

回顾

1.1. 微积分回顾

1.2. 二进制

1.3. 误差分析

概念

绝对误差和相对误差

有效数字

减少运算误差的原则

- 要避免相近两数相减
- 要防止“大数吃掉小数”
- 绝对值太小的数不宜做除数
- 简化计算步骤，减少运算次数
- 控制递推公式中误差的传播

第二章 非线性方程 $f(x) = 0$ 的解法

2.1 引言

2.2 二分法与试值法

2.3 不动点迭代法

2.4 牛顿-拉夫森法（简称：牛顿迭代法）

2.5 割线法

2.6 迭代收敛的加速办法（选讲）

计算机科学中的一个基本要素是迭代 (Iteration)
正如名字所表示的含义，迭代是指重复执行一个
计算过程，直到找到答案。

本章主要研究重复替换的迭代处理过程。

§ 2.1 引言

在科学和工程问题中，经常会遇到的一大类问题是非线性方程：

$$f(x) = 0$$

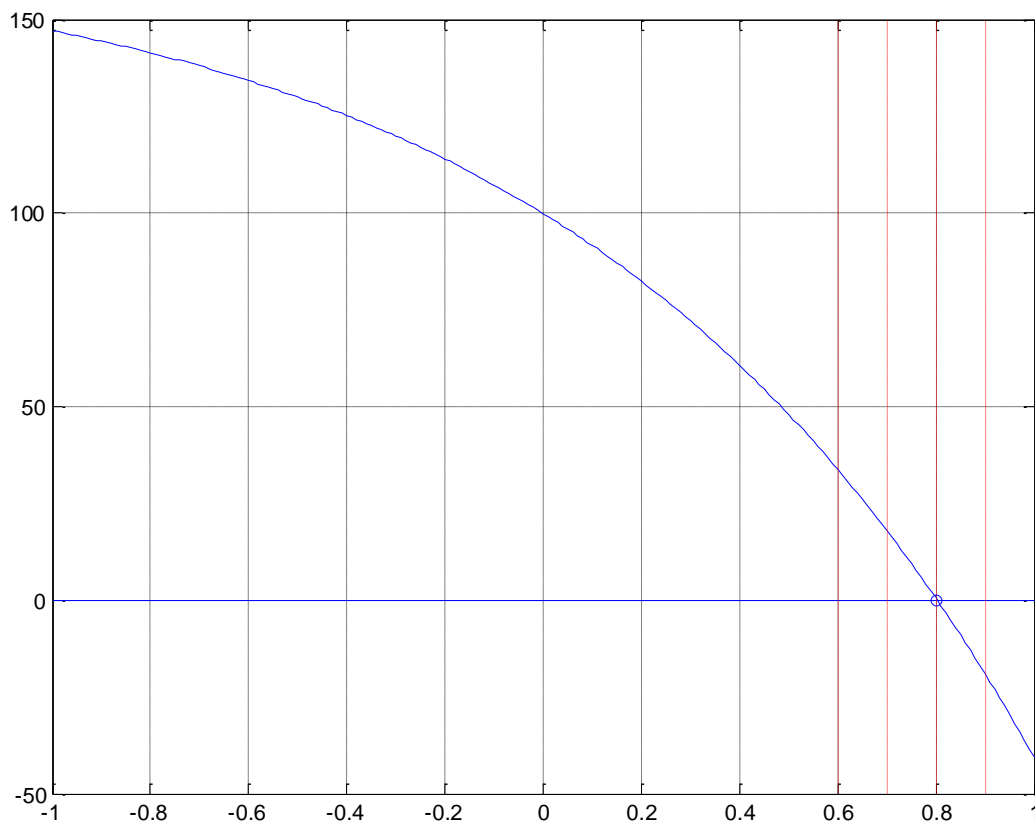
的求根问题，其中 $f(x)$ 为非线性函数。方程 $f(x)=0$ 的根，亦称为函数 $f(x)$ 的零点。

代数方程的求根问题是一个古老的数学问题。理论上， n 次代数方程在复数域内一定有 n 个根(考虑重数)。早在16世纪就找到了三次、四次方程的求根公式，但直到19世纪才证明大于等于5次的一般代数方程式不能用代数公式求解，而对于超越方程就复杂的多，如果有解，其解可能是一个或几个，也可能是无穷多个。一般也不存在根的解析表达式。

因此研究如何用数值方法求得满足一定精度要求的根的近似解具有重要的现实意义。

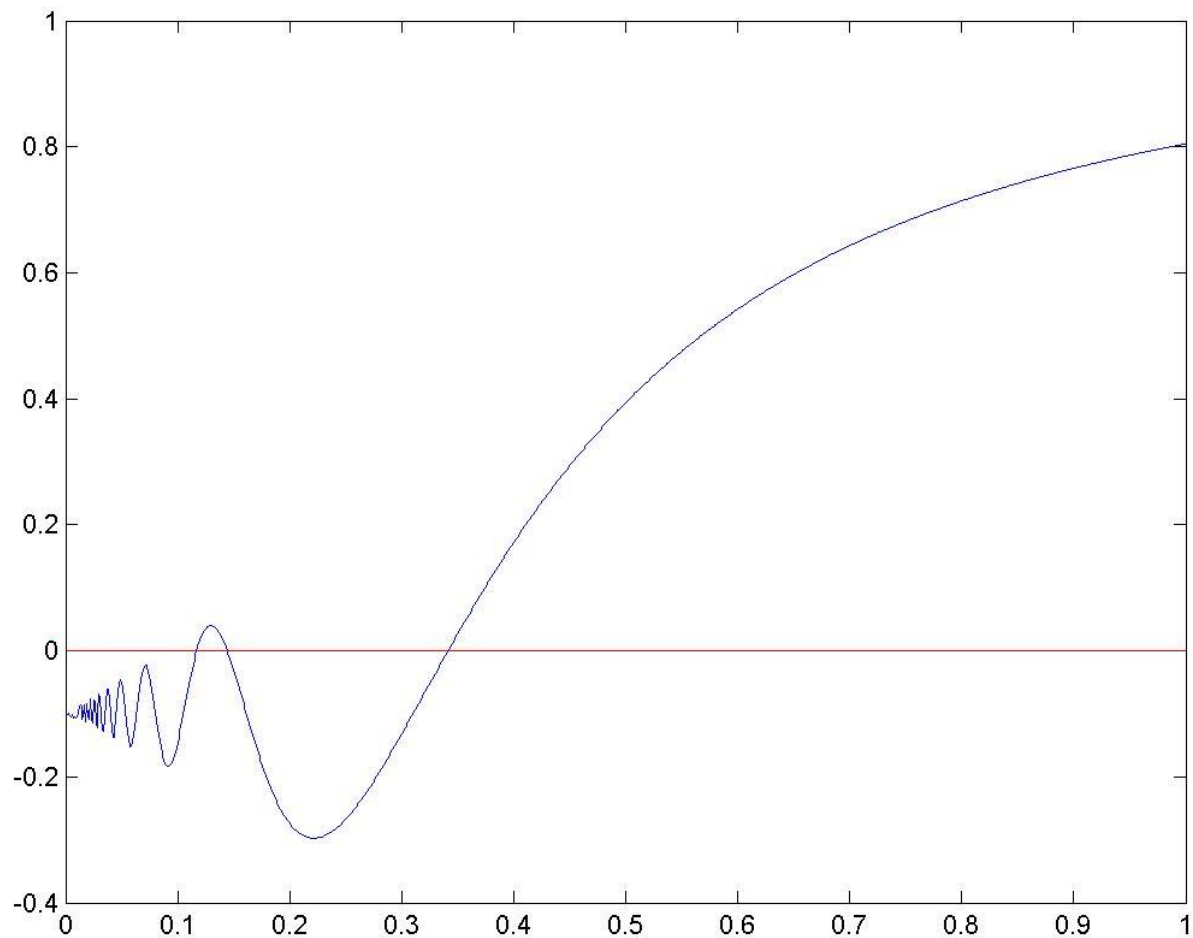
例2.1

$$y = f(x) = 156.4 - 100e^x + \frac{43.5}{x}(e^x - 1) = 0$$

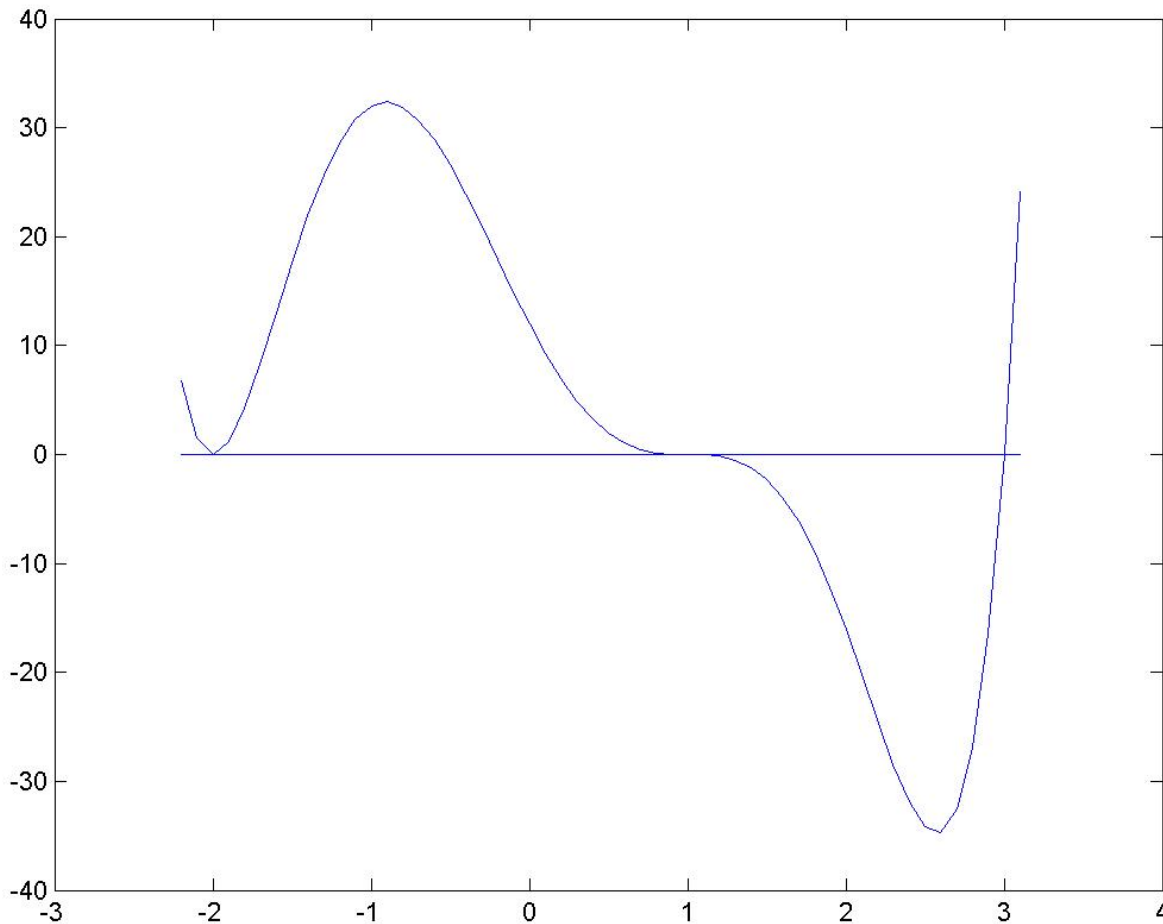


例2.2

$$f(x) = x \sin(1/x) - 0.1e^{-x}$$



例2.3 $f(x) = x^6 - 2x^5 - 8x^4 + 14x^3 + 11x^2 - 28x + 12$
 $= (x - 3)(x + 2)^2(x - 1)^3$



```
>> clear all  
>> a=[1,-2,-8,14,11,-  
28,12];  
>> x=-2.2:0.1:3.1;  
>> y=polyval(a,x);  
>> plot(x,y)  
>> hold on  
>> plot([-2.2 3.1],[0 0])
```

