.1.1二分法（Bisection or Binary-search method）

- 在编写程序的时候需要注意计算的机器精度
  - 尽量减少误差

2. 计算中间点

直接使用 $x_n = \dfrac{a_n + b_n}{2}$ 不好

应该用

$$x_n = a_n + \frac{b_n - a_n}{2}$$

原因是当 a_n 和 b_n 都接近机器的精度极限时，计算 $x_n = \dfrac{a_n + b_n}{2}$ 得到的值可能越出 $[a_n, b_n]$ 的范围

修订后的二分法代码

#程序2.1

```
import numpy as np

a = 2          # f(x) = x*x - a
LIMIT = 1e-20  #终止条件

#方程函数f()定义
def f(x):
    """函数值的计算"""
    return x * x - a
# f()函数结束

#-----主执行部分------
#初始区间设置
xlow = float(input("请输入x值下限:"))
xup = float(input("请输入x值上限:"))
while xup <= xlow:     #要求上限必须比下限值大
    print("请输入比下限%g大的值"%xlow)
    xup = float(input("请输入x值上限:"))

#循环处理
iter = 0      #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT: #满足终止条件前循环
    xmid = xlow + (xup - xlow) / 2      #计算新的中值点
    iter += 1                           #迭代计数加1
    if np.sign(f(xmid)) * np.sign(f(xlow)) < 0:    #中点与下限对应函数异号
        xup = xmid                      #更新xup
    else:                               #中点函数值为负
        xlow = xmid                     #更新xlow
    print("{:.15g},{:.15g},{:.15g}".format(iter,xlow, xup))
```

所有输入都必须检验边界，防止人为出错

数值变量化、流程通用化才可扩展

误差分析

定理2.1 二分法每次迭代值与实际解之间的误差

定理 2.1 设 $f\in C[a,b]$， $f(a)f(b)<0$，则二分法产生的数列 $\{x_n\}$ 满足

$$|x_n-p|\le\frac{b-a}{2^{n+1}}$$

其中 $p\in[a,b]$ 是 $f(x)=0$ 的根， $n=0,1,\cdots$。

证明：根据二分法的性质可知 $[a_n,b_n]$ 是由 $[a_{n-1},b_{n-1}]$ 二分得到的，即

$$b_n-a_n=(b_{n-1}-a_{n-1})/2=\cdots=(b_0-a_0)/2^n=(b-a)/2^n,$$

而 $p\in[a_n,b_n]$，且 $x_n=(a_n+b_n)/2$，所以

$$|x_n-p|\le x_n-a_n=(a_n+b_n)/2-a_n=(b_n-a_n)/2=(b-a)/2^{n+1}$$

定理证毕。

误差分析

定理2.1 二分法每次迭代值与实际解之间的误差

第1步产生的 $x_0 = \dfrac{a_0 + b_0}{2}$ 有误差 $|x_0 - p| \le \dfrac{b_0 - a_0}{2}$

第2步产生的 $x_1 = \dfrac{a_1 + b_1}{2}$ 有误差 $|x_1 - p| \le |x_1 - a_1|$

$$= \frac{a_1 + b_1}{2} - a_1$$

$$= \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}$$

第 **n** 步产生的 $\boldsymbol{x_n}$ 有误差 $|x_n - p| \le \dfrac{b-a}{2^{n+1}}$

对于给定的精度 ε,可估计二分法所需的步数 $\boldsymbol{n}+1$ ：

$$\frac{b-a}{2^{n+1}} < \varepsilon \quad \Rightarrow \quad n+1 \ge \left\lceil \frac{\ln(b-a) - \ln\varepsilon}{\ln 2} \right\rceil$$

二分法求解例子

• 例2.1 用二分法求方程

$$7x^5 - 13x^4 - 21x^3 - 12x^2 + 58x + 3 = 0$$

• 在区间[1, 2]上的根，设定停止条件为相邻求值得到的解之间的误差绝对值小于 $\varepsilon = 10^{-5}$，先估计需要的求解步数，并比较实际需要的求解步数。

解：记 $f(x) = 7x^5 - 13x^4 - 21x^3 - 12x^2 + 58x + 3$

有 $f(1) > 0$ 和 $f(2) < 0$，所以 f 在区间[1, 2]上有根。根据公式可得估计需要的求解步数为

$$n + 1 \geq \left\lceil \frac{\ln(b-a) - \ln\varepsilon}{\ln 2} \right\rceil = \left\lceil \frac{\ln(2-1) - \ln 10^{-5}}{\ln 2} \right\rceil = 17$$

