

非线性方程求根

浙江大学控制学院

非线性方程求根

- 方程

$$f(x) = ax^2 + bx + c = 0 \implies x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- 代数方程求根问题是一古老的数学问题，十六世纪人们找到了求三次、四次方程根的公式，十九世纪证明了大于等于五次的代数方程没有一般的求根公式。

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \implies x = ?$$
$$\sin x + x = 0 \implies x = ?$$

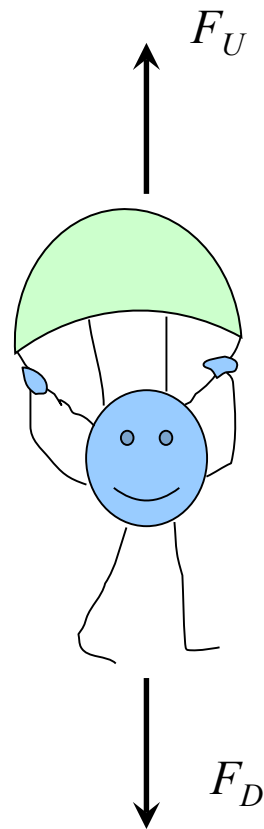
非线性方程求根

● 例1——问题：

- 求解降落伞的阻力系数 c ，其中降落伞的质量为 $m=68.1\text{kg}$ ，要求降落伞自由落体运动 $t=10\text{s}$ 后速率到 40m/s 。
- 解：求解下列方程的根

$$v(t) = \frac{gm}{c} \left(1 - e^{-(c/m)t} \right) = 40$$

$$f(c) = \frac{9.8(68.1)}{c} \left(1 - e^{-(c/68.1)10} \right) - 40 = 0$$

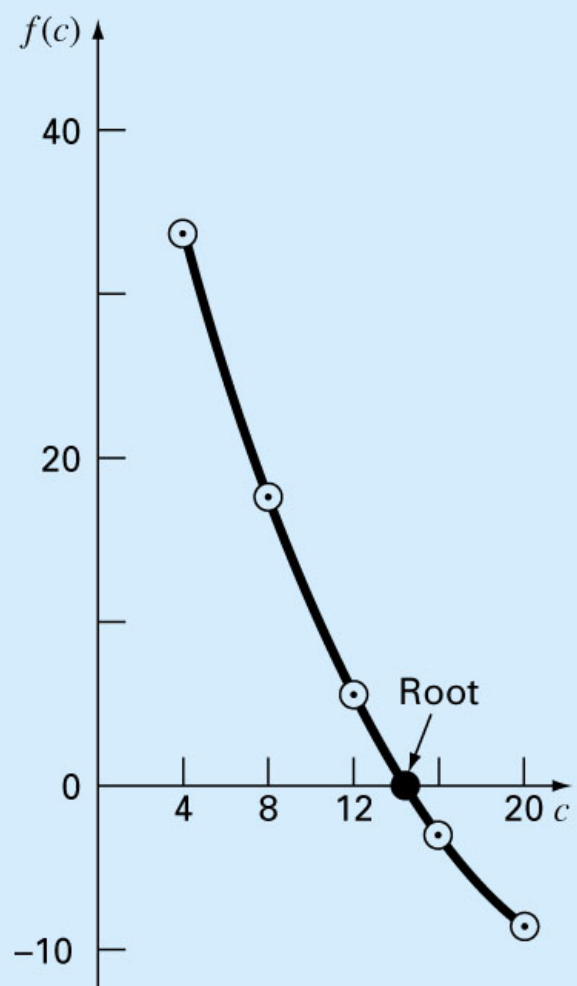


非线性方程求根——图解法

c	$f(c)$
4	34.115
8	17.653
12	6.067
16	-2.269
20	-8.401

- 根的初步估计值为14.75
- 检验： $f(14.75) = 0.059$
 $v = 40.059$

图解法不精确，但可以提供初始猜测值，理解函数性质和预测数值方法缺陷。



非线性方程求根

- 二分法
- 试位法
- 不动点迭代
- Newton-Raphson方法
- 割线法

划界法
Bracketing
Methods

开方法
Open Methods

非线性方程求根

- 设有一非线性方程

$$f(x) = 0$$

- 其中, $f(x)$ 为实变量 x 的非线性函数。

- 定义:

- 如果有 x^* 使 $f(x^*)=0$, 则称 x^* 为方程 $f(x)=0$ 的**根**, 或称为函数 $f(x)$ 的**零点**。

- 当 $f(x)$ 为多项式时, 即方程为 $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = 0 \quad (a_n \neq 0)$

称 $f(x)=0$ 为 n 次**代数方程**。当 $f(x)$ 包含指数函数或三角函数等特殊函数时, 称 $f(x)=0$ 为**超越方程**。

- 如果 $f(x)$ 可分为 $f(x) = (x - x^*)^m g(x)$

其中, $g(x^*) \neq 0$, m 为正整数, 则称 x^* 为 $f(x)=0$ 的 m **重根**, 当 $m=1$ 时称 x^* 为 $f(x)=0$ 的**单根**。

非线性方程求根的一般方法

- 两个问题

- 求解代数方程和超越方程的实数根

- 预先给出根的一个初略位置（**根的分离**），然后根据这个位置向真实根逼近（**近似根的精确化**）

- 求解多项式的所有实数根和复数根

- 专为多项式设计，系统化地求解多项式的所有根，而不仅仅是逼近的实数根

- 定理

- 对一般函数方程，若 $f(x)$ 在区间 $[a, b]$ 上连续，且 $f(a)f(b) < 0$ ，则方程 $f(x)=0$ 在 $[a, b]$ 上至少有一个实根， $[a, b]$ 上称为有根区间。

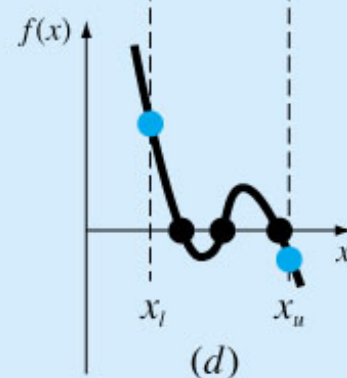
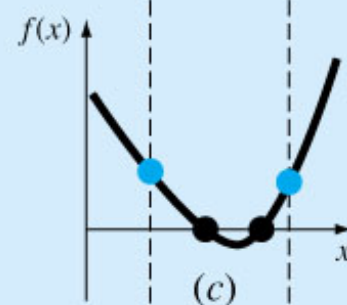
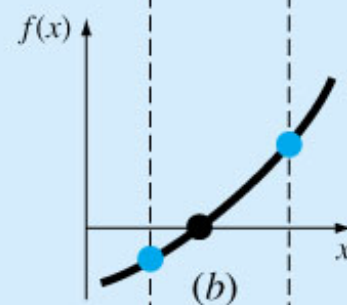
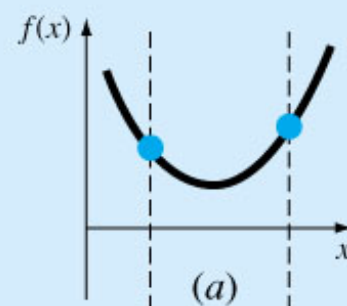
非线性方程根

- 由下界和上界确定的区间中根出现的情况：

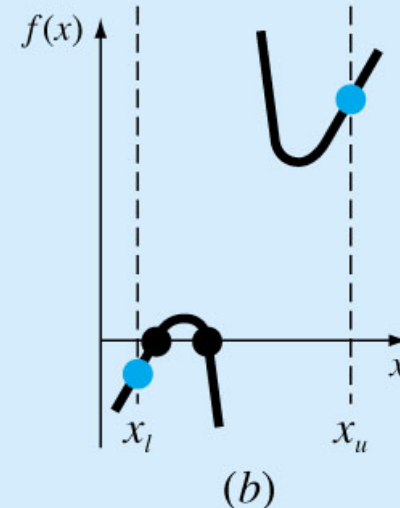
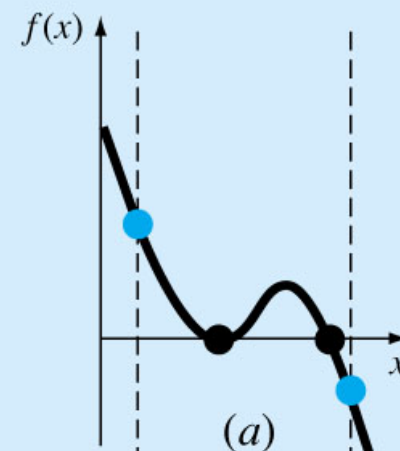
- (a) $f(x_l)$ 和 $f(x_u)$ 符号相同，无根；
- (b) $f(x_l)$ 和 $f(x_u)$ 符号相反，单根；
- (c) $f(x_l)$ 和 $f(x_u)$ 符号相同，偶数个根；
- (d) $f(x_l)$ 和 $f(x_u)$ 符号相反，奇数个根；

特殊情况：

- (a) 函数和 x 轴相切， $f(x_l)$ 和 $f(x_u)$ 符号相反，但有偶数个根；
- (b) 不连续函数的情况。



一般情况



特殊情况

非线性方程求根

- 二分法（二元截断法、区间等分法、波尔察诺(Bolzano)方法）
- 试位法
- 不动点迭代
- Newton-Raphson方法
- 割线法

二分法

- 基本思想

- 首先确定有根区间，将区间二等分，通过判断 $f(x)$ 的符号，逐步将有根区间缩小，直至有根区间足够地小，便可求出满足精度要求的近似根。

二分法

步骤

1. 猜测初始下界 x_l 和上界 x_u , 使 $f(x_l)f(x_u)<0$;
2. 计算中间点

$$x_r = \frac{x_l + x_u}{2}$$

3. 根据情况确定根所在的区间

- a) 如果 $f(x_l)f(x_r)<0$, 则根落在左边子区间, 取 $x_u=x_r$, 返回步骤2;
- b) 如果 $f(x_l)f(x_r)>0$, 则根落在右边子区间, 取 $x_l=x_r$, 返回步骤2;
- c) 如果 $f(x_l)f(x_r)=0$, 则 x_r 根为所要求的根, 算法终止。