如何用数值方法求得非线性方程的根

通常大致分为三个步骤进行：

- ① **判定根的存在性**。即方程有没有根？如果有根，有几个根？
- ② **确定根的分布范围**。即将每一个根用区间隔离开来，这个过程实际上是获得方程各根的初始近似值。
- ③ **根的精确化**。将根的初始近似值按某种方法逐步精确化，直到满足预先要求的精度为止。

本章研究内容：

有根的前提下求出方程的近似根。

§ 2.2 二分法与试值法

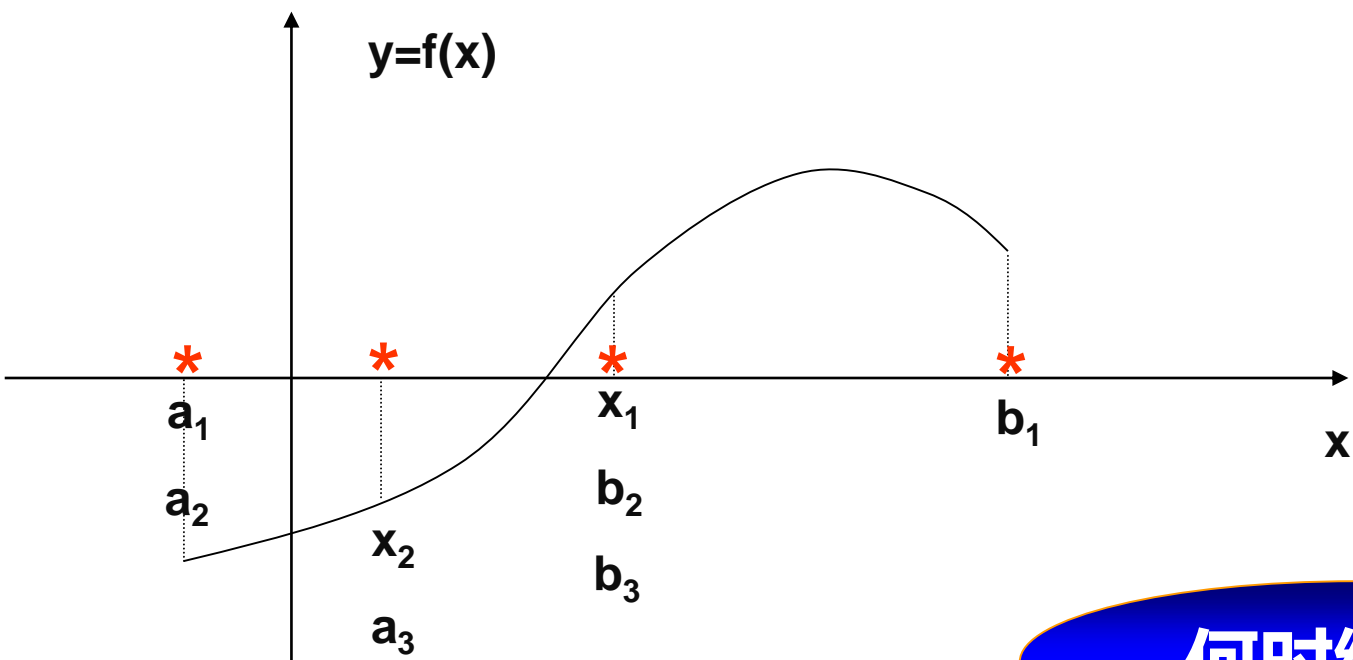
(1) 二分法

二分法又称二分区间法，是求解非线性方程近似根的一种常用的简单方法。

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。

□ 具体方法：

通过二等分不断缩小有根区间的长度，直到满足精度为止。



何时终止?

$$|x_{k+1} - x_k| < \varepsilon \text{ 或 } |f(x_k)| < \varepsilon$$

不能保证 x 的精度

算法(二分法)

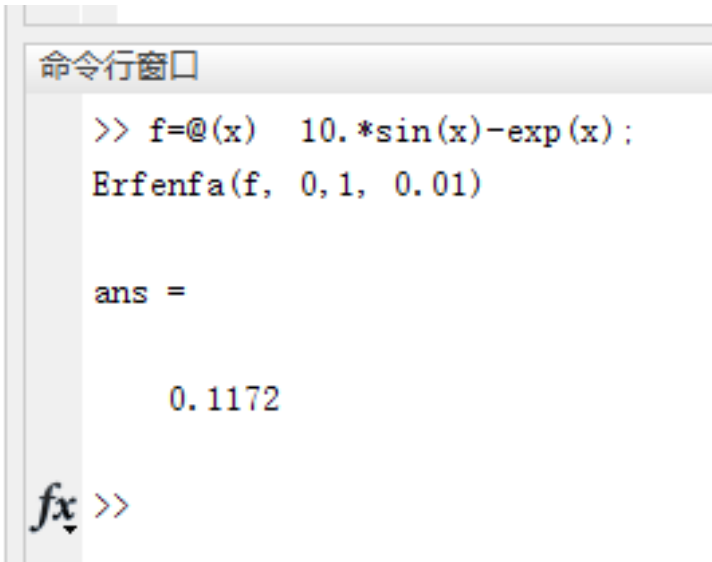
用二分法求根，通常先给出 $f(x)$ 草图以确定根的大概位置。

给定有根区间 $[a, b]$ ($f(a) \cdot f(b) < 0$) 和精度要求 ε

1. 令 $x = (a+b)/2$
2. 如果 $b - a \leq 2\varepsilon$, 停止计算, 输出 x , 否则执行第3步
3. 如果 $f(a) f(x) < 0$, 则令 $b = x$, 否则令 $a = x$, 返回第1步

Matlab源程序: Erfenfa.m

Erfa.m

A screenshot of the MATLAB Command Window. The title bar says '命令行窗口'. The command prompt is '>>'. The user has entered 'f=@(x) 10.*sin(x)-exp(x);' and 'Erfa(f, 0, 1, 0.01)'. The output shows 'ans =' followed by a blank line and then '0.1172'. At the bottom, there is a cursor icon and '>>'.

```
命令行窗口

>> f=@(x) 10.*sin(x)-exp(x);
Erfa(f, 0, 1, 0.01)

ans =

    0.1172

fx >>
```

```
function x =Erfa(fname, a, b, e)
```

```
% f=@(x) 10.*sin(x)-exp(x);
```

```
% Erfa(f, 0,1, 0.01)
```

```
if nargin<4, %nargin用来判断输入变量个数的函数
```

```
e = 1e-4;
```

```
end;
```

```
fa = feval(fname, a); fb = feval(fname, b);
```

```
if fa * fb > 0,
```

```
error('函数在两端点的值必须异号');
```

```
end
```

```
x = (a+b)/2;
```

```
while (b-a)>(2*e)
```

```
    fx = feval(fname,x);
```

```
if fa * fx <0,
```

```
    b = x; fb = fx;
```

```
else a = x; fa = fx;
```

```
end
```

```
    x = (a+b)/2;
```

```
end
```

误差分析

记 $a_0 = a, b_0 = b$, 第 k 步的有根区间为 $[a_k, b_k]$

$$\begin{aligned} |x_k - x^*| &= \left| \frac{b_k + a_k}{2} - x^* \right| \leq \frac{b_k - a_k}{2} = \frac{b_{k-1} - a_{k-1}}{2^2} = \dots \\ &= \frac{b_0 - a_0}{2^{k+1}} = \frac{b - a}{2^{k+1}} \end{aligned}$$

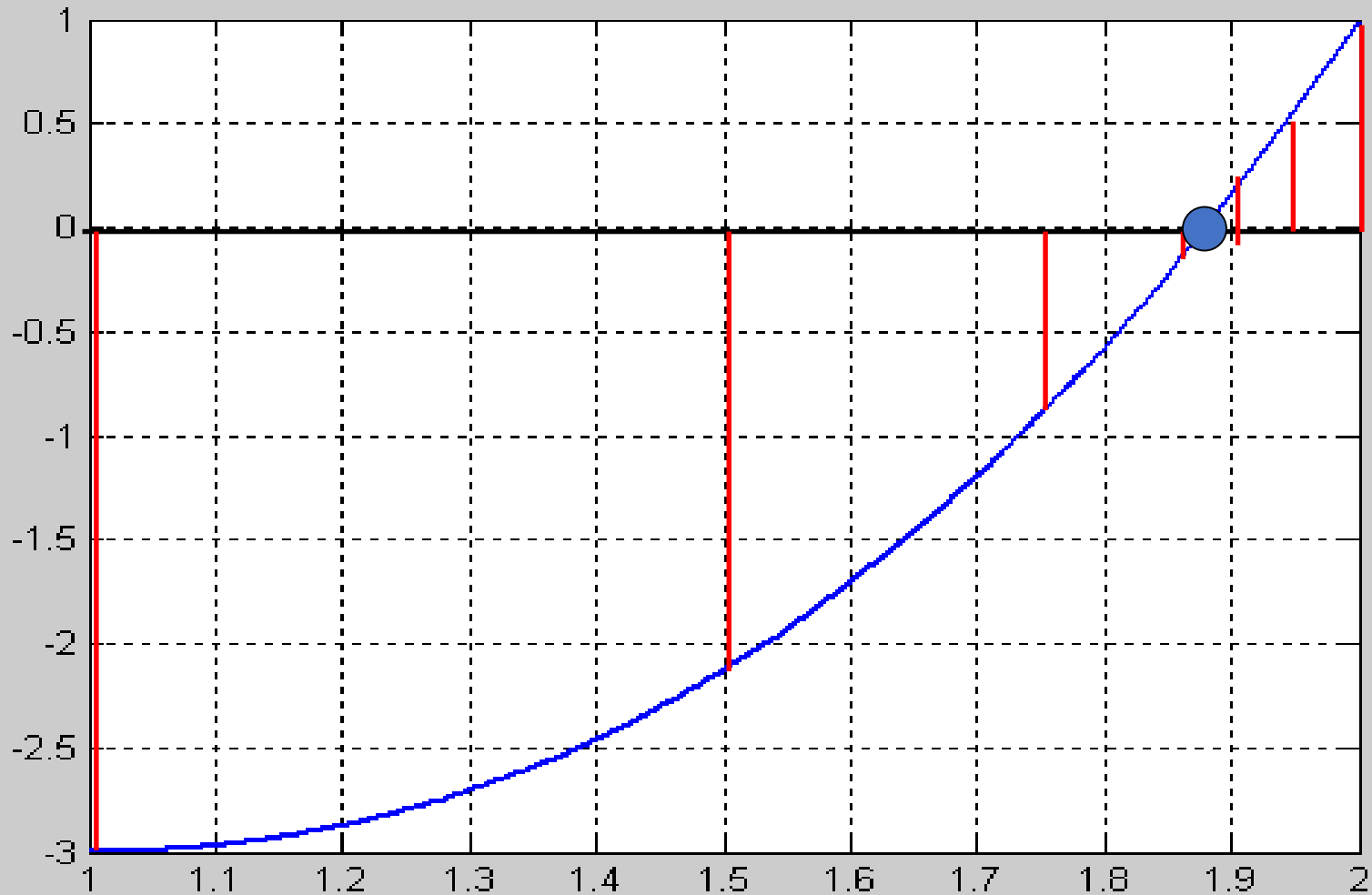
对于给定的精度 ε , 可估计二分法所需的步数 k :

$$\frac{b - a}{2^{k+1}} \leq \varepsilon \Rightarrow k \geq \log_2 \frac{b - a}{\varepsilon} - 1,$$

取 $k = \left\lceil \log_2 \frac{b - a}{\varepsilon} \right\rceil - 1$

例2.4 求 $x^3 - 3x - 1 = 0$ 在 $[1, 2]$ 内的根

两位有效数字 $\varepsilon = 0.05$, $k \geq \frac{\ln 20}{\ln 2} - 1$, 取 $k=4$





优点

- ① 简单易用;
- ② 对 $f(x)$ 要求不高(只要连续即可) .