二分法求解例子

- 例2.1 用二分法求方程

$$7x^5-13x^4-21x^3-12x^2+58x+3=0$$

- 在区间[1, 2]上的根，设定停止条件为相邻求值得到的解之间的误差绝对值小于$\varepsilon=10^{-5}$，先估计需要的求解步数，并比较实际需要的求解步数。

解：记 $f(x)=7x^5-13x^4-21x^3-12x^2+58x+3$

第一步，求[1, 2]的中点 $x_0=\dfrac{(1+2)}{2}=1.5$，得 $f(x_0)<0$

第二步，求[1,1.5]的中点 $x_1=\dfrac{(1+1.5)}{2}=1.25$，得 $f(x_1)>0$

类推结果见表2.1

表 2.1

n	a_n	b_n	x_n	$\|f(x_n)\|$	$\|x_n-x_{n-1}\|<\varepsilon$	$\|x_n-x_{n-1}\|/\|x_n\|<\varepsilon$
0	1.00000	2.00000	1.50000	20.53130	0.50000	0.33333
1	1.00000	1.50000	1.25000	5.35840	0.25000	0.20000
2	1.25000	1.50000	1.37500	6.59311	0.12500	0.09091
3	1.25000	1.37500	1.31250	0.34135	0.06250	0.04762
4	1.25000	1.31250	1.28125	2.58030	0.03125	0.02439
5	1.28125	1.31250	1.29688	1.13710	0.01563	0.01205
6	1.29688	1.31250	1.30469	0.40224	0.00781	0.00599
7	1.30469	1.31250	1.30859	0.03153	0.00391	0.00299
8	1.30859	1.31250	1.31055	0.15464	0.00195	0.00149
9	1.30859	1.31055	1.30957	0.06149	0.00098	0.00075
10	1.30859	1.30957	1.30908	0.01496	0.00049	0.00037
11	1.30859	1.30908	1.30884	0.00829	0.00024	0.00019
12	1.30884	1.30908	1.30896	0.00334	0.00012	9.32575e-005
13	1.30884	1.30896	1.3089	0.00248	6.10352e-005	4.66309e-005
14	1.30890	1.30896	1.30893	0.00043	3.05176e-005	2.33149e-005
15	1.30890	1.30893	1.30891	0.00102	1.52588e-005	1.16576e-005
16	1.30891	1.30893	1.30892	0.00030	7.62939e-006	5.82876e-006

表 2.1

n	a_n	b_n	x_n	$\|f(x_n)\|$	$\|x_n-x_{n-1}\|<\varepsilon$	$\|x_n-x_{n-1}\|/\| x_n \|<\varepsilon$
0	1.00000	2.00000	1.50000	20.53130	0.50000	0.33333
1	1.00000	1.50000	1.25000	5.35840	0.25000	0.20000
2	1.25000	1.50000	1.37500	6.59311	0.12500	0.09091
3	1.25000	1.37500	1.31250	0.34135	0.06250	0.04762
4	1.25000	1.31250	1.28125	2.58030	0.03125	0.02439
5	1.28125	1.31250	1.29688	1.13710	0.01563	0.01205
6	1.29688	1.31250	1.30469	0.40224	0.00781	0.00599
7	1.30469	1.31250	1.30859	0.03153	0.00391	0.00299
14	1.30890	1.30896	1.30893	0.00043	3.05176e-005	2.33149e-005
15	1.30890	1.30893	1.30891	0.00102	1.52588e-005	1.16576e-005
16	1.30891	1.30893	$\boxed{1.30892}$	0.00030	$\boxed{7.62939e-006}$	5.82876e-006

取停止条件为$|x_n - x_{n-1}|<\varepsilon$，在$\varepsilon=10^{-5}$下得到的解为 1.30892。实际的求解步数为 17，与估计的值一致。比较表中的后三列，可以看出 3 种判断停止条件的差别，其中$|f(x_n)|$的值并不是单调变化的，而另外两种方法得到的数据都单调递减，且与解的误差精度变化情况相近。

二分法的优缺点

①简单，总会收敛到解

② 对$f\,(x)$ 要求不高(只要连续即可)

①无法求复根及偶重根

② 收敛慢，只利用函数值的正负

用二分法求根，最好先给出 $f\,(x)$ 草图以确定根的大概位置。或用搜索程序，将$[a, b]$分为若干小区间，对每一个满足 $f\,(a_k)\cdot f\,(b_k) < 0$ 的区间调用二分法程序，可找出区间$[a, b]$内的多个根，且不必要求 $f\,(a)\cdot f\,(b) < 0$ 。