4. 终止条件

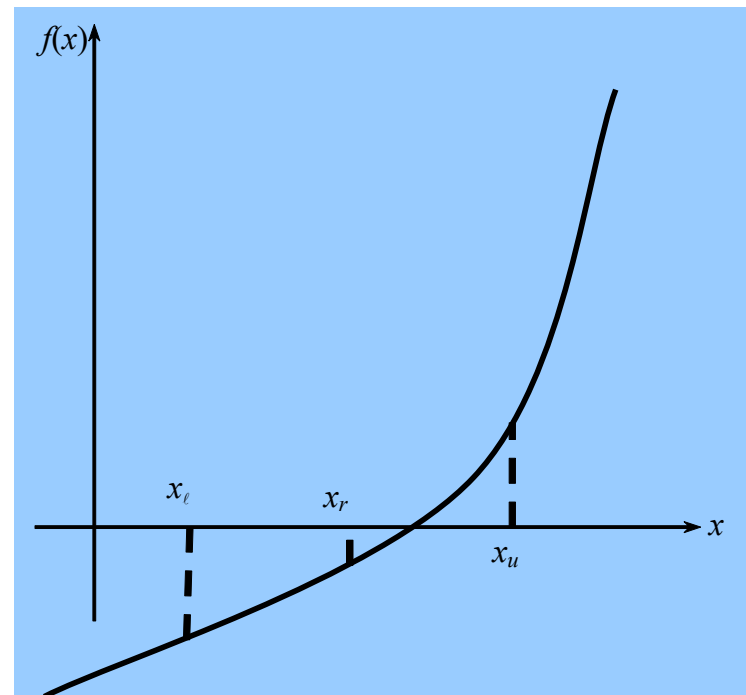
- 新的估计值

$$x_r^{new} = \frac{x_l + x_u}{2}$$

- 近似相对误差

$$\varepsilon_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$$

- 当 ε_a 小于既定的终止条件 ε_s 或迭代次数达到一个上界时, 计算终止。



二分法——例2

- 初始 $x_l=12$, $x_u=16$

$$x_r = \frac{12+16}{2} = 14$$

误差基于方程真实根估计，实际上是不可能的。

相对误差为 $\varepsilon_t = 5.3\%$ (真值为14.7802)

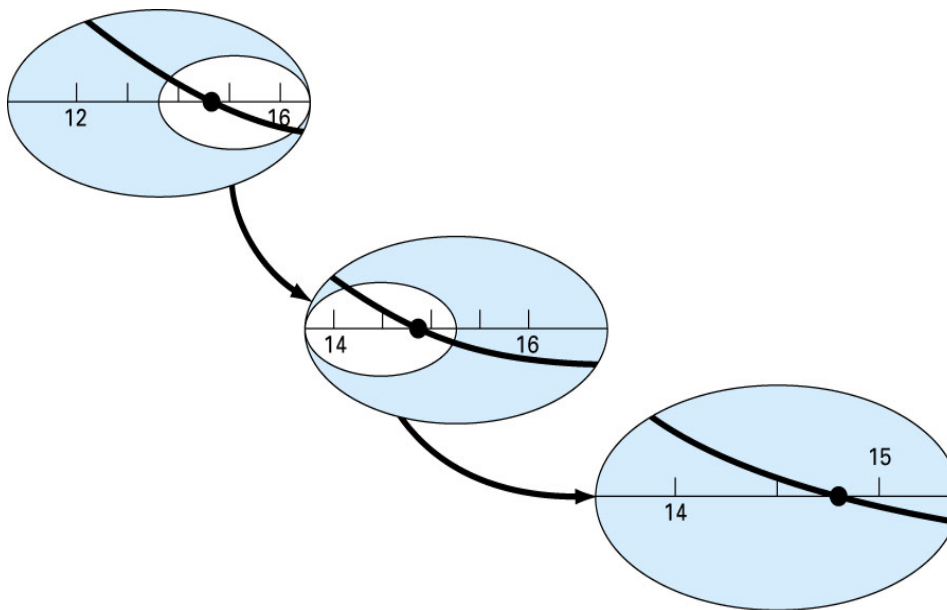
- 计算 $f(12)f(14)=9.517>0$
- 新的 $x_l=14$, 根的估计值

$$x_r = \frac{14+16}{2} = 15$$

近似相对误差为 $\varepsilon_a = 6.67\%$

- 计算 $f(14)f(15)=-0.666<0$
- 新的上界 $x_u=15$, $x_r = \frac{14+15}{2} = 14.5$

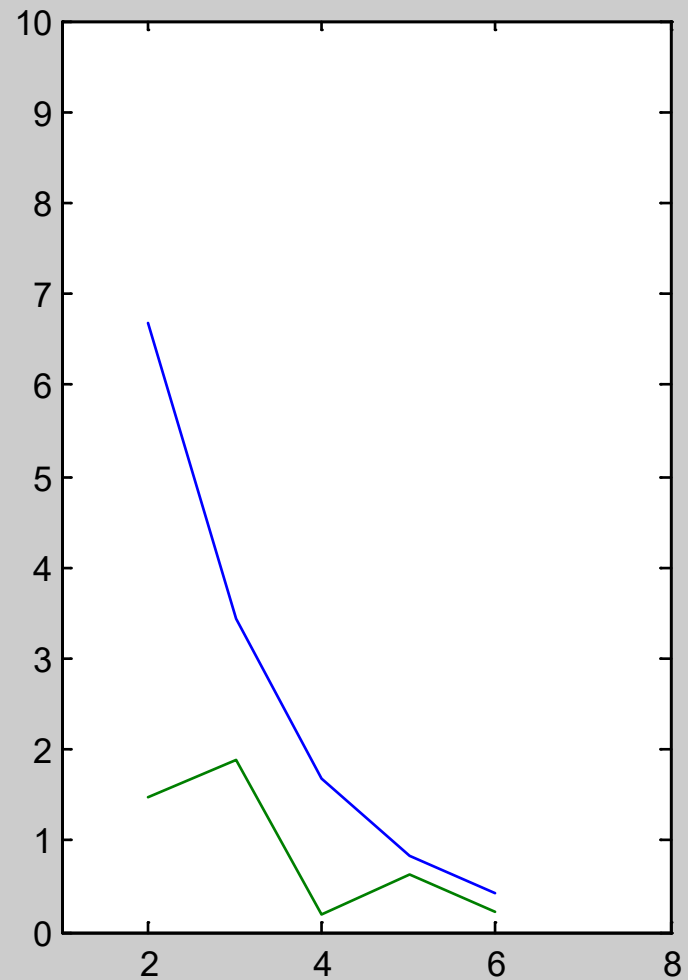
近似相对误差为 $\varepsilon_a = 3.45\%$



二分法——误差估计

迭代次数	x_l	x_u	x_r	ε_a	ε_t
1	12	16	14		
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14.75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

- 不是每次迭代后真实误差都会减小。
- 近似误差总是大于真实误差，当 ε_a 小于 ε_s 时，真实误差一定也小于 ε_s ，所以根可以达到要求的精度，算法可以终止。