缺点

- ① 无法求复根及偶重根
- ② 收敛慢

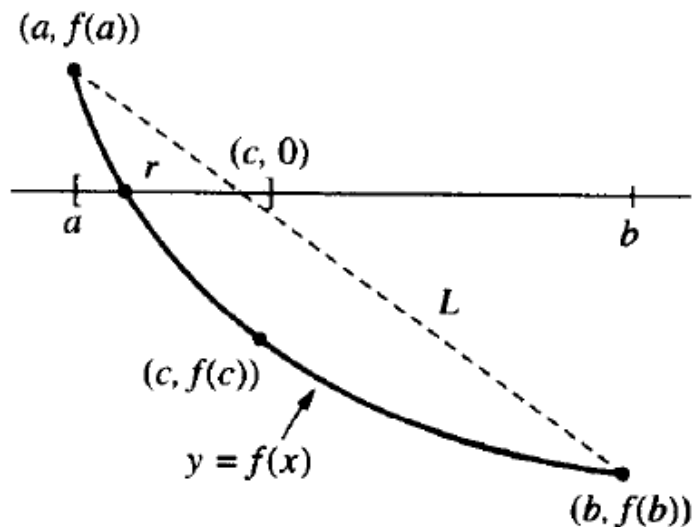
注：用二分法求根，最好先给出 $f(x)$ 草图以确定根的大概位置。

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 $f(a) \cdot f(b) < 0$ 。

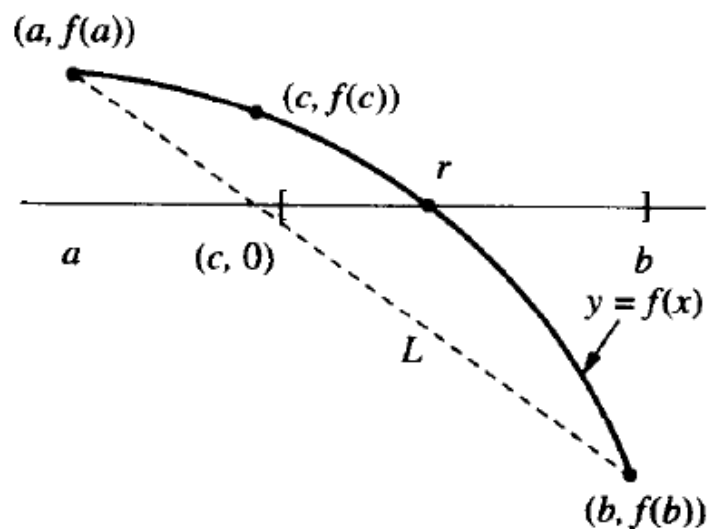
(2) 试值法 (又称试位法)

由于二分法收敛速度相对较慢, 因此试值法对他进行了改进。

与上述条件一样, 假设 $f(a)$ 和 $f(b)$ 符号相反。而试值法则考虑经过点 $(a, f(a))$ 和 $(b, f(b))$ 的割线 L 与 x 轴的交点 $(c, 0)$, 以得到一个更好的近似值。



(a) 如果 $f(a)$ 和 $f(c)$ 符号相反, 则从右边压缩



(b) 如果 $f(c)$ 和 $f(b)$ 符号相反, 则从左边压缩

图 2.8 试值法的判定过程

回顾切线、直线的点斜式、两点式。

经过点(a, f(a))和(b, f(b))的直线表达式

$$y - f(b) = \frac{f(b) - f(a)}{b - a} (x - b)$$

令y=0, 则有

$$x = b - \frac{f(b)(b-a)}{f(b) - f(a)} \quad \underline{\underline{=}} \quad c$$

- ✓ 如果f(a)和f(c)的符号相反, 则在[a,c]内有一个零点;
- ✓ 如果f(b)和f(c)的符号相反, 则在[c,b]内有一个零点;
- ✓ 如果f(c)=0, 则c是零点。

(2) 试值法的收敛性

由上可知，可构造 $\{[a_n, b_n]\}$ 区间集合，其中的每个序列包含零点。在每一步中，零点 r 的近似值为

$$c_n = b_n - \frac{f(b_n)(b_n - a_n)}{f(b_n) - f(a_n)} \quad \text{且} \quad c_n \rightarrow r$$

注意，尽管区间宽度 $b_n - a_n$ 越来越小，但它可能不趋近于零。

Matlab源程序： [Shizhifa.m](#)

(2) 试值法的收敛性

```
function [c,err,yc]=Shizhifa(f,a,b,delta,epsilon,max1)
```

```
%Input   - f is the function
%         - a and b are the left and right endpoints
%         - delta is the tolerance for the zero
%         - epsilon is the tolerance for the value of f at
the zero
%         - max1 is the maximum number of iterations
%Output - c is the zero
%         - yc=f(c)
%         - err is the error estimate for c
%Example
% f=@(x) 10.*sin(x)-exp(x);
% Shizhifa(f, 0, 1, 0.01, 0.01, 30)
ya=f(a);
yb=f(b);

if ya*yb>0
    disp('Note: f(a)*f(b) >0'),
    return,
end
```

```
for k=1:max1
    dx=yb*(b-a)/(yb-ya);
    c=b-dx;
    ac=c-a;
    yc=f(c);
    if yc==0,break;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    dx=min(abs(dx),ac);
    if abs(dx)<delta,break,end
    if abs(yc)<epsilon, break,end
end

c;
err=abs(b-a)/2;
yc=f(c);
```

作业 2.1

(1) 分别运用二分法和试值法求解函数

$f(x) = x^2 - 10x + 23$ 在区间 $[6, 7]$ 的零点。

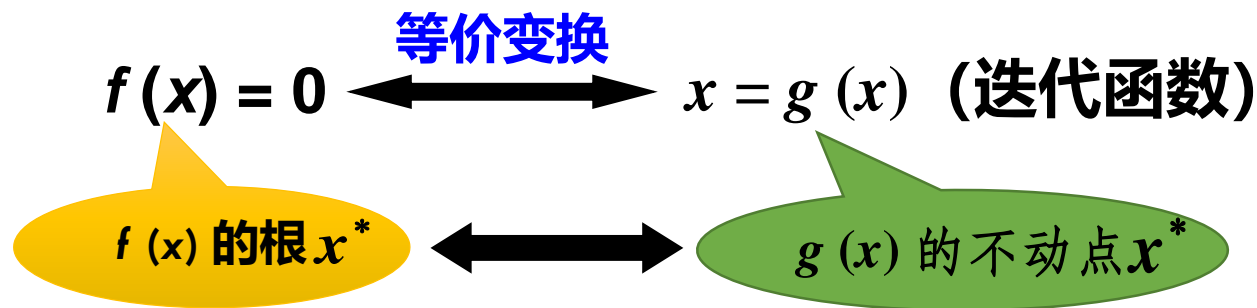
§ 2.3 不动点迭代法

计算机科学中的一个基本要素就是迭代 (iteration). 正如其名字所表示的含义, 迭代是指一种重复执行一个计算过程, 逐次逼近答案的方法。这种方法使用某个固定公式(迭代公式)反复校正根的近似值, 使之逐步精确化, 最后得到满足精度要求的结果。

它是解代数方程、超越方程、微分方程等的一种基本而重要的数值方法。

- 2.3.1 不动点迭代法的基本思想
- 2.3.2 不动点迭代法的几何解释
- 2.3.3 不动点迭代法的收敛性分析
- 2.3.4 不动点迭代法的算法实现

§ 2.3.1 不动点迭代法的基本思想



$$x_{k+1} = g(x_k) \quad k = 0, 1, 2, \dots \quad (*)$$

思路

从一个初值 x_0 出发, 计算 $x_1 = g(x_0)$, $x_2 = g(x_1)$, \dots , $x_{k+1} = g(x_k)$, \dots 若 $\{x_k\}_{k=0}^{\infty}$ 收敛, 即存在 x^* 使得 $\lim_{k \rightarrow \infty} x_k = x^*$, 且 g 连续, 则由 $\lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} g(x_k)$ 可知 $x^* = g(x^*)$, 即 x^* 是 g 的不动点, 也就是 f 的根。