主要内容

非线性方程的数值解法

- 二分算法
- 迭代法
  - 一般迭代法
  - 迭代算法理论
  - 加速收敛迭代法
- 牛顿切线法
- 弦截法
  - 定端点弦截法
  - 动端点弦截法

2.2.1 迭代法（Iteration Method)

现在要在区间$[a,b]$上解方程$f(x)=0$，可以把它改写成等价形式

$$x=\varphi(x)，\quad a\le x\le b$$

其中$\varphi\in C[a,b]$。对于$\forall x_0\in[a,b]$，定义数列$\{x_n\}$具有形式

$$x_n=\varphi(x_{n-1})，\quad n=1,2,\cdots$$

这个数列$\{x_n\}$称为解方程$f(x)=0$的一个**迭代数列**，$x_n=\varphi(x_{n-1})$称为解方程$f(x)=0$的一个**迭代格式**，x_0称为**迭代初值**。

如果具有$x_n=\varphi(x_{n-1})$形式的数列$\{x_n\}$收敛，并且假设$\lim\limits_{n\to\infty}x_n=p$，则有

$$p=\lim_{n\to\infty}x_{n+1}=\lim_{n\to\infty}\varphi(x_n)=\varphi(\lim_{n\to\infty}x_n)=\varphi(p)$$

此时p即为方程$f(x)=0$的根。

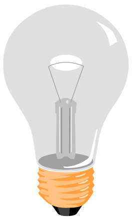
采用$x_n=\varphi(x_{n-1})$的迭代形式求解的方法称为**一般迭代法**，也称为**不动点迭代法**（Fixed-Point Iteration）或**函数迭代法**（Functional Iteration）。

2.2.1 迭代法（Iteration Method)

❖ 一般迭代法（不动点迭代法Fixed-point Iteration）

$$f(x)=0 \xleftrightarrow{\text{等价变换}} x=\varphi(x)$$

$f(x)$ 的根　　　　$\varphi(x)$ 的不动点



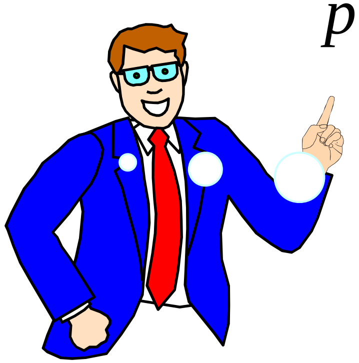
思路

从一个初值x_0出发，计算 $x_1=\varphi(x_0)$，$x_2=\varphi(x_1)$，…

$x_n=\varphi(x_{n-1})$，若 $\{x_n\}_{n=0}^{\infty}$ 收敛，即存在 p 使得

$\lim\limits_{n\to\infty} x_n = p$，且$\varphi$ 连续，则由 $\lim\limits_{n\to 0} x_{n+1} = \lim\limits_{n\to 0}\varphi(x_n)$ 可知