二分法——误差估计

- 每次根的估计值为

$$x_r = \frac{x_l + x_u}{2}$$

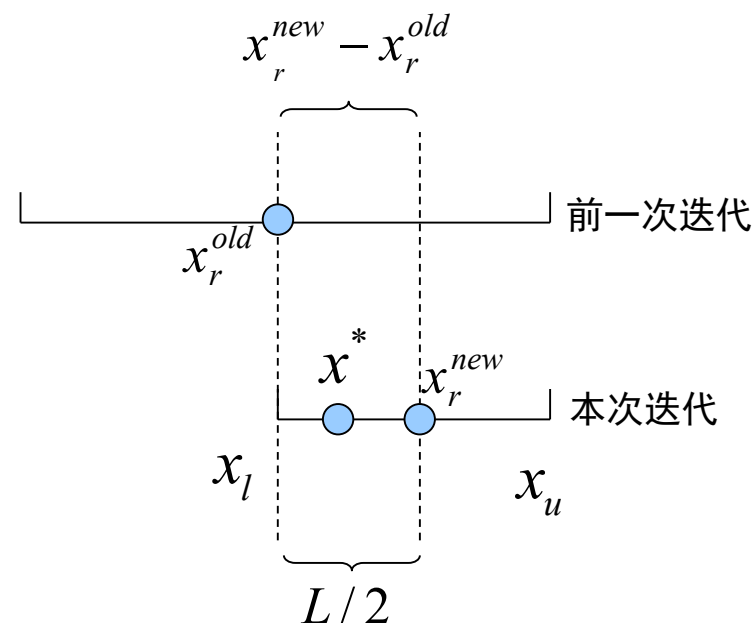
- 真实根落在长度为
的区间中，

$$\frac{x_u - x_l}{2} = \frac{L}{2}$$

$$x_r^{new} - x_r^{old} = \frac{x_u - x_l}{2}$$

$$|x_r^{new} - x_r^{old}| \geq |x_r^{new} - x^*|$$

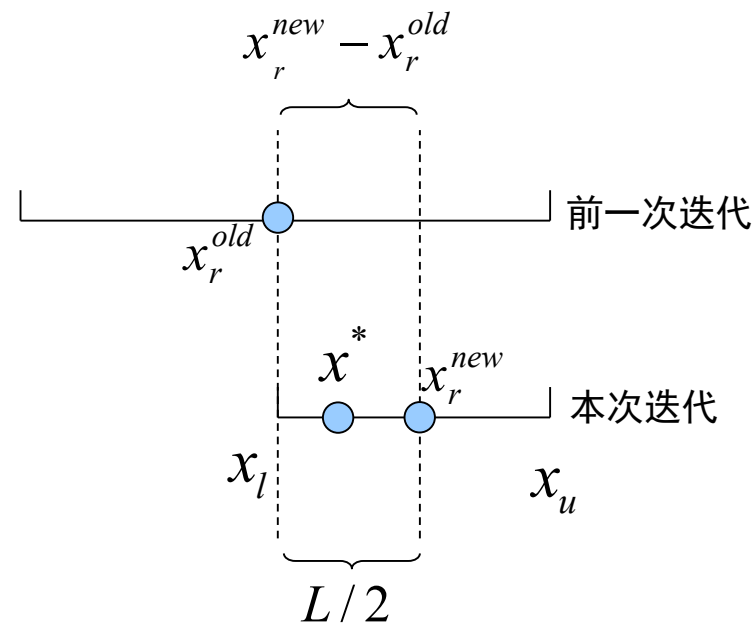
因此，近似误差为真实误差的一个精确上界。



二分法——迭代次数

- 每次迭代根的区间为上次迭代的一半：
 - 初始区间 L_0 ，第一次迭代后 $L_1 = L_0 / 2$ ，...，第 k 次迭代后区间长度 $L_k = L_0 / 2^k$
- 第 k 次迭代的绝对误差 $\leq L_k$
- 如果已知最终的期望绝对误差 $E_{a,d}$ ，
可以计算迭代次数

$$n = \frac{\log(L_0 / E_{a,d})}{\log 2}$$



二分法——参考程序

```
• function [c,err,yc]=bisect(f,a,b,delta)

%Input - f is the function input as a string 'f'
%       - a and b are the left and right endpoints
%       - delta is the tolerance
%Output - c is the zero
%        - yc= f(c)
%        - err is the error estimate for c

ya=feval(f,a);
yb=feval(f,b);
if ya*yb > 0, break, end
max1=1+round((log(b-a)-log(delta))/log(2));
for k=1:max1
    c=(a+b)/2;
    yc=feval(f,c);
    if yc==0
        a=c;
        b=c;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    if b-a < delta, break, end
end

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

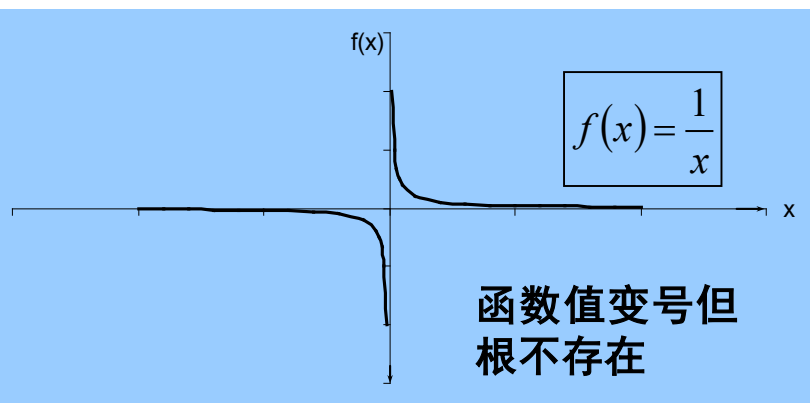
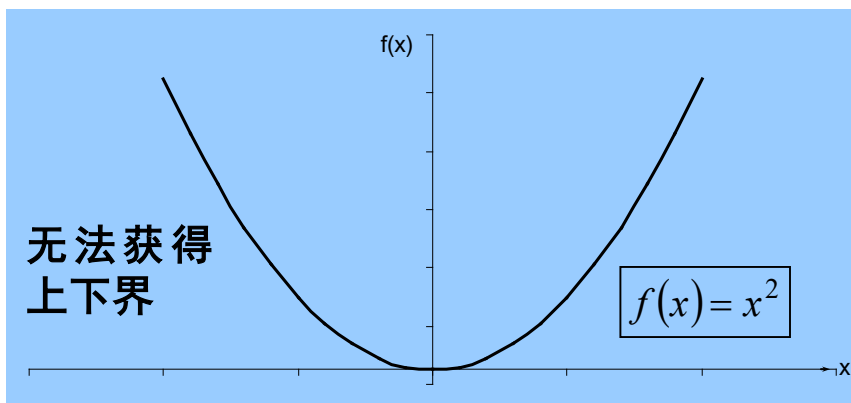

二分法——评价

● 优点

- 简单
- 通常可以找到根
- 可以计算出需要的迭代次数

● 缺点

- 慢
- 要求上下界已知
- 多根情况无法处理
- 没有考虑 $f(x_l)$ 和 $f(x_u)$ 的大小。如果 $f(x_l)$ 更接近0，那么根可能更加接近 x_l 。



非线性方程求根

- 二分法
- 试位法 (regula falsi、false-position、线性插值方法)
- 不动点迭代
- Newton-Raphson方法
- 割线法

试位法 (False position)

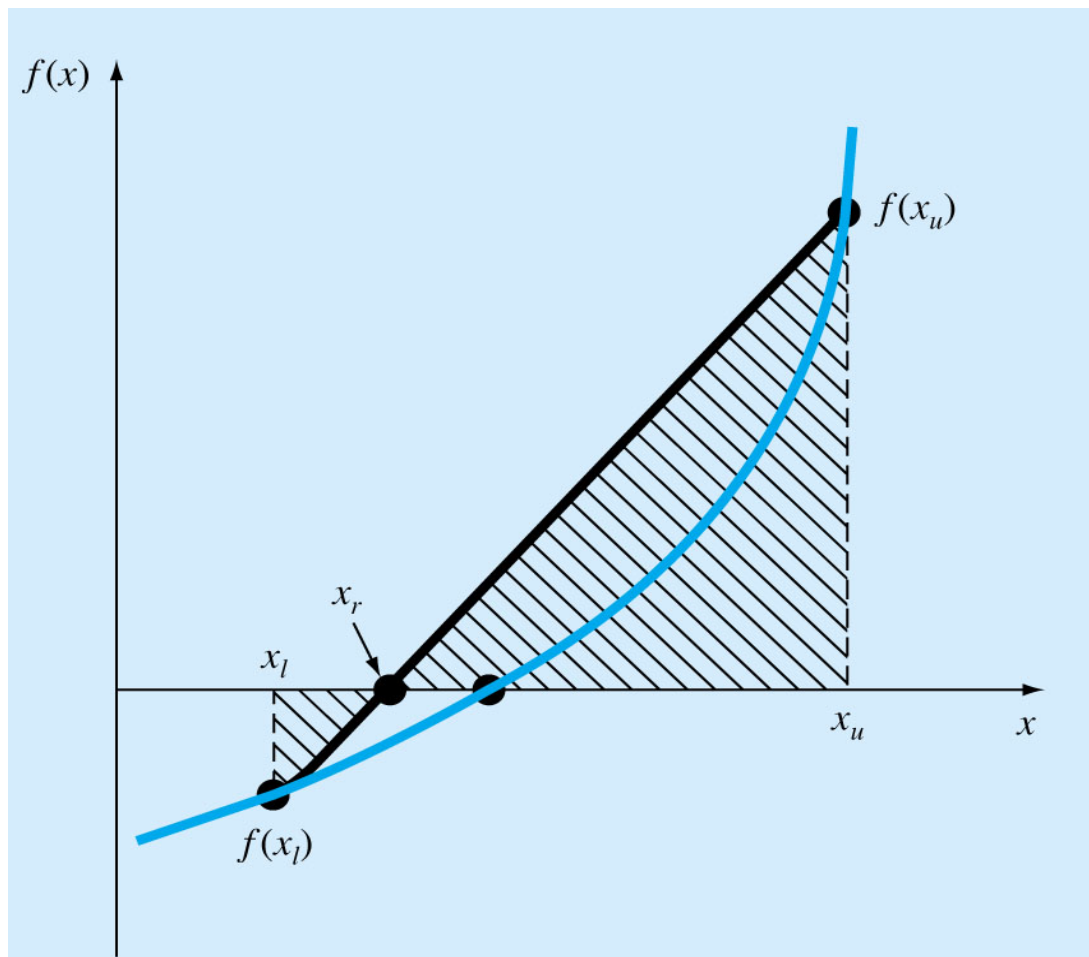
- 通过一条直线连接 $[x_l, f(x_l)]$ 和 $[x_u, f(x_u)]$, 直线与 x 轴的交点作为新的根的估计值。

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

$$x_r = \frac{x_l f_u - x_u f_l}{f_u - f_l}$$

$$x_r = x_u - \frac{f_u(x_l - x_u)}{f_l - f_u}$$

少一次乘法计算



试位法——参考程序

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

%Input - f is the function input as a string 'f'
%       - 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

ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0
    disp('Note: f(a)*f(b) >0'),
    break,
end
for k=1:max1
    dx=yb*(b-a)/(yb-ya);
    c=b-dx;
    ac=c-a;
    yc=feval(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);
yc=feval(f,c);
```