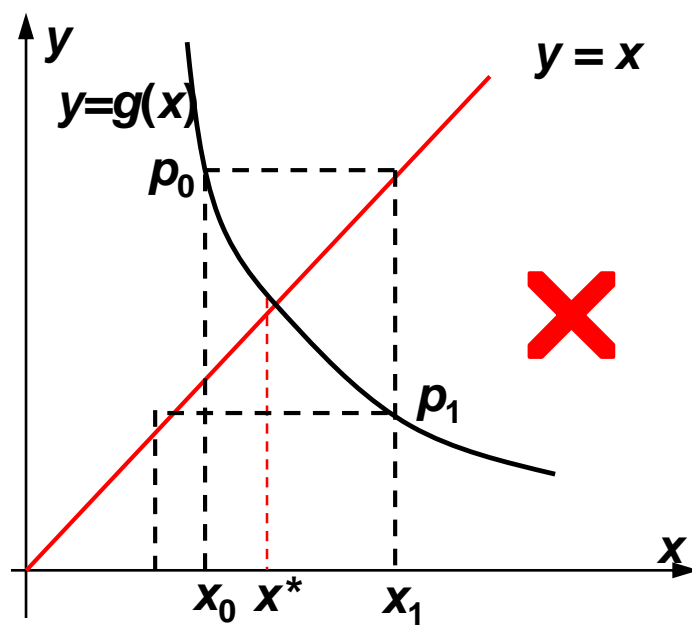
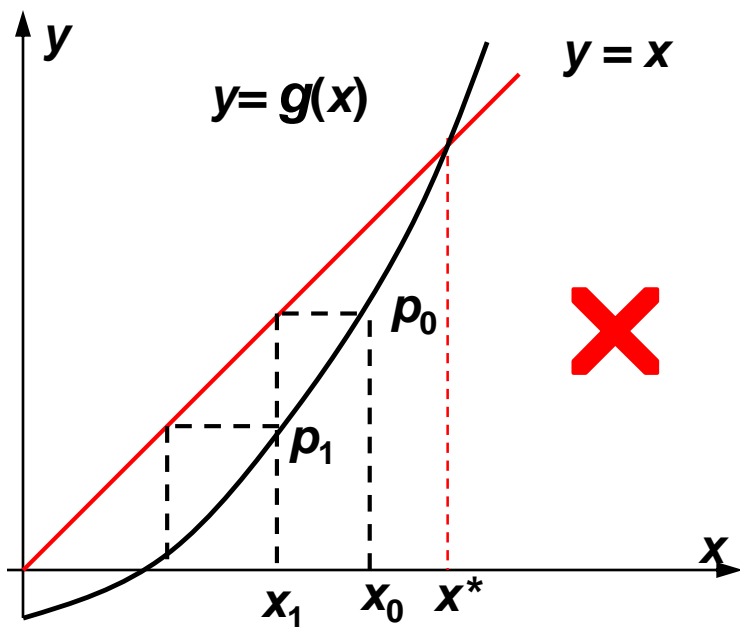
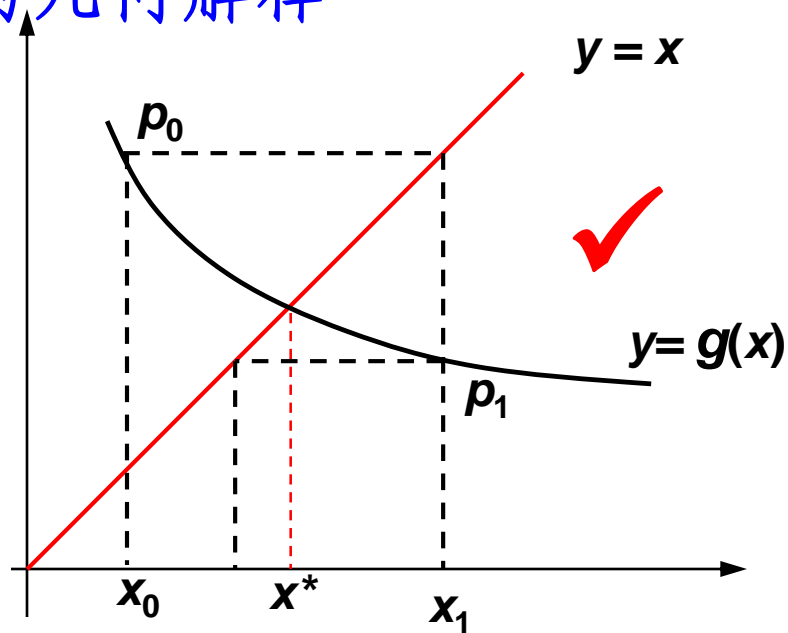
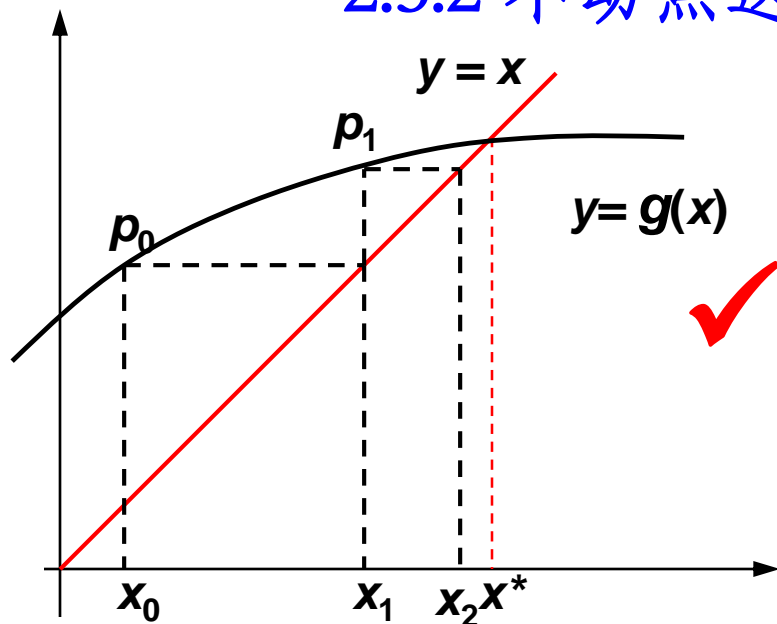


看起来很简单,
令人有点不相信,
那么问题是什么呢?

如何判定这种方法是收敛的呢?



2.3.2 不动点迭代法的几何解释



例 2.5. $x^3 - x - 1 = 0$, $[1, 2]$, 取 $x_0 = 1.5$

- 迭代公式1: $x_{k+1} = x_k^3 - 1$
- 迭代公式2: $x_{k+1} = (x_k + 1)^{\frac{1}{3}}$

• 计算结果:

公式1:

k	x_k
0	1.5
1	2.375
2	12.4
3	1904

公式2:

k	x_k
0	1.5
1	1.35721
2	1.33086
3	1.32588
4	1.32494
5	1.32476
6	1.32473
7	1.32472

精确解 $x^* =$
1.3247179...

怎么判断迭代公式收敛或发散呢?

例2.6：已知方程 $x^3 + 4x^2 - 10 = 0$ 在 $[1, 2]$ 上有一个根

(正根) 下面选取5种迭代格式：

1、 $x = x - x^3 - 4x^2 + 10$ **即** $g(x) = x - x^3 - 4x^2 + 10$

2、 $4x^2 = 10 - x^3$ $x = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$ **即** $g(x) = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$

3、 $x^2 = \frac{10}{x} - 4x$ $x = \left(\frac{10}{x} - 4x\right)^{\frac{1}{2}}$ **即** $g(x) = \left(\frac{10}{x} - 4x\right)^{\frac{1}{2}}$

4、 $x = \left(\frac{10}{4 + x}\right)^{\frac{1}{2}}$ **即** $g(x) = \left(\frac{10}{4 + x}\right)^{\frac{1}{2}}$

5、 $x = x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x}$ **即** $g(x) = x - \frac{f(x)}{f'(x)}$

取 $x_0 = 1.5$ 计算结果如下:

法1

$$x_1 = -0.875$$

$$x_2 = 6.732$$

$$x_3 = -469.720$$

$$x_4 = 1.0275 \times 10^8$$

法5

$$x_1 = 1.37333$$

$$x_2 = 1.36526$$

$$x_3 = 1.365230014$$

$$x_4 = 1.365230013$$

法2

$$x_1 = 1.28695$$

$$x_2 = 1.40254$$

$$x_3 = 1.34546$$

$$x_4 = 1.37517$$

$$x_4 = 1.37517$$

$$x_5 = 1.37517$$

...

$$x_{11} = 1.365137821$$

...

$$x_{29} = 1.365230013$$

法3

$$x_1 = 0.81650$$

$$x_2 = 2.99691$$

$$x_3 = (-8.65086)^{\frac{1}{2}}$$

法4

$$x_1 = 1.34840$$

$$x_2 = 1.36738$$

$$x_3 = 1.36496$$

$$x_4 = 1.36526$$

$$x_4 = 1.37517$$

$$x_5 = 1.365225$$

...

$$x_{11} = 1.365230013$$

回顾

第二章 非线性方程 $f(x) = 0$ 的解法

2.1 引言

2.2 二分法与试值法

2.3 不动点迭代法

2.4 牛顿-拉夫森法（简称：牛顿迭代法）

2.5 割线法

2.6 迭代收敛的加速办法(选讲)

2.3.3 不动点迭代法的收敛性分析

不动点迭代法的收敛条件

由于对方程 $f(x)=0$ 可以构造不同的迭代公式,但迭代公式:

$$\mathbf{x}_{k+1}=\mathbf{g}(\mathbf{x}_k), \quad \mathbf{k}=0, 1, 2\ldots$$

并非总是收敛的。那么,当迭代函数 $\varphi(x)$ 满足什么条件时,相应的迭代公式才收敛呢?

即使迭代收敛时,也不可能迭代很多次,而是迭代有限次后就停止,这就需要估计迭代值的误差,以便适时终止迭代。

不动点原理 (压缩映像定理)

定理2.1 设 $g(x)$ 在 $[a, b]$ 上连续, 且一阶导数连续, 若

(1) $a \leq g(x) \leq b$ 对一切 $x \in [a, b]$ 都成立,

(2) 存在 $0 \leq L < 1$, 使得 $|g'(x)| \leq L$ 对 $\forall x \in [a, b]$ 成立,

则函数 $f(x) = x - g(x)$ 在 $[a, b]$ 中有唯一的零点 x^* 。

x^* 称为 $g(x)$ 的不动点, 即 $g(x^*) = x^*$

简证: 取 $f(x) = x - g(x)$, 我们有

$$f(a) = a - g(a) \leq 0, f(b) = b - g(b) \geq 0.$$

➡ $f(x)$ 在 $[a, b]$ 上有零点。

唯一性: 反证法, 假设存在 $x^*, y^* \in [a, b]$ 使得

$$\begin{cases} x^* = g(x^*) \\ y^* = g(y^*) \end{cases}$$

➡ $|x^* - y^*| = |g(x^*) - g(y^*)| = |g'(\xi)| \cdot |x^* - y^*| \leq L|x^* - y^*|$ 矛盾!

收敛性分析

定理2.2 设 $g(x)$ 在 $[a, b]$ 上连续, 且一阶导数连续, 若

(1) $a \leq g(x) \leq b$ 对一切 $x \in [a, b]$ 都成立,

(2) 存在 $0 \leq L < 1$, 使得 $|g'(x)| \leq L$ 对 $\forall x \in [a, b]$ 成立,


则


(a) 迭代 $x_{k+1} = g(x_k)$ 将收敛到 $g(x)$ 在 $[a, b]$ 中的唯一不动点 x^* 。

证明:


(a) 由压缩映像定理可知, 不动点 x^* 存在且唯一。


$$|x_k - x^*| = |g(x_{k-1}) - g(x^*)| = |g'(\xi)| \cdot |x_{k-1} - x^*| \leq L |x_{k-1} - x^*|$$


$$|x_k - x^*| \leq L |x_{k-1} - x^*| \leq L^2 |x_{k-2} - x^*| \leq \dots \leq L^k |x_0 - x^*|$$


$$\lim_{k \rightarrow \infty} |x_k - x^*| = 0 \quad \text{即} \quad \lim_{k \rightarrow \infty} x_k = x^*.$$

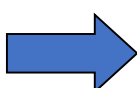
(b) 有如下的误差估计:

后验估计: $|x_k - x^*| \leq \frac{L}{1-L} |x_k - x_{k-1}|$  可用 $|x_k - x_{k-1}|$ 来控制收敛精度


先验估计 $|x_k - x^*| \leq \frac{L^k}{1-L} |x_1 - x_0|$  L 越小收敛越快

证明: 由于 $|x_{k+1} - x^*| \leq L |x_k - x^*|$

可得 $|x_{k+1} - x_k| = |(x_{k+1} - x^*) - (x_k - x^*)| \geq |x_k - x^*| - |x_{k+1} - x^*|$

 $|x_k - x^*| \leq \frac{1}{1-L} |x_{k+1} - x_k| \quad \geq (1-L) |x_k - x^*|$

又 $|x_{k+1} - x_k| = |g(x_k) - g(x_{k-1})| = |g'(\xi)| \cdot |x_k - x_{k-1}| \leq L |x_k - x_{k-1}|$

 $|x_k - x^*| \leq \frac{1}{1-L} |x_{k+1} - x_k| \leq \frac{L}{1-L} |x_k - x_{k-1}| \leq \dots \leq \frac{L^k}{1-L} |x_1 - x_0|$

例2.7 用不同方法求 $x^2-3=0$ 的根 $\sqrt{3}$, 取 $x_0=2$.
讨论合理性和收敛性

- 迭代公式1: $x_{k+1} = x_k^2 + x_k - 3$
- 迭代公式2: $x_{k+1} = 3 / x_k$
- 迭代公式3: $x_{k+1} = x_k - (x_k^2 - 3) / 4$
- 迭代公式4: $x_{k+1} = (x_k + 3 / x_k) / 2$
- 计算结果:

精确值:

$\sqrt{3} = 1.7320508...$

k	x_k	方法1	方法2	方法3	方法4
0	x_0	2	2	2	2
1	x_1	3	1.5	1.75	1.75
2	x_2	9	2	1.734375	1.732143
3	x_3	87	1.5	1.732361	1.732051
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

怎么判断收敛的迭代公式的速度快慢呢?