$p=\varphi(p)$，即 p 是φ 的不动点，也就是 f 的根。



真有这么简单？那你
知道什么是收敛？

定义2.1

- 对于函数φ, 若点p使得$\varphi(p)=p$，那么称p为φ 的不动点。

定理2.2 若函数 $\varphi\in[a,b]$且对于$\forall x\in[a,b]$有 $\varphi(x)\in[a,b]$,

那么 φ在$[a, b]$上有不动点。

定理2.3 除了满足定理2.2外，如果函数φ满足李普希茨条件

$$|\varphi(x)-\varphi(y)|\le L|x-y|\ ,\quad \forall x,y\in[a,b]$$

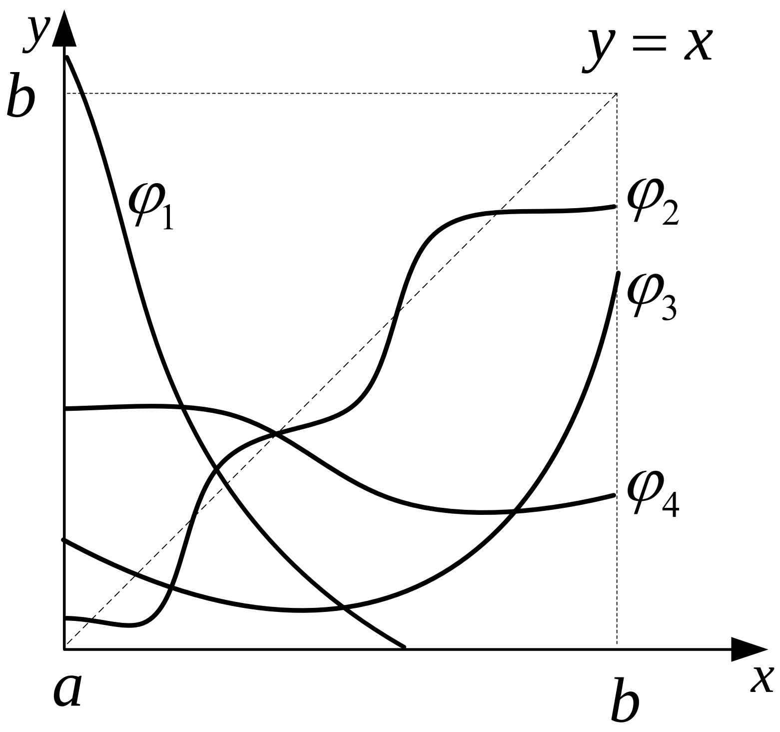
并且常数$0<L<1$，那么 φ在$[a, b]$上有唯一的不动点。

定理2.4 除了满足定理2.2外，如果函数 φ 在(a, b)上有一阶连续导数，并且存在常数$0<L<1$使得

$$|\varphi'(x)|\le L\ ,\qquad \forall x\in[a,b]$$

那么φ在$[a, b]$上有唯一的不动点。

不动点几何意义比较图示

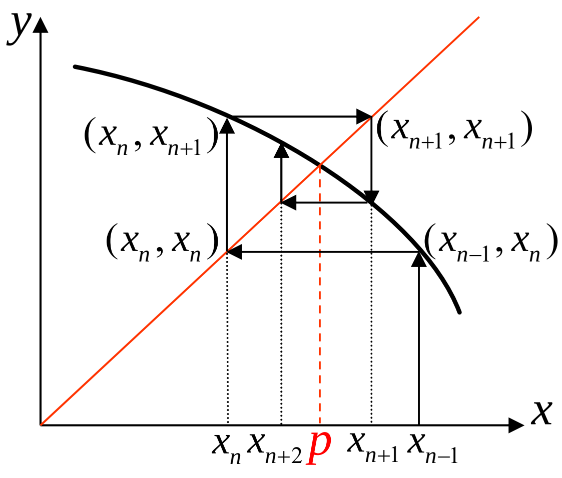


不动点的几何意义实际上就是在区间$[a,b]$上，函数φ 与$y=x$直线的交点。左图画出了4条函数曲线，每条曲线在区间$[a,b]$上都有不动点。

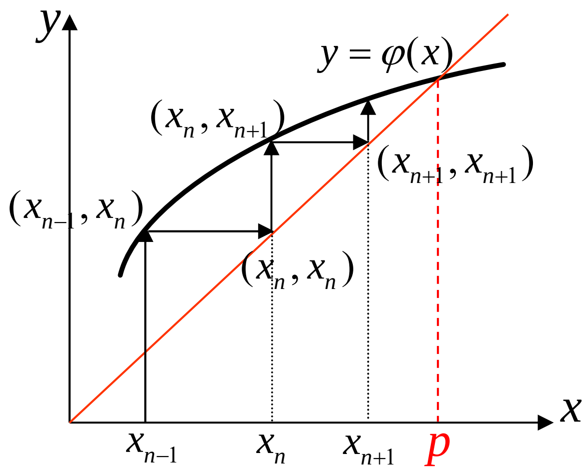
除了φ_1外，另外三个函数都满足定理2.2。由于φ_2和φ_3在区间$[a,b]$上都存在某些x使得$|\varphi'(x)|>1$的情况，因此这两个函数都不满足定理2.3和定理2.4。