试位法

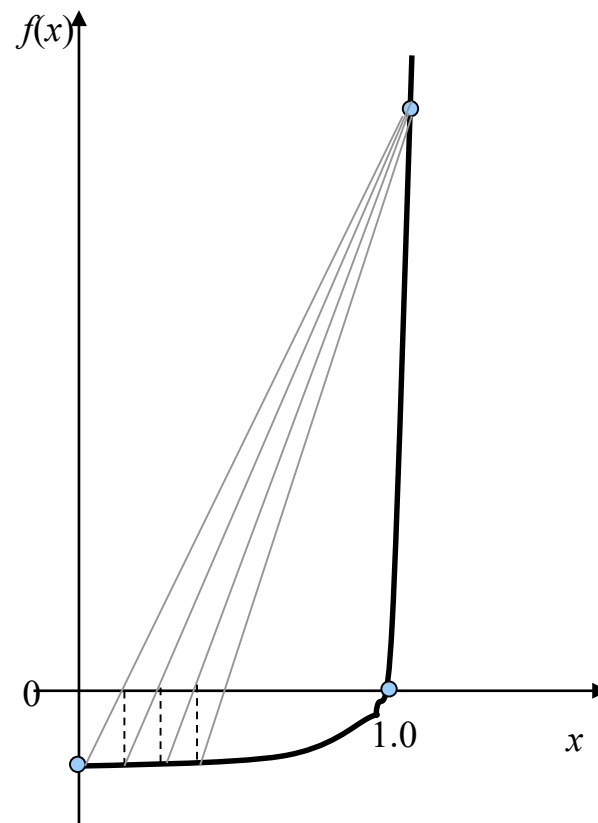
- 使用试位法求解例2中方程的根，并与二分法进行对比。
- 解： $x_l=12$ ， $x_u=16$

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	ε_t	ε_a
1	12	16	6.0699	-2.2688	14.9113	0.89%	
2	12	14.9113	6.0699	-0.2543	14.7942	0.09%	0.79%

试位法比二分法误差减小得快。

试位法——缺陷

- 问题：求解方程 $f(x)=x^{10}-1$ 在 $x=0$ 和 1.3 之间的根
- 试位法收敛慢！
 - 原因——违反了试位法的前提！在这个例子中， $f(x_l)$ 更接近 0 ，但根接近 x_u 。
- 近似误差小于真实误差
 - 除了检查迭代近似误差外，把根的估计值代入原方程中，检验是否接近 0
- 一个划界点保持不动可能导致很差的收敛性。



试位法——改进

- 修正的试位法
 - 检测一个边界何时固定不变
 - 使用计数器来确定一个边界两次迭代保持不变
 - 如果出现这种情况，将停滞的边界点处的函数值变为原来的一半

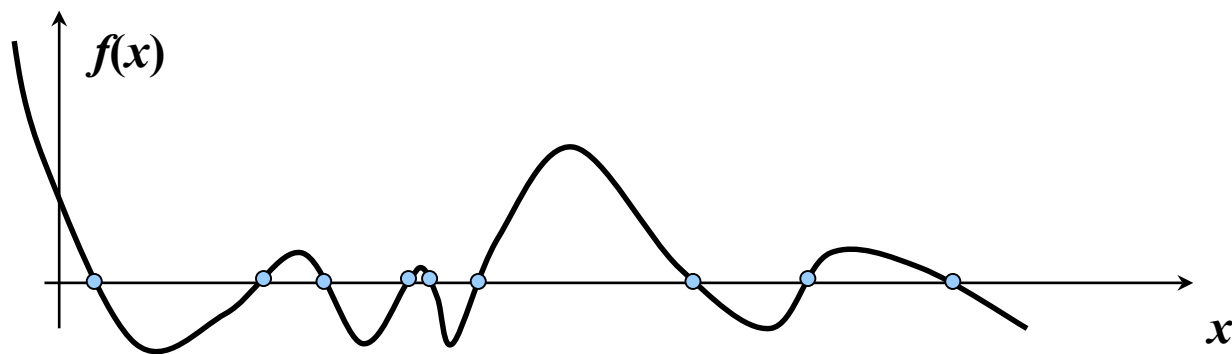
$$x_r = x_u - \frac{f_u(x_l - x_u)}{f_l - f_u}$$

增量搜索和确定初始猜值

- 函数作图

- 增量搜索

- 从感兴趣区域的一端开始，小增量地穿过区域并计算函数值，如果有函数值改变符号时（ $f(x_k)f(x_k + \Delta x) < 0$ ），可以确定由一个根落在这个小增量区间内。
- 小增量区间的两个端点可以作为初始猜值
- 增量长度的确定：太小费时，太大丢失彼此相差近的根。
- 计算区间端点处的一阶导数值，如果有符号改变，则区间中存在最大或最小值，减小区间继续搜索

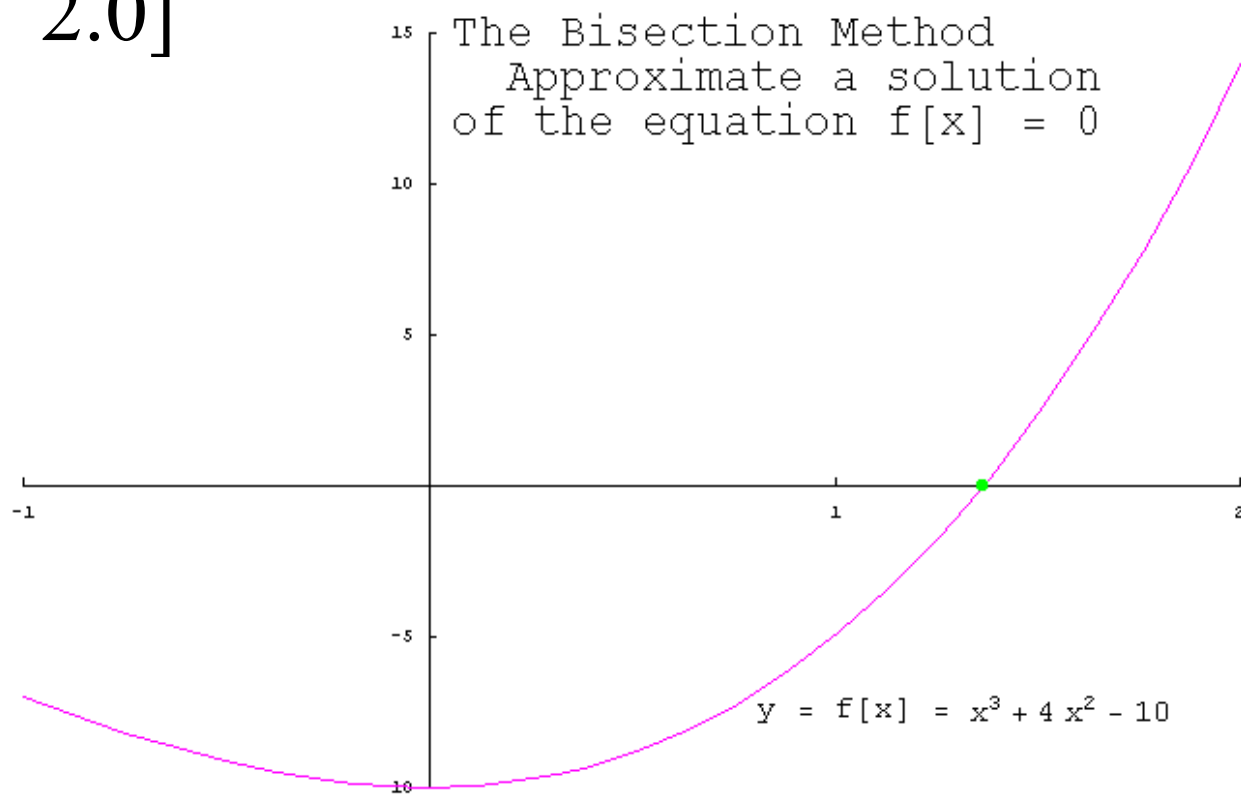


非线性方程求根——动画

- 求方程 $f(x)=x^3+4x^2-10$ 的根
 - 二分法 [0.5 2.0]
 - 试位法 [-1.0 2.0]

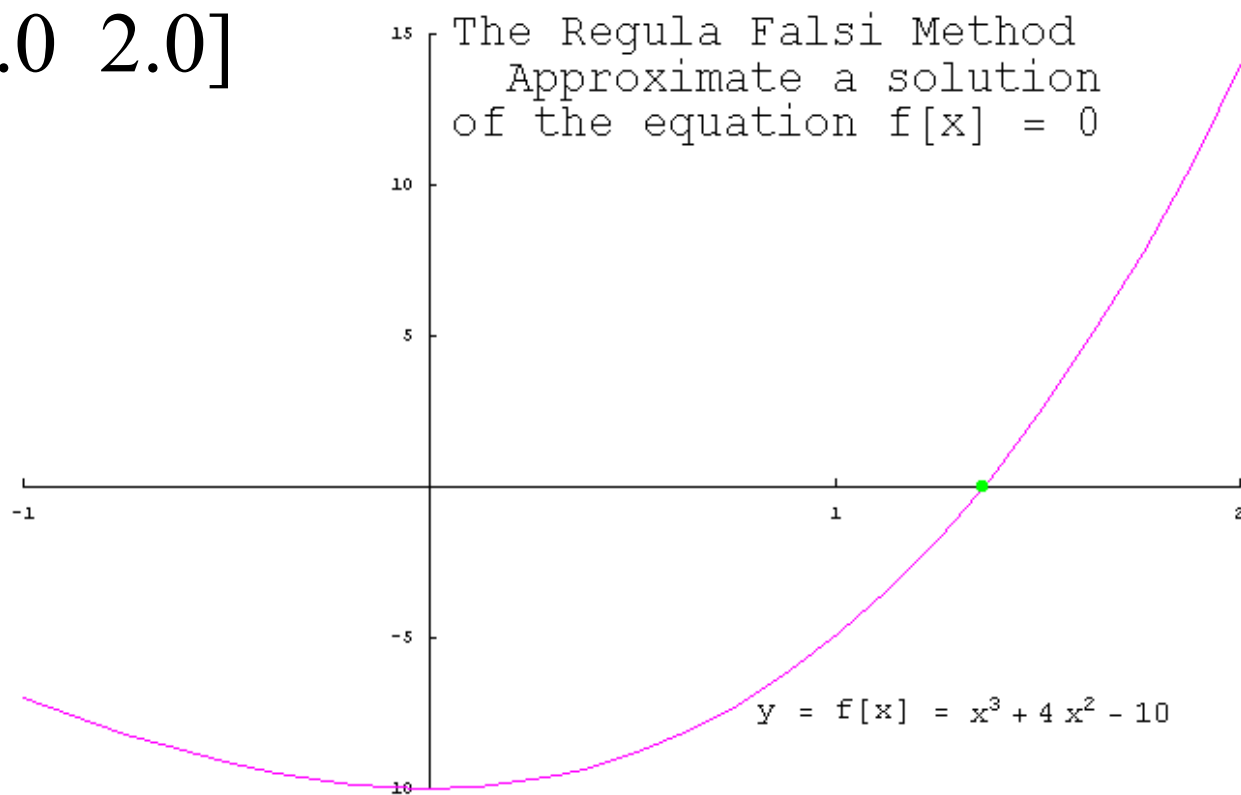
二分法

[0.5 2.0]