收敛速度—收敛阶

定义2.1 设迭代 $x_{k+1} = g(x_k)$ 收敛到 $g(x)$ 的不动点 x^* 。记绝对误差 $e_k = x_k - x^*$ ，若

$$\lim_{k \rightarrow \infty} \left| \frac{e_{k+1}}{e_k^p} \right| = C \neq 0 \quad (C \text{ 为常数})$$

则称该迭代为以收敛阶 p 收敛到 x^* 。数 C 称为渐近误差常数。

(1) 当 $p=1$ 时称为线性收敛，此时 $|C| < 1$ ；

(2) 当 $p=2$ 时称为二次收敛，或平方收敛；

(3) 当 $p>1$ 时称为超线性收敛。

□ 不动点迭代中，若迭代数列 $\{x_k\}$ 收敛，且 $g'(x^*) \neq 0$ ，则

$$e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*) = g'(\xi)e_k$$

取极限得 $\lim_{k \rightarrow \infty} \left| \frac{e_{k+1}}{e_k} \right| = g'(x^*) \neq 0 \longrightarrow$ 线性收敛。

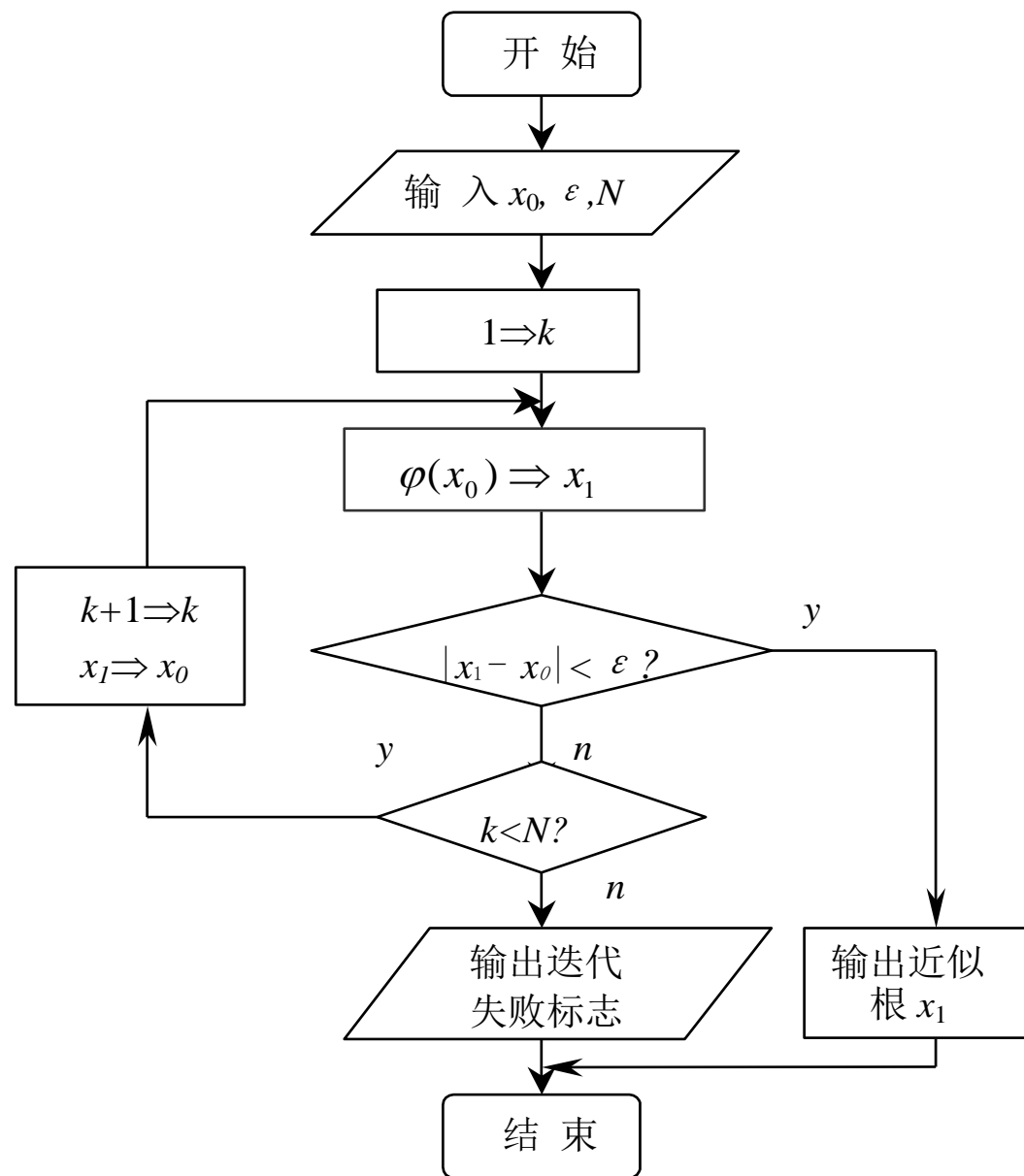
2.3.4 不动点迭代法的算法实现

迭代法就是通过有限次的计算，来求出给定方程的满足精度要求的近似根。它的突出优点是算法的逻辑结构简单，且在计算时，中间结果若有扰动，仍不会影响计算结果。

计算步骤

- 1) 确定有根区间的范围 $[a,b]$ ，使 $f(a)f(b)<0$ ；
- 2) 确定方程 $f(x)=0$ 的等价形式 $x=\varphi(x)$ 及初始值 x_0 ；为确保迭代收敛，要求 $\varphi(x)$ 满足定理2.1的条件；
- 3) 建立迭代格式 $x_{k+1}=\varphi(x_k)$ ，计算出 $x_{k+1}(K=0,1,2\cdots)$ ；
- 4) 若 $|x_{K+1}-x_K|<\varepsilon$ (ε 为事先给定的精度)，则终止迭代，输出 x_{k+1} 为 x 的近似值；否则，继续迭代。

算法流程图



程序：

fixed_iteration.m

图 迭代法的算法流程图

程序：

fixed_Iteration.m

```
function [k,p,err,P] = fixed_Iteration(g,p0,tol,max1)
%Input - g is the iteration function
%      - p0 is the initial guess for the fixed-point
%      - tol is the tolerance
%      - max1 is the maximum number of iterations
%Output- k is the number of iterations
%      - p is the approximation to the fixed-point
%      - err is the error in the approximation
%      - P' contains the sequence {pn}
% f=@(x) 2*sqrt(x-1);
% p0=1; tol=1e-4; max1=500;
% fixed_Iteration(f, p0, tol, max1)
P(1)= p0;

for k=2:max1
    P(k)=g(P(k-1));
    err=abs(P(k)-P(k-1));
    relerr=err/(abs(P(k))+eps);
    p=P(k);
    if (err<tol) | (relerr<tol), break; end
End

P=P'
```

作业 2.2

2. 当

$$g(x) = -4 + 4x - \frac{1}{2}x^2$$

时,研究不动点迭代的性质。

- (a) 求解 $g(x) = x$, 且证明 $P = 2$ 和 $P = 4$ 是不动点。
- (b) 用起始值 $p_0 = 1.9$ 计算 p_1, p_2 和 p_3 。
- (c) 用起始值 $p_0 = 3.8$ 计算 p_1, p_2 和 p_3 。
- (d) 对于(b)和(c)中的 p_k , 寻找误差 E_k 和相对误差 R_k 。
- (e) 从定理 2.3 中可得出什么结论?

2.4 牛顿-拉夫森法（简称：牛顿迭代法）

用迭代法可逐步精确方程 $f(x)=0$ 根的近似值，但必须要找到 $f(x)=0$ 的等价方程 $x=\varphi(x)$ ，如果 $\varphi(x)$ 选得不合适，不仅影响收敛速度，而且有可能造成迭代格式发散。

能否找到一种迭代方法，既结构简单，收敛速度快，又不存在发散的问题。这就是本节要介绍的牛顿迭代法。

2.4.1 牛顿迭代法的基本思想

2.4.2 牛顿迭代法的几何解释

2.4.3 牛顿迭代法的收敛性分析

2.4.3 牛顿迭代法的算法实现

2.4.1 牛顿迭代法的基本思想

牛顿迭代法是一种重要和常用的迭代法，它的基本思想是**将非线性函数 $f(x)$ 逐步线性化**，从而将非线性方程 $f(x)=0$ 近似地转化为线性方程求解。

对于方程 $f(x)=0$ ，**设其近似根为 x_k** ，函数 $f(x)$ 可在 x_k 附近作泰勒展开。

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(\xi)}{2!}(x - x_k)^2, \quad \xi \text{ 在 } x_k \text{ 和 } x \text{ 之间.}$$

忽略高次项，用其线性部分作为函数 $f(x)$ 的近似，有

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

设 $f(x)=0$ 的根 x^* ，则有 $f(x^*)=0$ ，即

$$f(x_k) + f'(x_k)(x^* - x_k) \approx 0$$

2.4.1 牛顿迭代法的基本思想

$$x^* \approx x_k - \frac{f(x_k)}{f'(x_k)} \quad f(x_k) + f'(x_k)(x^* - x_k) \approx 0$$

将左端取为 x_{k+1} ，即 x_{k+1} 是比 x_k 更接近于 x^* 的近似值，即

$$x_{k+1} \approx x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots$$

这就是著名的牛顿迭代公式，相应的迭代函数为

$$\varphi(x) \approx x - \frac{f(x)}{f'(x)}.$$

2.4.4 牛顿迭代法的基本思想

例2.8 试建立计算 $\sqrt[3]{a}$ 的牛顿迭代格式，并求 $\sqrt[3]{411.791}$ 的近似值，要求迭代误差不超过0.005。

解：令 $x = \sqrt[3]{a}$, $f(x) = x^3 - a = 0$

则牛顿迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^3 - a}{3x_k^2} = \frac{2}{3}x_k + \frac{a}{3x_k^2}$$

$7^3=343$, $8^3=512$, $f(7)f''(7)<0$ 而 $f(8)f''(8)>0$ 。

取 $x_0=8$, 有

$$x_{k+1} = x_1 = \frac{2}{3}x_0 + \frac{a}{3x_0^2} = \frac{2}{3}*8 + \frac{411.791}{3*8*8} \approx 7.478078$$

$$x_1 = \frac{2}{3}x_0 + \frac{a}{3x_0^2} = \frac{2}{3}*8 + \frac{411.791}{3*8^2} \approx 7.439\ 078$$

$$x_2 = \frac{2}{3}x_1 + \frac{a}{3x_1^2} = \frac{2}{3}*7.478\ 078 + \frac{411.791}{3*7.478\ 078^2} \approx 7.439\ 956$$

$$|x_2 - x_1| = 0.038\ 122$$

$$x_3 = \frac{2}{3}x_2 + \frac{a}{3x_2^2} = \frac{2}{3}*7.439\ 956 + \frac{411.791}{3*7.439\ 956^2} \approx 7.439\ 760$$

$$|x_3 - x_2| = 0.000\ 196$$

于是取 $x^* \approx 7.439\ 760$

2.4.4 牛顿迭代法的基本思想

例2.9 用牛顿迭代法求 $x = e^{-x}$ 的根, $\epsilon=10^{-4}$

解:

因 $f(x) = xe^x - 1$, $f'(x) = e^x(x + 1)$ 建立迭代公式

$$x_{n+1} = x_n - \frac{x_n e^{x_n} - 1}{e^{x_n}(1 + x_n)} = x_n - \frac{x_n - e^{-x_n}}{1 + x_n}$$

取 $x_0=0.5$, 逐次计算得

$$x_1=0.57102$$

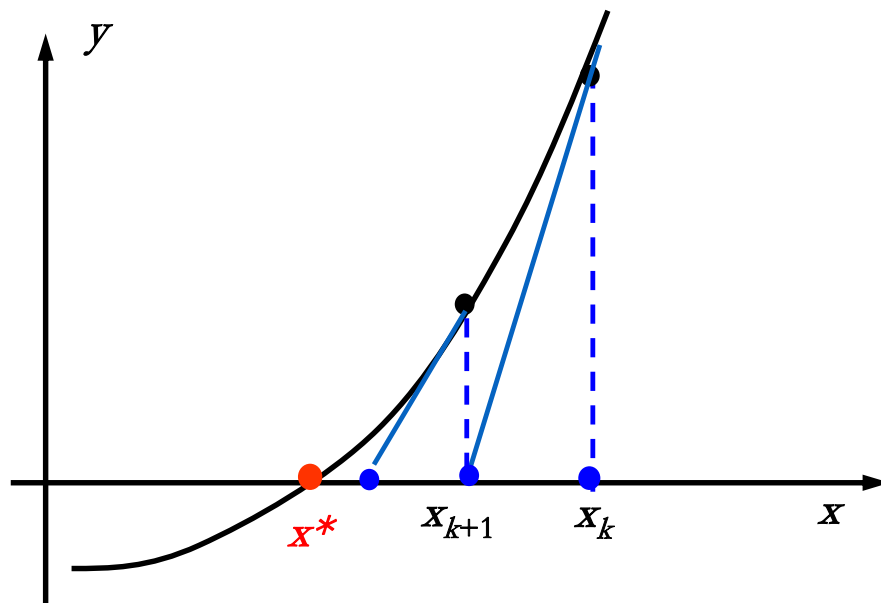
$$x_2=0.56716$$

$$x_3=0.56714$$

2.4.1 牛顿迭代法的几何解释

方程 $f(x)=0$ 的根 x^* 是曲线 $y=f(x)$ 与 x 轴交点的横坐标，设 x_k 是根 x^* 的某个近似值，过曲线 $y=f(x)$ 的横坐标为 x_k 的点 $P_k=(x_k, f(x_k))$ 引切线交 x 轴于 x_{k+1} ，并将其作为 x^* 新的近似值。

$$x_{k+1} \approx x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots$$



重复上述过程,一次次用切线方程可求解方程 $f(x)=0$ 的根。

2.4.3 牛顿迭代法的收敛性分析

例2.10：只用**加减乘除**运算，设计一个**二阶收敛**算法计算 \sqrt{a} ($a > 0$)

解：转化为求 $x^2 - a = 0$ 的正根

Newton 迭代： $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{1}{2} \left(x_k + \frac{a}{x_k} \right)$

$\Rightarrow x_{k+1} - \sqrt{a} = \frac{x_k^2 + a}{2x_k} - \sqrt{a} = \frac{x_k^2 + a - 2x_k\sqrt{a}}{2x_k} = \frac{(x_k - \sqrt{a})^2}{2x_k}$

$\Rightarrow \frac{x_{k+1} - \sqrt{a}}{(x_k - \sqrt{a})^2} = \frac{1}{2x_k} \Rightarrow \frac{1}{2\sqrt{a}} \Rightarrow$ 二阶收敛

回顾收敛阶的定义

设迭代 $x_{k+1} = g(x_k)$ 收敛到 $g(x)$ 的不动点 x^* 。记绝对误差 $e_k = x_k - x^*$ ，若

$$\lim_{k \rightarrow \infty} \left| \frac{e_{k+1}}{e_k^p} \right| = C \neq 0 \quad (C \text{ 为常数})$$

则称该迭代为**以收敛阶** p 收敛到 x^* 。数 C 称为渐近误差常数。

2.4.3 牛顿迭代法的收敛性分析

定理2.3 设 $f(x)$ 在其零点 x^* 的某个邻域内二阶连续可导且 x^* 是**单根**，则存在 x^* 的某个 δ 邻域 $N(x^*) = [x^* - \delta, x^* + \delta]$ ，使得对 $\forall x_0 \in N(x^*)$ ，牛顿迭代法产生的如下迭代定义的序列以**不低于二阶**的收敛速度收敛到 x^* ：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots$$

(局部收敛定理)

证明：若 x^* 是方程 $f(x)=0$ 的单根，则有

$$f(x^*) = 0, \quad f'(x^*) \neq 0.$$

Newton 法可以看作下面的不动点迭代：

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

考虑Taylor展开式

$$0 = f(x^*) = f(x_k) + f'(x_k)(x^* - x_k) + \frac{1}{2} f''(\xi)(x^* - x_k)^2 \quad \xi \in [x^*, x_k]$$

可得

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

$$x_k - \frac{f(x_k)}{f'(x_k)} - x^* = \frac{f''(\xi)}{2f'(x_k)} (x^* - x_k)^2$$

$$x_{k+1} - x^* = \frac{f''(\xi)}{2f'(x_k)} (x^* - x_k)^2$$

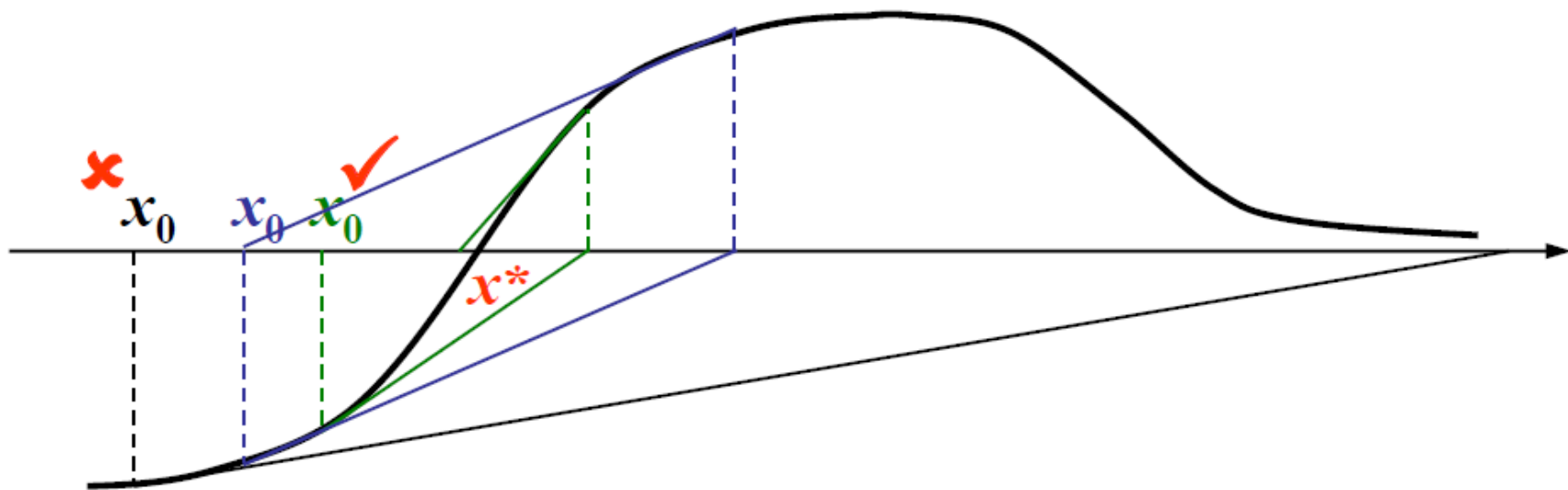
于是

$$\lim_{k \rightarrow \infty} \frac{|x^* - x_{k+1}|}{|x^* - x_k|^2} = \frac{|f''(x^*)|}{2|f'(x^*)|}$$

所以

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^2} = \lim_{k \rightarrow \infty} \frac{|x^* - x_{k+1}|}{|x^* - x_k|^2} = \frac{|f''(x^*)|}{2|f'(x^*)|}$$

Newton 法至少
二阶 局部收敛



牛顿迭代法对初值 x_0 的要求比较高， x_0 必须充分靠近 x^* ，才能保证局部收敛。

2.4.3 牛顿迭代法的收敛性分析

重根情形

□ 设 x^* 是 $f(x)$ 的 $m(m \geq 2)$ 重根, Newton法是否收敛?

$$f(x^*) = f'(x^*) = \cdots = f^{(m-1)}(x^*) = 0, f^{(m)}(x^*) \neq 0$$

Taylor 展式 $\Rightarrow f(x) = \frac{1}{m!} f^{(m)}(\xi_1)(x - x^*)^m$

Taylor 展式 $\Rightarrow f'(x) = \frac{1}{(m-1)!} f^{(m)}(\xi_2)(x - x^*)^{m-1}$

Taylor 展式 $\Rightarrow f''(x) = \frac{1}{(m-2)!} f^{(m)}(\xi_3)(x - x^*)^{m-2}$

Newton 迭代: $\varphi(x) = x - \frac{f(x)}{f'(x)}$

$$\Rightarrow \varphi'(x^*) = \lim_{x \rightarrow x^*} \varphi'(x) = \lim_{x \rightarrow x^*} \frac{f(x)f''(x)}{[f'(x)]^2} = 1 - \frac{1}{m}$$

\Rightarrow 线性收敛。 且重数 m 越高, 收敛越慢。

2.4.3 牛顿迭代法的收敛性分析

□ 提高收敛速度—提高收敛阶

法一：取 $\varphi(x) = x - m \frac{f(x)}{f'(x)} \Rightarrow \varphi'(x^*) = 0 \Rightarrow$ 二阶收敛

但 m 通常无法预先知道！

法二：将求 $f(x)$ 的重根转化为求另一个函数的单根。

令 $\mu(x) = \frac{f(x)}{f'(x)}$ ，则 x^* 是 $\mu(x)$ 的单重根。

构造针对 $\mu(x)$ 的具有二阶收敛的 Newton 迭代：

$$\varphi(x) = x - \frac{\mu(x)}{\mu'(x)} = x - \frac{f(x)f'(x)}{[f'(x)]^2 - f(x)f''(x)}$$