从直观上可以看出φ_4在区间$[a,b]$上的斜率绝对值都小于1，因此满足定理2.4的要求，不动点是存在且唯一的。

迭代法的几何意义



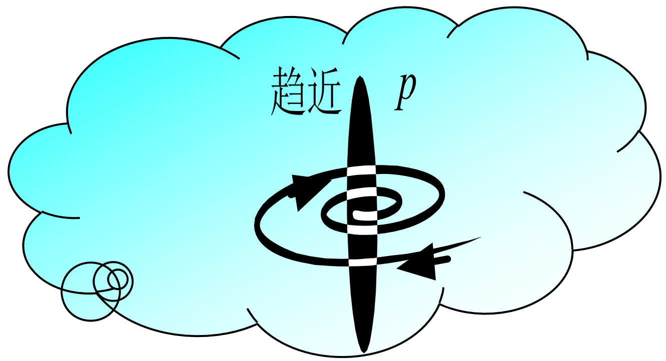
收敛

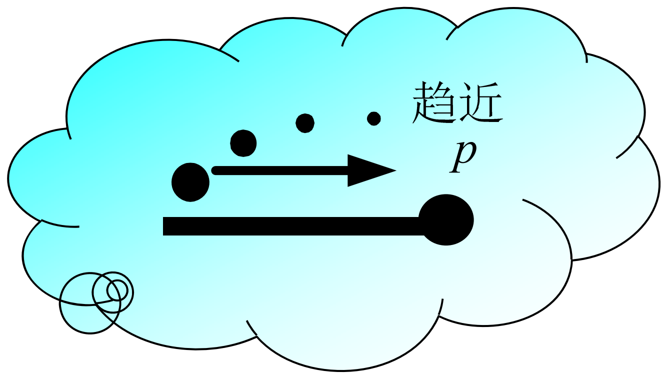


收敛

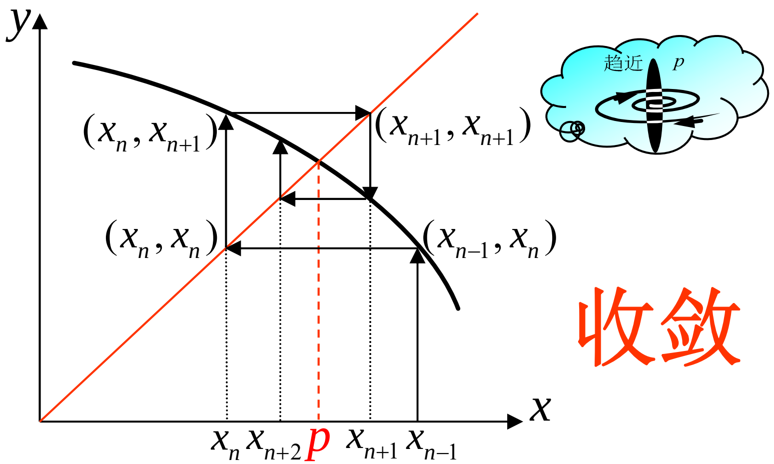
1) $-1<\varphi'(x)<0$

2) $0<\varphi'(x)<1$

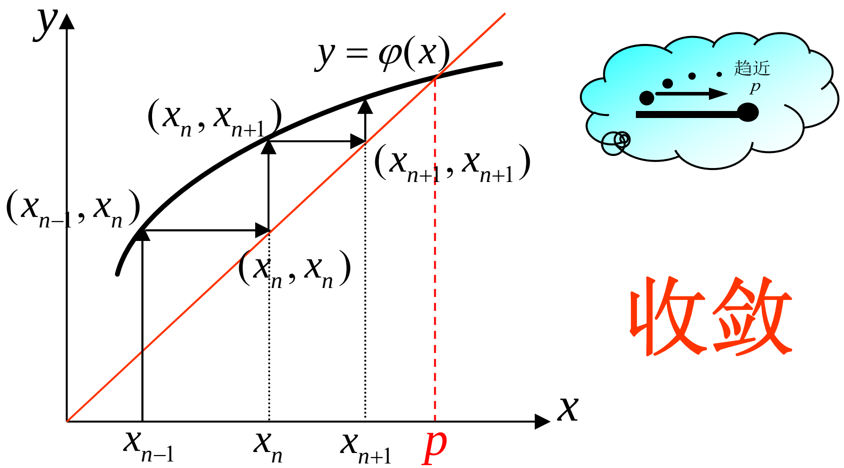




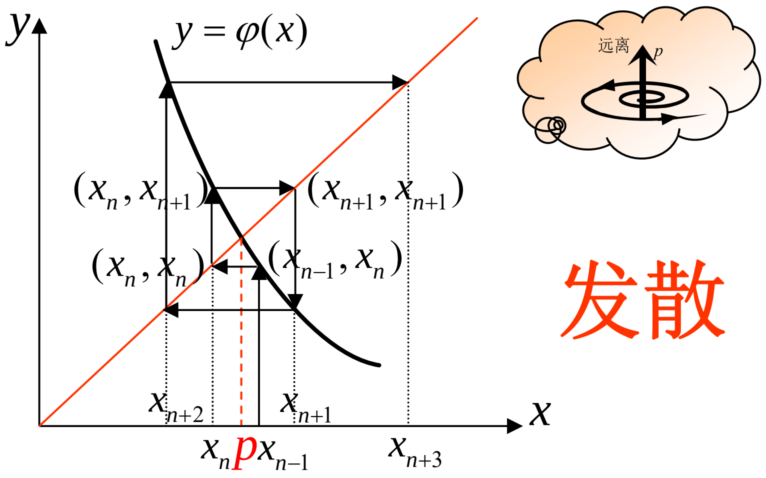
迭代法的几何意义



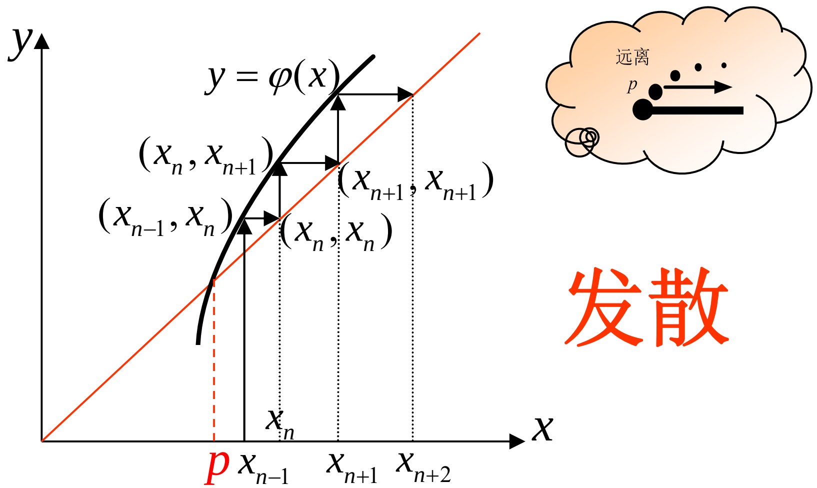
1) $-1 < \varphi'(x) < 0$



2) $0 < \varphi'(x) < 1$



3) $\varphi'(x) < -1$



4) $\varphi'(x) > 1$

定理2.5 压缩映射定理

❖ 压缩映射——不动点存在且唯一 ⟶ 收敛到域内唯一解

考虑方程 $x=\varphi(x)$，$\varphi(x)\in C[a,b]$，若

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

(Ⅱ) $\exists\ 0\le L<1$ 使得 $|\varphi'(x)|\le L<1$ 对 $\forall\ x\in[a, b]$ 成立。

则任取 $x_0\in[a, b]$，由 $x_n+1=\varphi(x_n)$ 得到的数列 $\{x_n\}$ 收敛于 $\varphi(x)$ 在 $[a, b]$ 上的唯一解。

注：定理条件非必要条件，可将 $[a, b]$ 缩小，定义**局部收敛性**：若在 p 的某 δ 领域 $B_\delta=\{x\,|\,|x-p|\le\delta\}$ 有 $\varphi\in C^1[a, b]$ 且 $|\varphi'(p)|<1$，则由 $\forall x_0\in B_\delta$ 开始的迭代收敛。即调整初值可得到收敛的结果。