试位法

$[-1.0 \ 2.0]$



非线性方程求根

- 二分法
- 试位法
- 不动点迭代（简单定点迭代，Fixed-point Iteration）
- Newton-Raphson方法
- 割线法

不动点迭代

- 重组方程

$$f(x) = 0 \Rightarrow g(x) = x$$

$$x_k = g(x_{k-1}) \quad x_0 \text{ given, } k = 1, 2, \dots$$

$$\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100\%$$

- 定义：

- 函数 $g(x)$ 的一个不动点(Fixed point)是指一个实数 P ，满足 $P=g(P)$ 。
 - 从图形上看，函数 $y=g(x)$ 的不动点是 $y=g(x)$ 和 $y=x$ 的交点。
- 迭代 $p_{n+1}=g(p_n)$ ，其中 $n=0,1,2,\dots$ ，称为不动点迭代。

- 定理：

- 设 g 是一个连续函数，且 $\{p_n\}_{n=0}^{\infty}$ 是由不动点迭代生成的序列。如果 $\lim_{n \rightarrow \infty} p_n = P$ ，则 P 是 $g(x)$ 的一个不动点。
- 设函数 $g \in C[a, b]$ ，如果对于所有 $x \in [a, b]$ ，映射 $y=g(x)$ 的范围满足 $y \in [a, b]$ ，则函数 g 在 $[a, b]$ 内有一个不动点。此外，设 $g'(x)$ 定义在 (a, b) 内，且对于所有 $x \in (a, b)$ ，存在正常数 $K < 1$ ，使得 $|g'(x)| \leq K < 1$ ，则函数 g 在 $[a, b]$ 内有唯一的不动点 P 。

不动点迭代

- 迭代过程的收敛性？

方程

$$f(x) = x - \sin x - 0.5 = 0$$

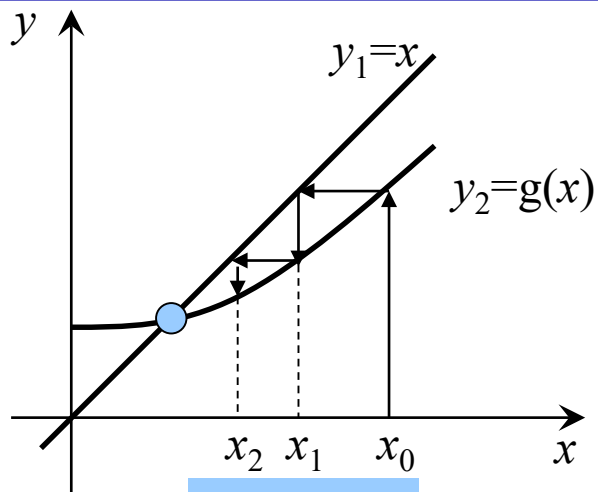
可以用不同方法转化为等价方程

$$(a) \quad x = \sin x + 0.5 \equiv g_1(x) \quad \text{收敛}$$

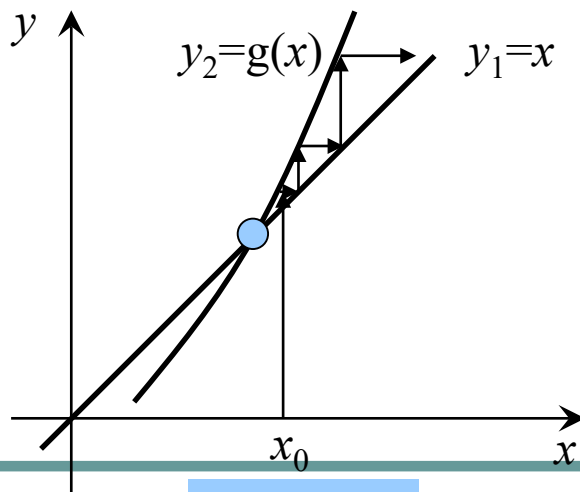
$$(b) \quad x = \sin^{-1}(x - 0.5) \equiv g_2(x) \quad \text{发散}$$

在由方程 $f(x)=0$ 转化为等价的方程 $x=g(x)$ 时，选择不同的迭代函数 $g(x)$ ，就会产生不同的序列，且这些序列的收敛情况也不会相同。因此，需要研究如何选取迭代函数使迭代过程收敛？

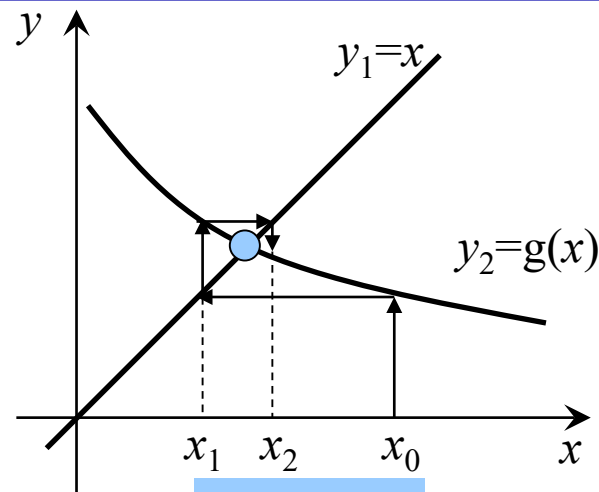
不动点迭代——图形解释



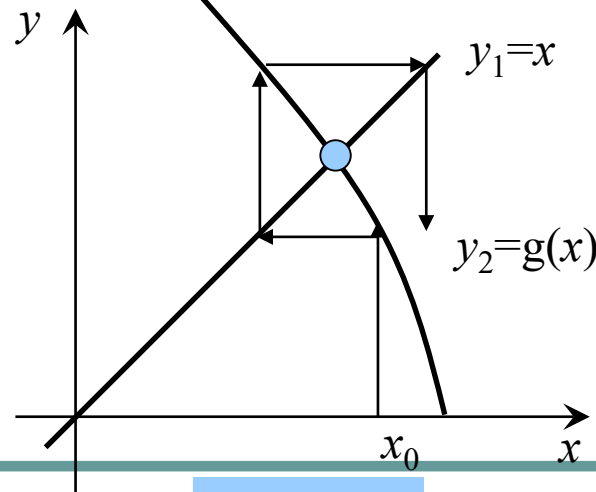
单调收敛



单调发散



振荡收敛



振荡发散

不动点迭代

- 定理（不动点定理）

- 设有(i) $g, g' \in C[a, b]$, (ii) K 是一个正常数, (iii) $p_0 \in (a, b)$, (iv)对于所有 $x \in [a, b]$, 有 $g(x) \in [a, b]$.
 - 如果对于所有 $x \in [a, b]$, 有 $|g'(x)| \leq K < 1$, 则迭代 $p_n = g(p_{n-1})$ 将收敛到唯一的不动点 $P \in [a, b]$ 。在这种情况下, P 称为吸引(attractive)不动点。
 - 如果对于所有 $x \in [a, b]$, 有 $|g'(x)| > 1$, 则迭代 $p_n = g(p_{n-1})$ 将不会收敛到 P 。在这种情况下, P 称为排斥(repelling)不动点, 而且迭代显示出局部发散性。

不动点迭代——例

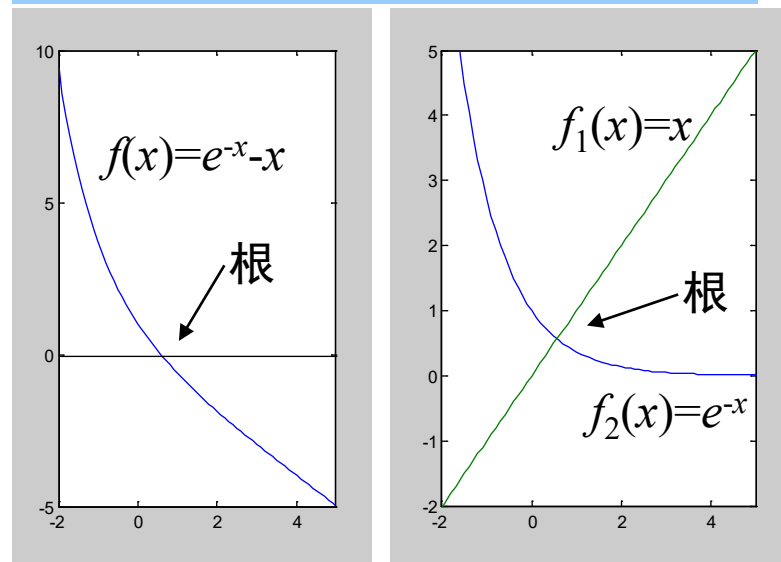
● 问题：使用不动点迭代法求解函数 $f(x)=e^{-x}-x$ 的根。

● 解： $x_{i+1}=e^{-x_i}$, $x_0=0$

i	x_i	ε_a	ε_t
0	0		100
1	1.000000	100	76.3
2	0.367879	171.8	35.1
3	0.692201	46.9	22.1
4	0.5040473	38.3	11.8
5	0.606244	17.4	6.89
6	0.545396	11.2	3.83
7	0.579612	5.90	2.20
8	0.560115	3.48	1.24
9	0.571143	1.93	0.705
10	0.564879	1.11	0.399