2.4.4 牛顿迭代法的算法实现

计算步骤

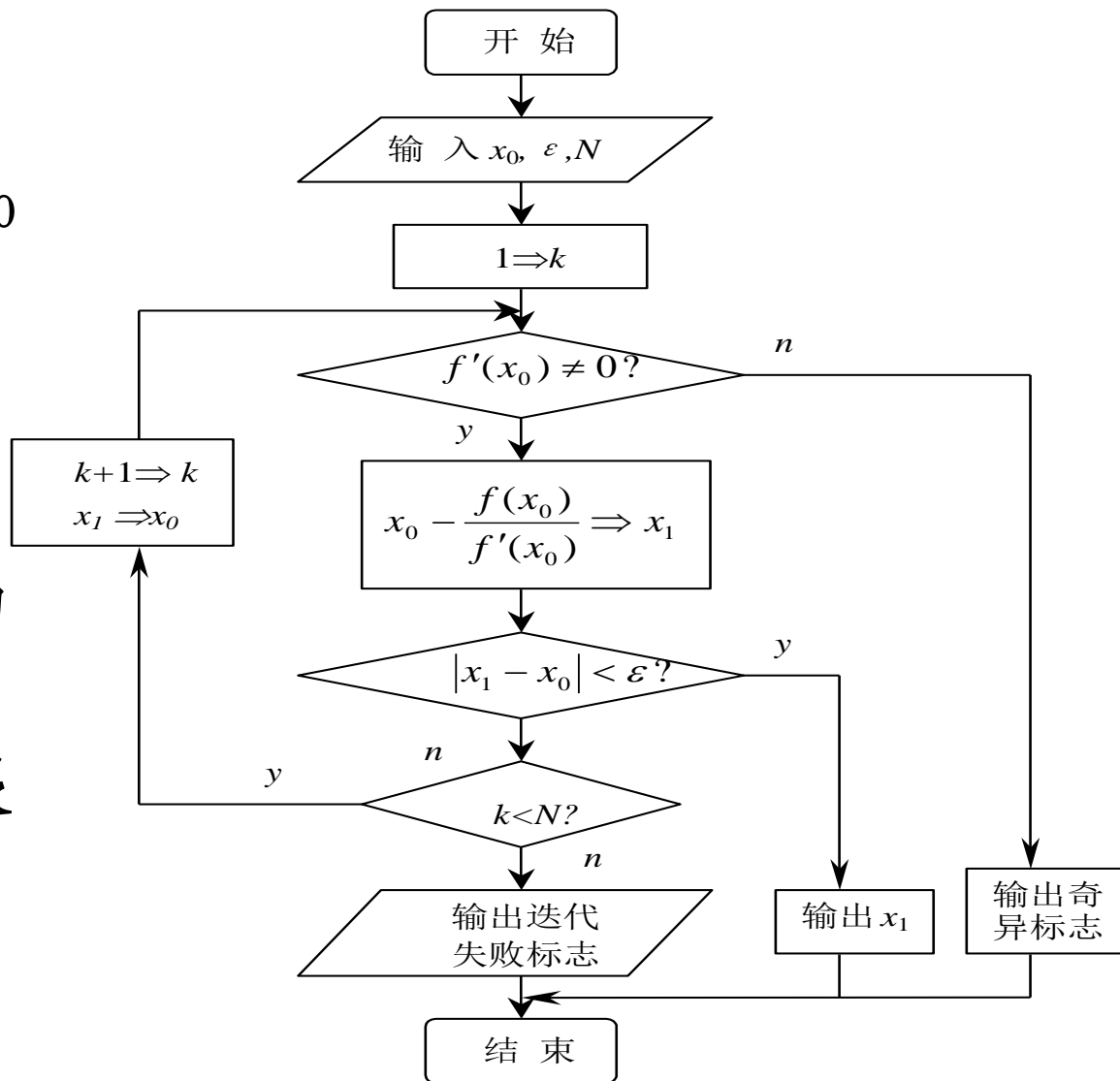
① 给出初始近似值 x_0 及精度 ε

② 计算

$$\left[x_0 - \frac{f(x_0)}{f'(x_0)} \right] \rightarrow x_1$$

① 若 $|x_1 - x_0| < \varepsilon$ 则转向 4)；否则转向 2)

② 输出满足精度的根 x_1 ，结束。



Matlab源程序:

Newton_Iteration.m

```

function [p0,err,k,y]=Newton_Iteration(f,df,p0,delta,epsilon,max1)
%Input   - f is the object function
%         - df is the derivative of f
%         - p0 is the initial approximation to a zero of f
%         - delta is the tolerance for p0
%         - epsilon is the tolerance for the function values y
%         - max1 is the maximum number of iterations
%Output  - p0 is the Newton-Raphson approximation to the zero
%         - err is the error estimate for p0
%         - k is the number of iterations
%         - y is the function value f(p0)
% f=@(x) x*exp(x)-1; df=@(x) exp(x)*(x+1);
% p0=1; delta=1e-4; epsilon=1e-3; max1=500;
% Newton_Iteration(f,df, p0, delta, epsilon, max1)
for k=1:max1
    p1=p0-f(p0)/df(p0);
    err=abs(p1-p0);
    relerr=2*err/(abs(p1)+delta);
    p0=p1;
    y=f(p0);
    if (err<delta)|(relerr<delta)|(abs(y)<epsilon),
        break,
    end
end
end

```

作业2.3

4. 设 $f(x) = x^3 - 3x - 2$ 。

(a) 求出牛顿 - 拉夫森公式 $p_k = g(p_{k-1})$ 。

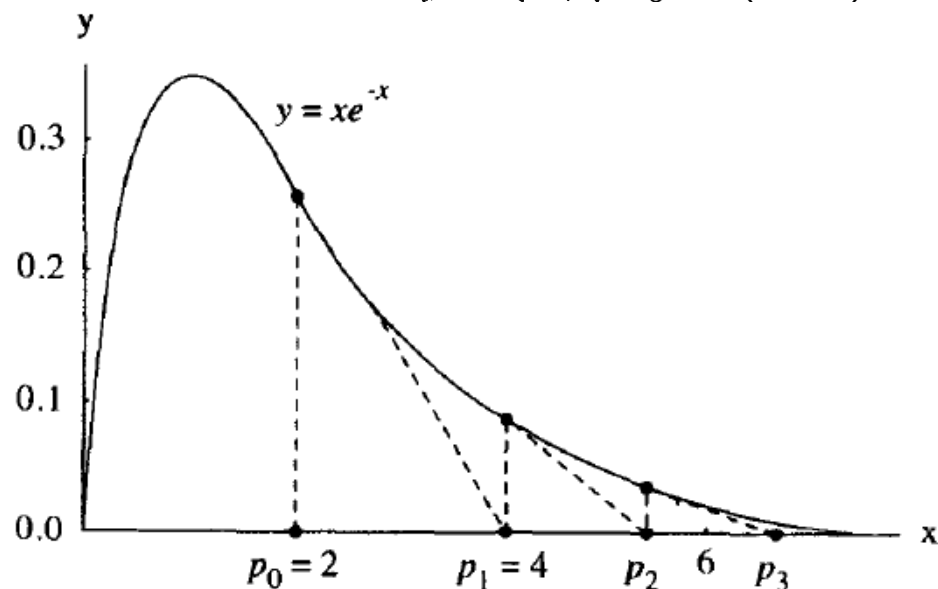
(b) 从 $p_0 = 2.1$ 开始, 求 p_1, p_2, p_3 和 p_4 。

(c) 序列是二次收敛的还是线性收敛的?

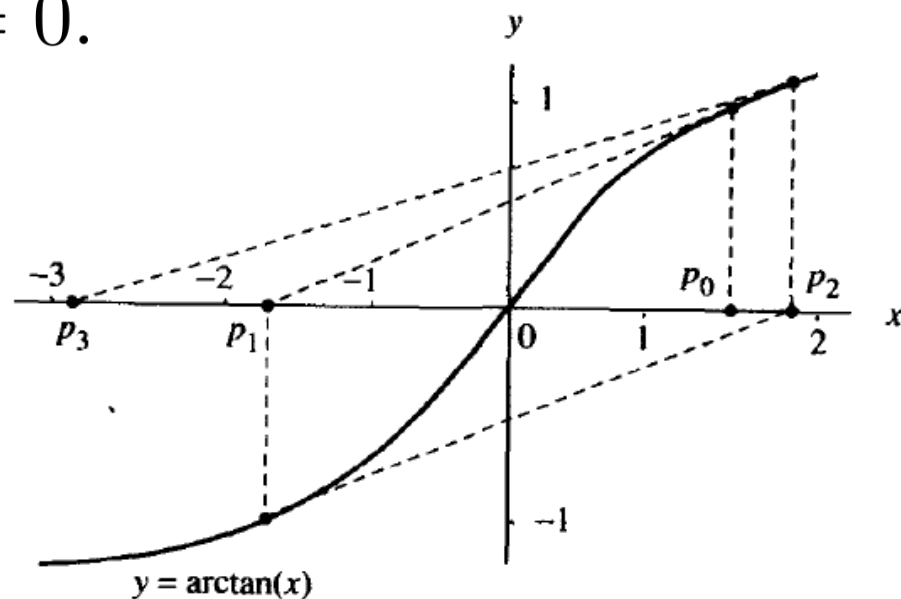
进一步分析

牛顿迭代法虽然具有收敛速度快的优点，但每迭代一次都要计算导数 $f'(x_k)$ ，当 $f(x)$ 比较复杂时，不仅每次计算 $f'(x_k)$ 带来很多不便，而且还可能十分麻烦。如果不用计算导数的迭代方法，往往只有线性收敛的速度。

缺陷： 1. 被零除 $f'(x_k) = 0$.



2. 迭代产生一个发散序列



3. 迭代产生一个发散循环序列

2.5 割线法

本节介绍的割线法便是一种不必进行导数运算的求根方法。割线法在迭代过程中不仅用到前一步 x_k 处的函数值，而且还使用 x_{k-1} 处的函数值来构造迭代函数，这样做能提高迭代的收敛速度。

2.5.1 割线法的基本思想

2.5.2 割线法的几何意义

2.5.3 割线法的收敛性分析

2.5.4 割线法的算法实现

2.5.1 割线法的基本思想

为避免计算函数的导数 $f'(x_k)$ ，使用差商

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

替代牛顿公式中的导数 $f'(x_k)$ ，便得到迭代公式

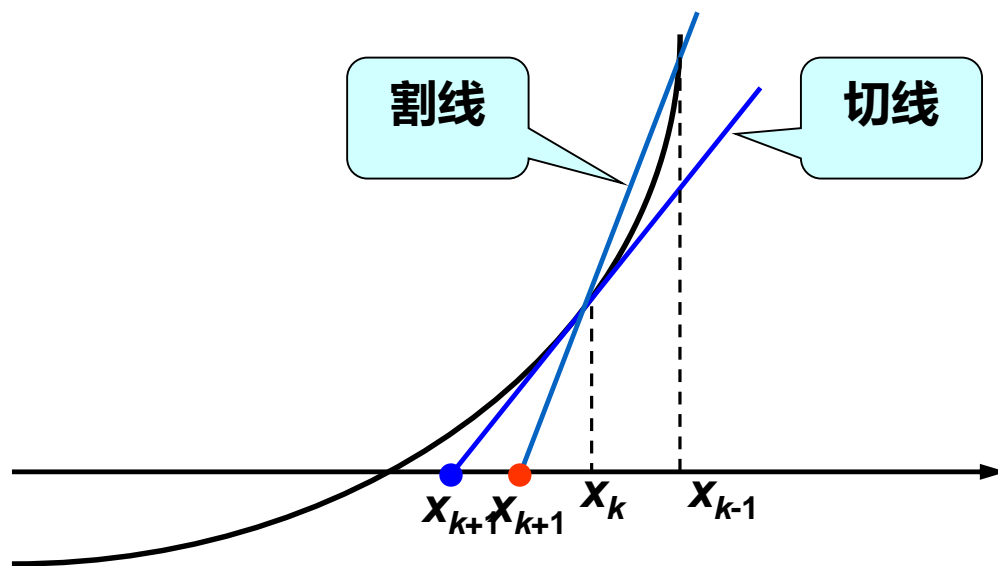
$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})} (x_k - x_{k-1}), \quad k = 1, 2, 3, \dots$$