迭代收敛速度的比较

❖ 对于两种不同的迭代格式，从相同的初值x_{n-1}开始迭代，收敛到同一个不动点p的速度

会不同吗？

分析：**定理2.6**

$$|\varphi'(x)| \le L\ ,\qquad \forall x \in [a,b]$$

若 φ是区间[a, b]上的压缩映射，那么迭代过程中解的误差满足：

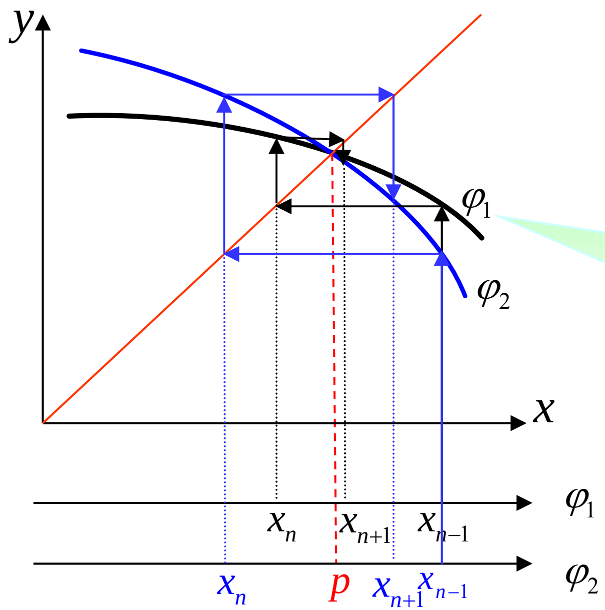
*L*越小收敛越快

$$|x_n - p| \le L^n \max\{x_0 - a, b - x_0\}$$

和 $$|x_n - p| \le \frac{L}{1-L}|x_n - x_{n-1}| \le \frac{L^n}{1-L}|x_1 - x_0|$$

其中p为区间[a, b]上的不动点，L ($0<L<1$)为φ在区间[a, b]上的压缩系数，$n=1,2,\ldots$

❖ 对于两种不同的迭代格式，从相同的初值x_{n-1}开始迭代，收敛到同一个不动点p的速度



$\varphi_1(x)$ 的收敛速度更快

不动点迭代法的计算例子

• 例2.2 设计多种迭代格式，使用不动点迭代法求解方程

$$7x^5-13x^4-21x^3-12x^2+58x+3=0\text{，}\quad x\in[1,2]$$

解：把要求解的方程写成迭代形式，以下给出 5 种不同的迭代形式：

（1） $x=\varphi_1(x)=7x^5-13x^4-21x^3-12x^2+59x+3$

（2） $x=\varphi_2(x)=\left(\dfrac{13x^4+21x^3+12x^2-58x-3}{7}\right)^{\frac{1}{5}}$

（3） $x=\varphi_3(x)=\dfrac{13+\dfrac{21}{x}+\dfrac{12}{x^2}-\dfrac{58}{x^3}-\dfrac{3}{x^4}}{7}$

前三种形式都
不满足
压缩映射要求

不动点迭代法的计算例子

• 例2.2 设计多种迭代格式，使用不动点迭代法求解方程

$$7x^5-13x^4-21x^3-12x^2+58x+3=0\,,\quad x\in[1,2]$$

$$（4）\quad x=\varphi_4(x)=\left(\frac{12x^2-58x-3}{7x^2-13x-21}\right)^{\frac{1}{3}}$$

这两种格式满足区间[1, 2]压缩映射要求

$$（5）\quad x=\varphi_5(x)=\left(\frac{-58x-3}{7x^3-13x^2-21x-12}\right)^{\frac{1}{2}}$$