● 每次迭代使根的估计越来越接近真实根0.56714329

● 每次迭代的真实百分误差与上次迭代相比大致成比例关系，线性收敛



不动点迭代——参考程序

```
• function [k,p,err,P] = fixpt(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 that were carried out
%         - p is the approximation to the fixed-point
%         - err is the error in the approximation
%         - P' contains the sequence {pn}

P(1)= p0;
for k=2:max1
    P(k)=feval(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

if k == max1
    disp('maximum number of iterations exceeded')
end
P=P';
```

不动点迭代——收敛性

- 迭代方程 $x_{i+1} = g(x_i)$

- 真实解 $x_r = g(x_r)$

- 两个方程相减：

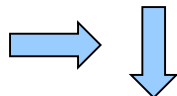
$$x_r - x_{i+1} = g(x_r) - g(x_i)$$

- 导数中值定理：

$$g'(\xi) = \frac{g(b) - g(a)}{b - a}$$

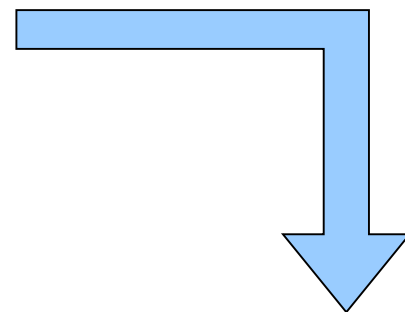
- 令

$$a = x_i \quad b = x_r$$



$$g(x_r) - g(x_i) = (x_r - x_i)g'(\xi)$$

定义第*i*次迭代的真实误差为 $E_{t,i} = x_r - x_i$



$$x_r - x_{i+1} = (x_r - x_i)g'(\xi)$$

$$E_{t,i+1} = g'(\xi)E_{t,i}$$

- 如果 $|g'(\xi)| > 1$ ，误差会增大
- 如果导数是正的，误差将是正的，迭代的解单调；如果导数是负的，误差振荡
- 方法收敛时，误差大致与前一次的迭代误差成比例，并且小于前一次的迭代误差（线性收敛）

非线性方程求根

- 二分法
- 试位法
- 不动点迭代
- **Newton-Raphson方法**
- 割线法

Newton-Raphson方法

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

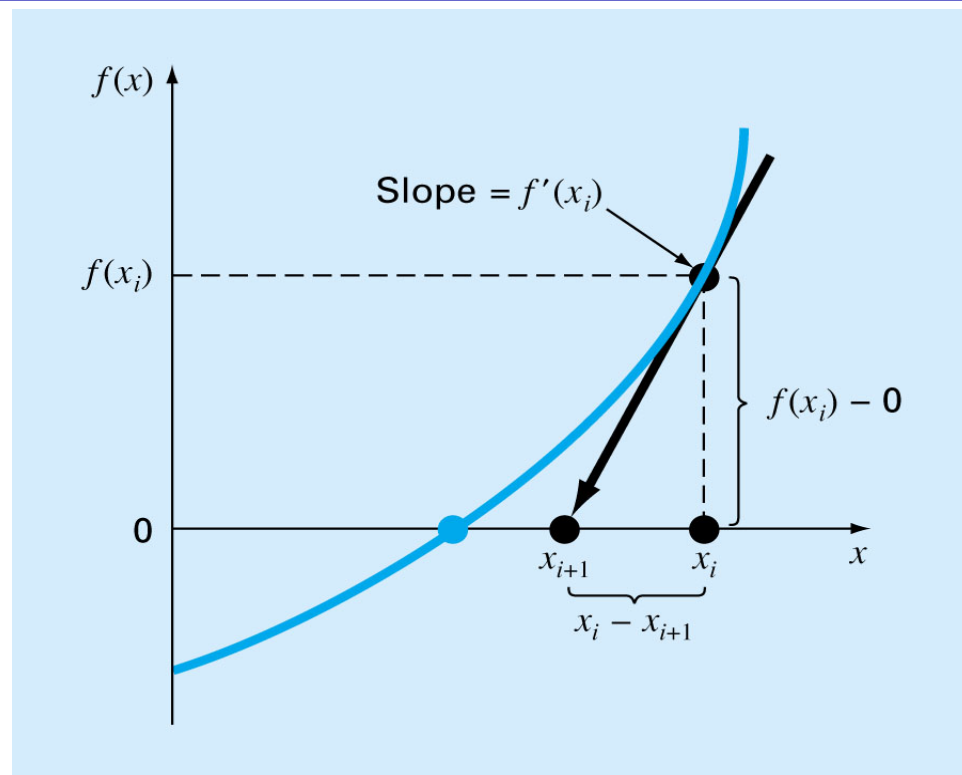
- $f(x)$ 与 x 轴的交点处, $f(x)=0$

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



寻找 $g(x) = x - \frac{f(x)}{f'(x)}$ 的不动点



Newton-Raphson方法——例

- 问题：使用Newton-Raphson方法求解函数 $f(x)=e^{-x}-x$ 的根，根的初始值为 $x_0=0$ 。
- 解：计算函数的一阶导数为 $f'(x)=-e^{-x}-1$
- 迭代公式为：

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

i	x_i	ε_i
0	0	100
1	0.5	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$<10^{-8}$

算法很快收敛到真实根，
每次迭代的真百分比相
对误差比不动点迭代法
减小快得多。

Newton-Raphson方法——终止条件和误差估计

- 终止条件:

$$\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100\%$$

- 误差分析:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(\xi)}{2!}(x_{i+1} - x_i)^2$$

$$x_{i+1} = x_r \quad f(x_r) = 0$$

$$0 = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

$$0 = f'(x_i)(x_r - x_{i+1}) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

假设方法收敛, x_i 和 ξ 都应该靠近根 x_r

$$E_{t,i+1} \approx \frac{-f''(x_r)}{2f'(x_r)} E_{t,i}^2$$

误差大致与前一次迭代的误差平方成正比。
(二次收敛)

收敛阶

- 定义

- 设序列 $\{x_n\}_{n=0}^{\infty}$ 收敛到 x_r ，并令 $E_n = x_r - x_n$ ， $n \geq 0$ 。如果存在两个常量 $A \neq 0$ 和 $R > 0$ ，并且

$$\lim_{n \rightarrow \infty} \frac{|x_r - x_{n+1}|}{|x_r - x_n|^R} = \lim_{n \rightarrow \infty} \frac{|E_{n+1}|}{|E_n|^R} = A$$

- 则称该序列以收敛阶 R 收敛到 x_r ， A 称为渐进误差常数。
- $R=1$ ，称为线性收敛
- $R=2$ ，称为二次收敛
- 一些序列的收敛阶不是整数。

Newton-Raphson方法——收敛速度

- 定理：

- 设Newton-Raphson迭代产生的序列 $\{x_n\}$ 收敛到函数 $f(x)$ 的根 x_r ，如果 x_r 是单根，则 $\{x_n\}$ 是二次收敛，而且对于足够大的 n 有

$$|E_{n+1}| \approx \frac{|f''(x_r)|}{2|f'(x_r)|} |E_n|^2$$

- 如果 x_r 是 M 阶多重根，则 $\{x_n\}$ 是线性收敛，而且对于足够大的 n 有

$$|E_{n+1}| \approx \frac{M-1}{M} |E_n|$$

Newton-Raphson方法——收敛速度

- 问题：方程 $f(x)=x^3-3x+2=0$ 的根为 $x_1=-2$, $x_{2,3}=1$ ，用Newton-Raphson方法分别从 $x_0=-2.4$ 和 $x_0=1.2$ 开始迭代，检查序列的收敛性。

在单根处

i	x_i	$ E_{t,i} $	$ E_{t,i} / E_{t,i-1} ^2$
0	-2.400000000	0.400000000	
1	-2.076190676	0.076190476	0.476190475
2	-2.003596011	0.003596011	0.619469086
3	-2.000008589	0.000008589	0.664202613
4	-2.000000000	0.000000000	

$A \approx 2/3$

Newton-Raphson方法——收敛速度

在二重根处

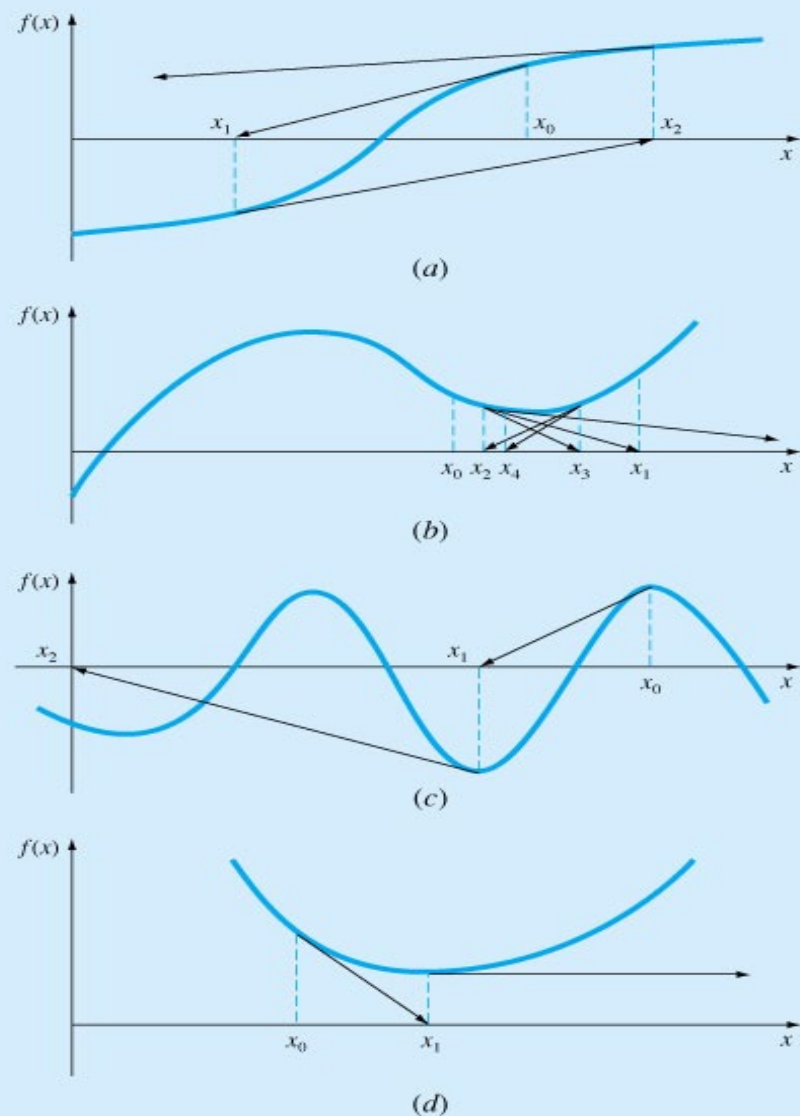
i	x_i	$ E_{t,i} $	$ E_{t,i} / E_{t,i-1} $
0	1.200000000	0.200000000	
1	1.103030303	0.103030303	0.515151515
2	1.052356420	0.052356420	0.508165253
3	1.026400811	0.026400811	0.496751115
4	1.013257730	0.013257730	0.509753688
5	1.006643419	0.006643419	0.501097775