称为割线迭代公式，相应的迭代法称为割线法。

这种方法称为多点迭代法。

2.5.2 割线法的几何意义

用过曲线上两点 $P_0(x_0, f(x_0))$ 、 $P_1(x_1, f(x_1))$ 的割线来代替曲线，用割线与 x 轴交点的横坐标作为方程的近似根 x_2 。再过 P_1 点和点 $P_2(x_2, f(x_2))$ 作割线求出 x_3 。以此类推，当收敛时可求出满足精度要求的 x_k 。



切线斜率 \approx 割线斜率

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

- ✓ 只需计算函数值，避免计算导数；
- ✓ 需要两个初始点；
- ✓ 收敛比牛顿迭代法稍慢，但对初始点要求同样高。

2.5.2 割线法的几何意义

例2.11 用割线法求方程 $x=e^{-x}$ 在 $x_0=0.5$ 初始值邻近的一个根。

要求 $|x_{k+1}-x_k|<0.0001$ 。

解：取 $x_0=0.5$ 、 $x_1=0.6$ ，令 $f(x)=x-e^{-x}$ 利用割线迭代公式，有

$$x_{k+1} = x_k - \frac{(x_k - e^{-x_k})}{(x_k - x_{k-1}) - (e^{-x_k} - e^{-x_{k-1}})}(x_k - x_{k-1})$$

计算结果如下表

k	x_k	$ x_{k+1}-x_k $
0	0.500 00	
1	0.600 00	
2	0.567 54	0.032 46
3	0.567 15	0.000 39
4	0.567 14	0.000 01

取近似根 $x_4=0.56714$ ，
即可满足精度要求。

2.5.3 割线法的收敛性分析

定理2.4 设 x^* 是 $f(x)$ 的单重零点, $f''(x)$ 在 x^* 的某个邻域内连续, 则存在 x^* 的一个邻域 $N(x^*) = [x^* - \delta, x^* + \delta]$, 使得当 $x_0, x_1 \in N(x^*)$ 时, 由两步弦截法产生的序列收敛到 x^* , 且收敛阶为 $(\sqrt{5}+1)/2 \approx 1.618$

2.5.4 割线法的算法实现

割线法的收敛速度比牛顿迭代法要慢，但优点是不需要计算导数。

1. 计算步骤

(1) 选定初值 x_0 、 x_1 ，并计算相应的函数值 $f(x_0)$ 、 $f(x_1)$ ；

(2) 计算

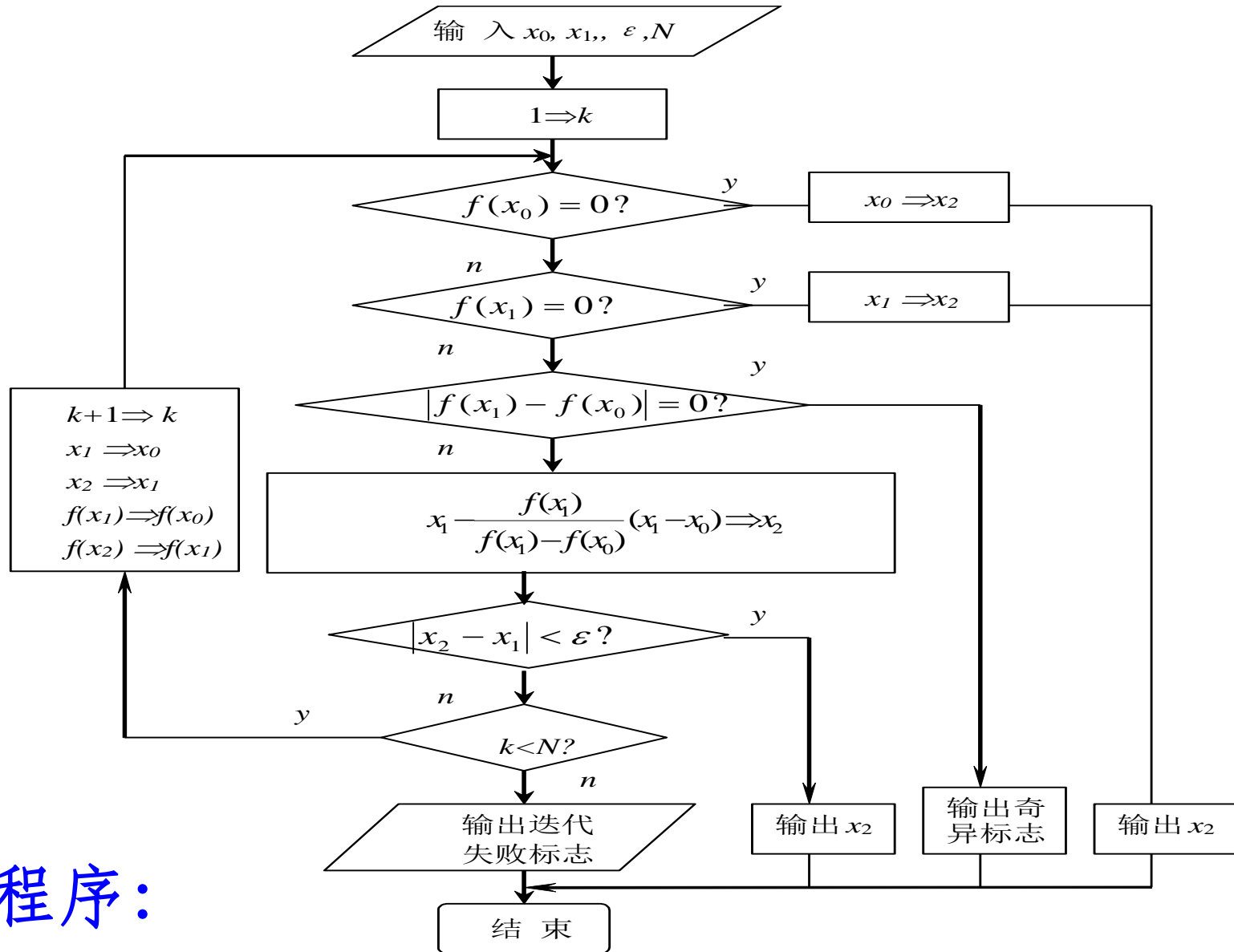
$$x_2 = x_1 - \frac{f(x_1)}{f(x_1) - f(x_0)}(x_1 - x_0) \quad x_1 \rightarrow x_0, x_2 \rightarrow x_1$$

(3) 若 $|x_1 - x_0| < \varepsilon$ (ε 为给定的精度要求)，

则转(4)，否则转向(2)；

(4) 输出满足精度要求的根 x_1 ，结束。

2.5.4 割线法的算法实现



Matlab源程序:
Gexianfa_Iteration.m

Matlab源程序： Gexianfa_Iteration.m

```
function [p1,err,k,y]=Gexianfa_Iteration(f,p0,p1,delta,epsilon,max1)

%Input   - f is the object function
%         - p0 and p1 are the initial approximations to a zero of f
%         - delta is the tolerance for p1
%         - epsilon is the tolerance for the function values y
%         - max1 is the maximum number of iterations
%Output  - p1 is the secant method approximation to the zero
%         - err is the error estimate for p1
%         - k is the number of iterations
%         - y is the function value f(p1)
% f=@(x) x*exp(x)-1; ;
% p0=1; p1=1.2; delta=1e-4; epsilon=1e-3; max1=500;
% Gexianfa_Iteration(f,p0, p1, delta, epsilon, max1)

for k=1:max1
    p2=p1-f(p1)*(p1-p0)/(f(p1)-f(p0));
    err=abs(p2-p1);
    relerr=2*err/(abs(p2)+delta);
    p0=p1;
    p1=p2;
    y=f(p1);
    if (err<delta)|(relerr<delta)|(abs(y)<epsilon),
        break,
    end
end
end
```

2.6 迭代收敛的加速办法(选讲)

对于收敛的迭代过程足够多次，就可以使结果达到任意的精度。但有时迭代过程收敛缓慢，从而使计算量变得很大。因此，可考虑对迭代作加速处理。

2.6.1 埃特金过程

埃特金加速技术可加速任意线性收敛的序列。

2.6.1 埃特金(Aitken)过程

定义2.2 设有序列 $\{p_n\}_{n=0}^{\infty}$, 用如下表达式定义前向微分 Δp_n :

$$\Delta p_n = p_{n+1} - p_n, n \geq 0.$$

高阶 $\Delta^k p_n$ 可定义为

$$\Delta^k p_n = \Delta^{k-1} (\Delta p_n), k \geq 2.$$

定义2.3 (埃特金加速) 设序列 $\{p_n\}_{n=0}^{\infty}$ 线性收敛到极限 p , 而且对所有 $n \geq 0$, 有 $p - p_n \neq 0$. 如果存在实数 A , 且 $|A| < 1$, 满足

$$\lim_{n \rightarrow \infty} \frac{p - p_{n+1}}{p - p_n} = A.$$

则定义为

$$q_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n} = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$$

的序列 $\{q_n\}_{n=0}^{\infty}$ 收敛到 p , 且比 $\{p_n\}_{n=0}^{\infty}$ 快, 即 $\lim_{n \rightarrow \infty} \frac{q - q_n}{p - p_n} = 0$.

2.6.1 埃特金过程

例2.12 用埃特金方法对不动点迭代序列

$$p_0 = 0.5, p_{k+1} = e^{-p_k}, k = 0, 1, 2, \dots$$

进行加速.

$$q_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$$

表 2.10 线性收敛序列 $\{p_n\}$

n	p_n	$E_n = p_n - p$	$A_n = \frac{E_n}{E_{n-1}}$
1	0.606530660	0.039387369	-0.586616609
2	0.545239212	-0.021904079	-0.556119357
3	0.579703095	0.012559805	-0.573400269
4	0.560064628	-0.007078663	-0.563596551
5	0.571172149	0.004028859	-0.569155345
6	0.564862947	-0.002280343	-0.566002341

表 2.11 用埃特金 Δ^2 过程得到的序列 $\{q_n\}$

n	q_n	$q_n - p$
1	0.567298989	0.000155699
2	0.567193142	0.000049852
3	0.567159364	0.000016074
4	0.567148453	0.000005163
5	0.567144952	0.000001662
6	0.567143825	0.000000534

显然，序列 $\{q_n\}_{n=0}^{\infty}$ 的收敛要比 $\{p_n\}_{n=0}^{\infty}$ 快。

作业2.5

6. 序列 $p_n = 1/(4^n + 4^{-n})$ 线性收敛到 0。利用埃特金公式(4)求 q_1, q_2 和 q_3 , 从而加速收敛。

n	p_n	q_n
0	0.5	-0.26437542
1	0.23529412	
2	0.06225681	
3	0.01562119	
4	0.00390619	
5	0.00097656	

另外两种加速方法自学：

米勒法和斯蒂芬森加速法

本章教学要求及重点难点

- 理解方程根的基本概念
- 掌握二分法、迭代法、牛顿迭代法、割线法求方程近似根的基本方法
- 掌握迭代过程的敛散性证明
- 掌握迭代法、Newton迭代法的算法设计思想
- 重点：迭代法求根原理、迭代算法的基本思想
- 难点：迭代法的收敛性分析与证明
- 了解：埃特金加速技术