不动点迭代法的计算例子

- 例2.2 设计多种迭代格式，使用不动点迭代法求解方程

$$7x^5-13x^4-21x^3-12x^2+58x+3=0\text{，}\quad x\in[1,2]$$

取初值 x_0=1.5

列出迭代过程

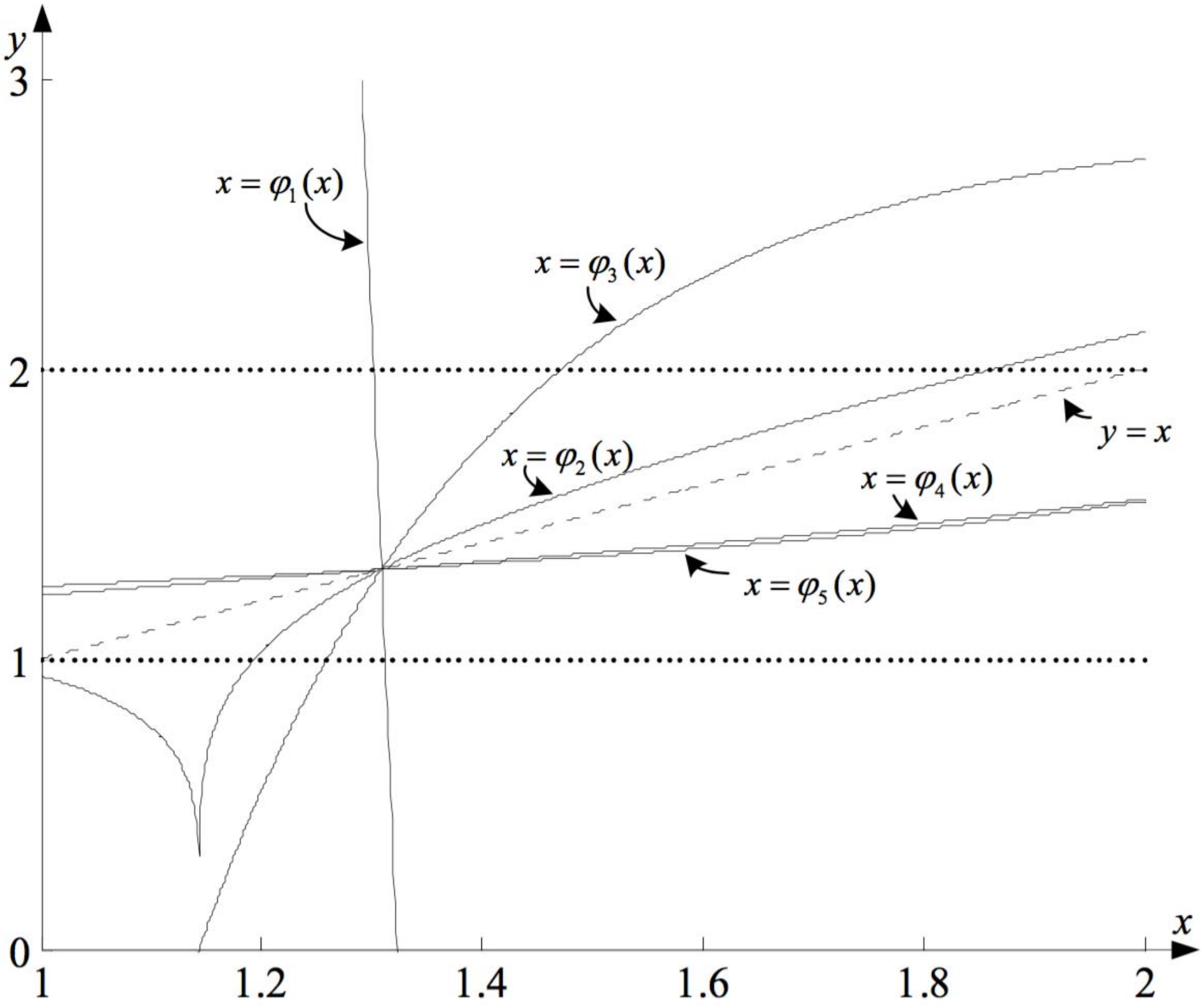
停止条件设为

$$|x_n-x_{n-1}|<\varepsilon=10^{-5}$$

得到满足精度的解为1.30893

n	(1)	(2)	(3)	(4)	(5)
0	1.5	1.5	1.5	1.5	1.5
1	-19.0313	1.60125	2.07937	1.36538	1.35354
2	-1.9e+007	1.7239	2.75186	1.32528	1.31852
3	-1.8e+037	1.85819	2.76861	1.31364	1.31095
4	-1.2e+187	1.9935	2.76664	1.31028	1.30935
5	…	2.12139	2.76687	1.30932	1.30901
6		2.23655	2.76685	1.30904	1.30894
7		2.33657	2.76685	1.30896	1.30893
8		2.42114		1.30893	1.30893
9		2.49122		1.30893	
10		…			

不动点迭代法的计算例子



本例中 5 种迭代格式的函数图示

收敛速度讨论

❖ 收敛阶

定义2.3 对于一个收敛到 $\boldsymbol{p}$ 的数列 $\{x_n\}$，$x_n \neq p$，如果存在实数 λ 和 α 使得

$$\lim_{n\to\infty}\frac{|x_{n+1}-p|}{|x_n-p|^{\alpha}}=\lambda$$

当 $\lambda \neq 0$ 时称数列的收敛阶是 α，λ 为渐近误差常数。

(1)若 $\alpha = 1$，称数列线性收敛或 1阶收敛。

(2)若 $\alpha = 2$，称数列平方收敛或 2阶收敛。

(3)若 $\alpha > 1$，称数列超线性收敛或 α 阶收敛。

又当 $\lambda \neq 0$ 时称数列 $\{x_n\}$ 的收敛阶高于 α。

收敛速度讨论

❖ 收敛阶

定理2.6

满足压缩映射的迭代函数φ得到的数列$\{x_n\}$

当$\varphi'(p)\neq 0$都线性收敛到$[a,b]$上唯一的不动点p

当$\varphi'(p)=0$且$\varphi''(p)\neq 0$时二阶收敛到$[a,b]$上唯一的不动点p

收敛阶的计算例子

• 例2.3 假定如下迭代格式在区间内收敛，试用定理2.6分析以下函数数列 $\{x_n\}$ 的收敛阶

（1） $\varphi_1(x)=\dfrac{14}{x+1}$

（2） $\varphi_2(x)=x-\dfrac{x^2+x-14}{2x+1}$

解：以上函数实际上都等价于方程 $f(x)=x^2+x-14=0$，解之得根 $p=\dfrac{-1\pm\sqrt{57}}{2}$。

对（1）求导得 $\varphi_1'(x)=\dfrac{-14}{(x+1)^2}<0$，即 $\varphi_1'(p)\neq 0$，因此函数数列 $\{x_n\}$ 的收敛阶是 1。

对（2）求得 $\varphi_2'(x)=\dfrac{2(x^2+x-14)}{(2x+1)^2}$，$\varphi_2'(p)=0$，而 $\varphi_2''(x)=\dfrac{114}{(2x+1)^3}$，$\varphi_2''(p)\neq 0$，因此函数数列 $\{x_n\}$ 的的收敛阶是 2。