$A \approx 1/2$

Newton-Raphson方法——缺点

- $f(x)$ 与 x 轴的交点处, $f(x)=0$
 - (a) 在根附近出现拐点, $f''(x)=0$, 迭代逐渐远离根;
 - (b) 在局部极值点附近振荡;
 - (c) 跳过了几个根;
 - (d) $f'(x)=0$, 被零除, 解沿水平方向永远不会和 x 轴相交;

收敛性依赖于函数的性质和初始猜测的准确度



Newton-Raphson方法——算法

- 画图
- 将最终的估计根代入原始函数中检查 $f(x)$ 是否为0，防止由于收敛很慢或振荡收敛导致很小的 ε_a 但根却离真实根仍然很远。
- 确定迭代上限，避免因振荡、慢收敛或发散的情况导致无限循环。
- 判断 $f'(x)=0$ 的概率并通知用户。

Newton-Raphson方法——参考程序

```
● function [p0,err,k,y]=newton(f,df,p0,delta,epsilon,max1)

%Input - f is the object function input as a string 'f'
%       - df is the derivative of f input as a string 'df'
%       - 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)

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

非线性方程求根

- 二分法
- 试位法
- 不动点迭代
- Newton-Raphson方法
- 割线法（正割法，Secant Method）

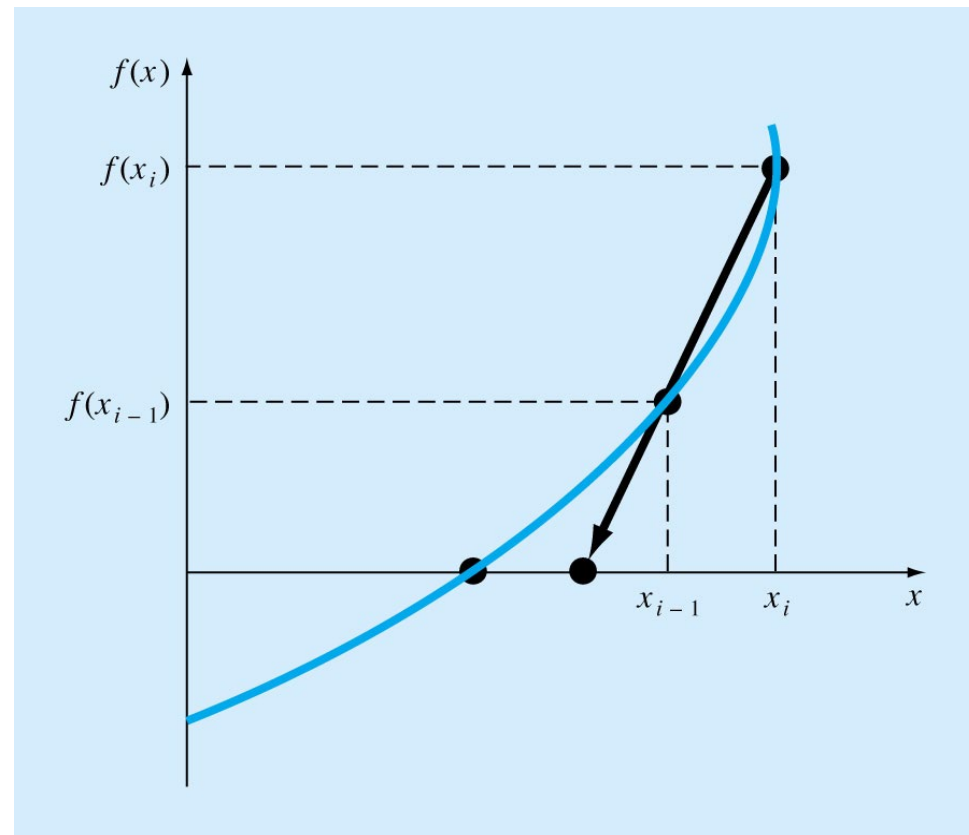
割线法（正割法）

$$\frac{1}{f'(x_i)} \cong \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$
$$x_{i+1} = x_i - f(x_i) \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \quad i = 1, 2, 3, \dots$$

- 类似于Newton-Raphson方法：用差商估计斜率

- 类似于试位法：都是使用两个初始估计值来计算函数斜率的近似，并将其投射到x轴上获得一个新的估计值。

- 收敛阶：1.618



割线法——例

- 问题：使用割线法求解函数 $f(x)=e^{-x}-x$ 的根，初始估计为 $x_{-1}=0$ 和 $x_0=1.0$ 。
- 解：

i	x_{i-2}	x_{i-1}	$f(x_{i-2})$	$f(x_{i-1})$	x_i	ε_i
1	0	1	1.00000	-0.63212	0.61270	8.0
2	1	0.61270	-0.63212	-0.07081	0.56384	0.58
3	0.61270	0.56384	-0.07081	0.00518	0.56717	0.0048

两个估计值都在根
的同边。

割线法——参考程序

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

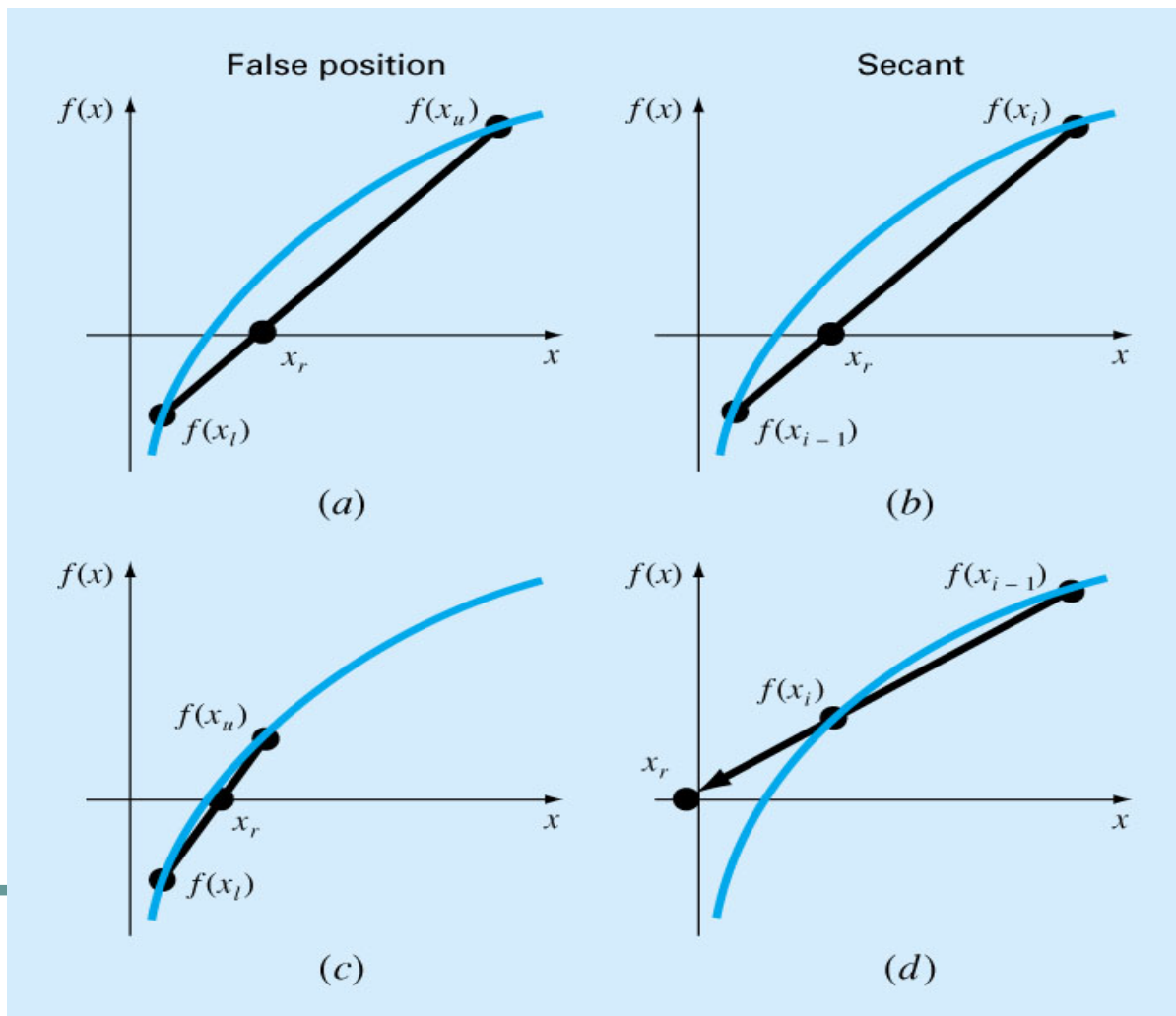
%Input - f is the object function input as a string 'f'
%       - p0 and p1 are the initial approximations to a zero of f
%       - delta is the tolerance for p1
%       - epsilon is the tolerance for the function values y
%       - max1 is the maximum number of iterations
%Output - p1 is the secant 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)

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

割线法——与试位法的比较

- 用两种方法估计 $f(x)=\ln x$ 的根。

对于所有现实情况，试位法通常是收敛的，但割线法可能会发散，如果割线法收敛，通常比试位法快。



割线法——改进

- 估计导数：通过独立变量的少量扰动来估计。

$$f'(x_i) \cong \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i}$$

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)}$$

- 求解函数 $f(x)=e^{-x}-x$ 的根：取 $\delta=0.01$

➤ 算法不会自动选择合适的 δ ，如果 δ 太小，可能会被分母的减性抵销引起的舍入误差淹没。

➤ 如果选择正确，当导数难以计算或两个初始值不易获取时，是一种好的替代算法。

i	x_{i-1}	$x_{i-1} + \delta x_{i-1}$	$f(x_{i-1})$	$f(x_{i-1} + \delta x_{i-1})$	x_i	ε_i
1	1	1.01	-0.63212	-0.64578	0.537263	5.3
2	0.537263	0.542635	0.047083	0.038579	0.56701	0.0236
3	0.56701	0.572680	0.000209	-0.00867	0.567143	2.365×10^{-5}

针对重根的改进

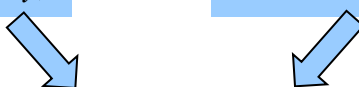
- 定义一个新函数

$$u(x) = \frac{f(x)}{f'(x)}$$

- 这个函数和原函数有相同的根
- 改进的Newton-Raphson方法

$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)}$$

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


$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

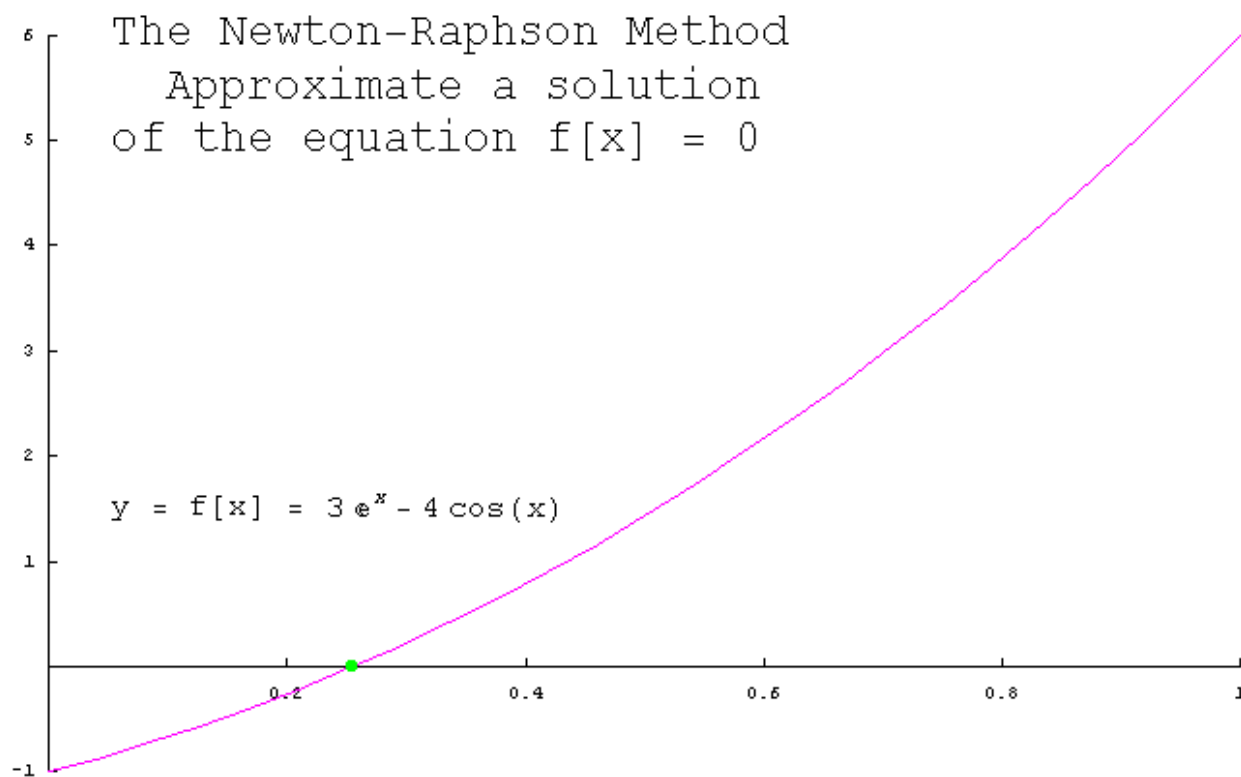
- 改进的割线法

$$x_{i+1} = x_i - u(x_i) \frac{x_i - x_{i-1}}{u(x_i) - u(x_{i-1})}$$

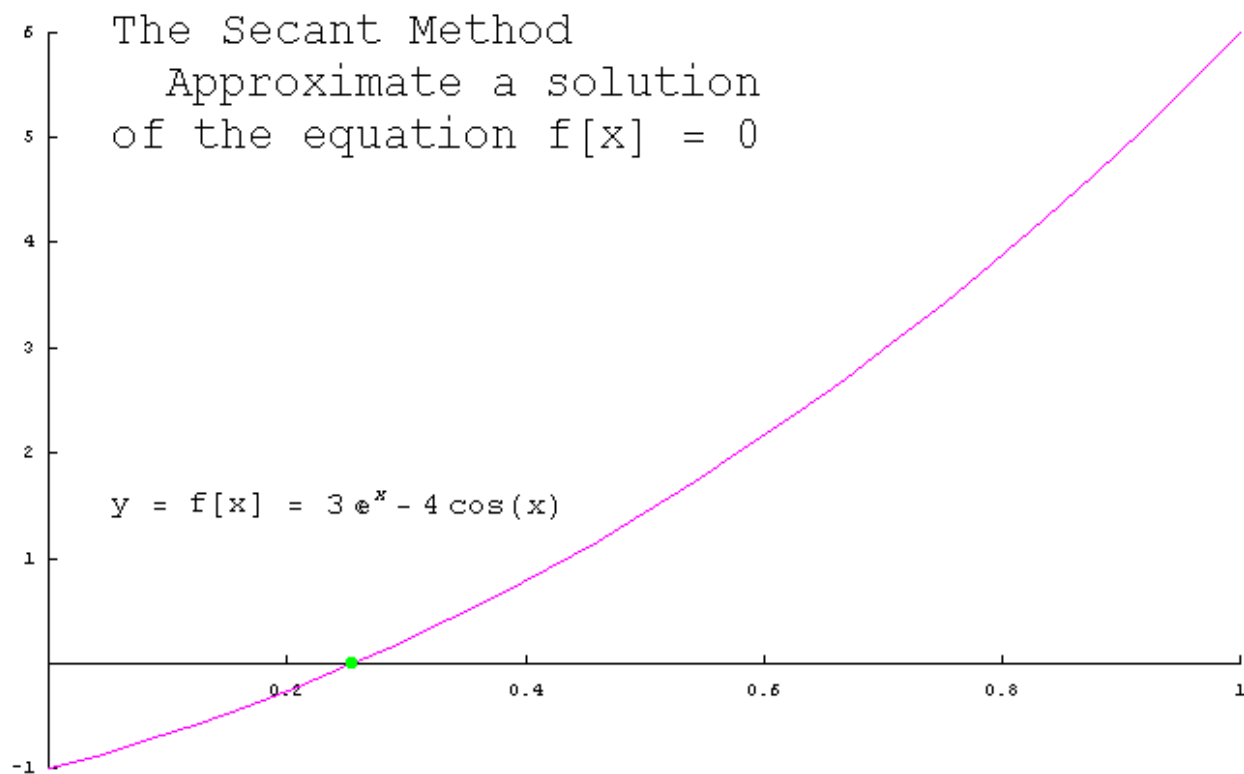
非线性方程求根——动画

- 求方程 $f(x)=3e^x-4\cos x$ 的根
 - Newton-Raphson方法 (1.0)
 - 割线法 (1.0, 0.9)

Newton-Raphson方法



割线法



扩展内容

- 方法的改进
- 非线性方程组求根
- 多项式求根

第二章 总结——各种方法

方法	初始估计	收敛速度	稳定性	精度	应用范围	编程难度	备注
直接法	—	—	—	—	受限		
图解法	—	—	—	差	求实数根	—	比数值法时间开销大
二分法	2	慢	通常收敛	好	求实数根	容易	
试位法	2	慢/中等	通常收敛	好	求实数根	容易	
不动点迭代	1	慢	可能发散	好	求一般根	容易	
Newton-Raphson	1	快	可能发散	好	求一般根	容易	需要求导
割线法	2	中等到快	可能发散	好	求一般根	容易	初始估计不用界定根

第二章 总结——重要内容

- 二分法
- 试位法
- 不动点迭代
- Newton-Raphson法
- 割线法
- 迭代公式
- 图解描述
- 误差和终止准则

MATLAB中的函数

函数	描述
fzero	单函数求根
roots	求多项式的根
poly	用已知的根构建多项式
polyval	求多项式的值
polyvalm	求带有矩阵变量的多项式的值