第二章 方程求根

问题的提出

（1）电磁波在圆柱波导中的传播，需要求$J'(x)=0$的根

（2）在光的衍射理论中，需要求$x-\tan x=0$的根

（3）在行星轨道（ planetary orbits）的计算中，对任意的a和b，需要求$x-a\sin x=b$的根

（4）在数学中，需要求n次多项式

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

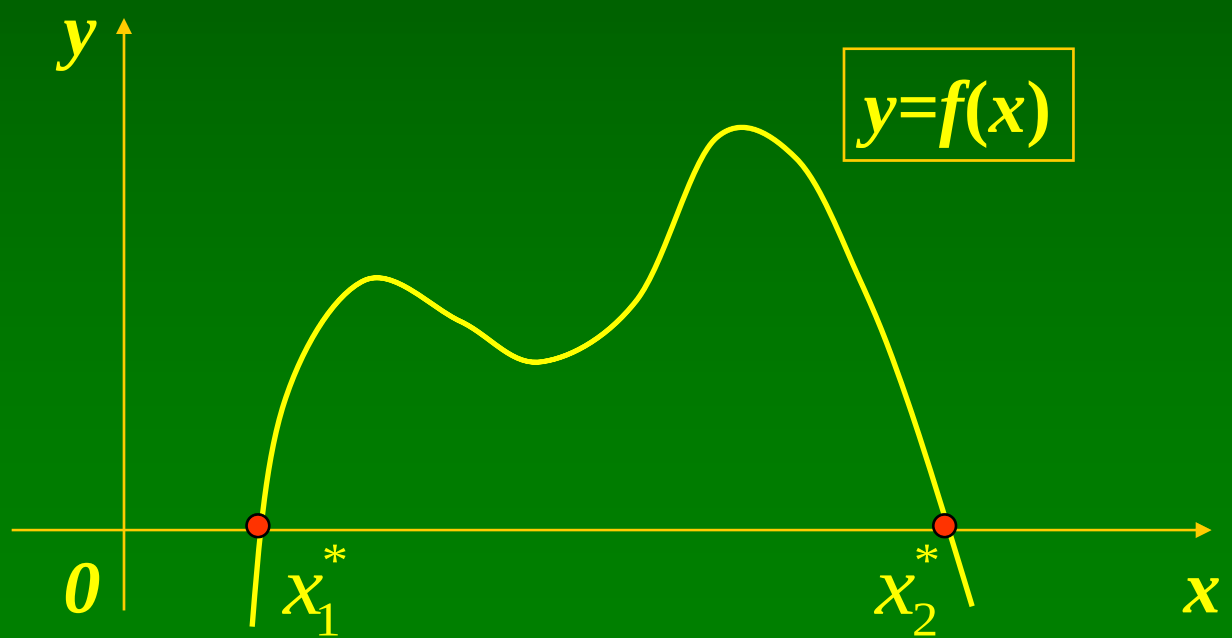
的根。

问题的提出

数学物理中的很多问题可以归结于解方程

$$f(x)=0$$

如果有x^*使得 $f(x^*)=0$，则称x^*为方程的**根**，或函数$f(x)$的**零点**。



- 公元前1700年古巴比伦已经提出了关于一次、二次方程的解法；
- 1535年意大利数学家TorTaglia发现3次方程解法；
- 1545年H Cardano公布该公式成为卡当算法；
- 卡当的学生Ferrari提出4次方程的解法；
- 1799年高斯证明*n*阶代数方程必然有*n*个解；
- 1824年挪威数学家N Abell发表“5次代数方程的解法不可能存在”
- 1830年法国数学家伽罗华再次提出，泊松表示看不懂；
- 1870年群论诞生。

问题的提出

$$f(x)=0$$

$$f(x)=3x^5-2x^4+8x^2-7x+1$$

$$f(x)=e^{2x}+1-x\ln(\sin x)-2$$

理论上已证明:

- **次数$n\leq4$的多项式方程,它的根可以用公式表示;**
- **次数$n\geq5$的多项式方程, 不能用解析表达式表示;**
- **超越方程 $\longrightarrow$ 一般无解析解;**

本章介绍各种求近似根的方法

非线性方程求根需要解决的问题

1．根的存在性。方程有没有根？如果有根，有几个根?

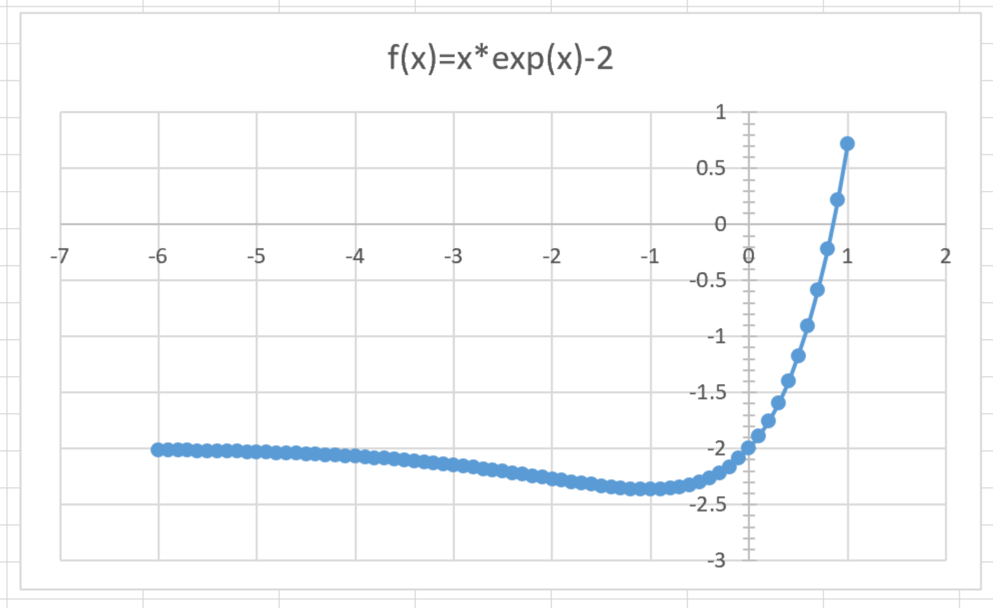
2．这些根大致在哪里？如何把根隔离开来?

3．根的精确化程度以及要求的求解速度。

判断根所在的区间

1．绘图法；

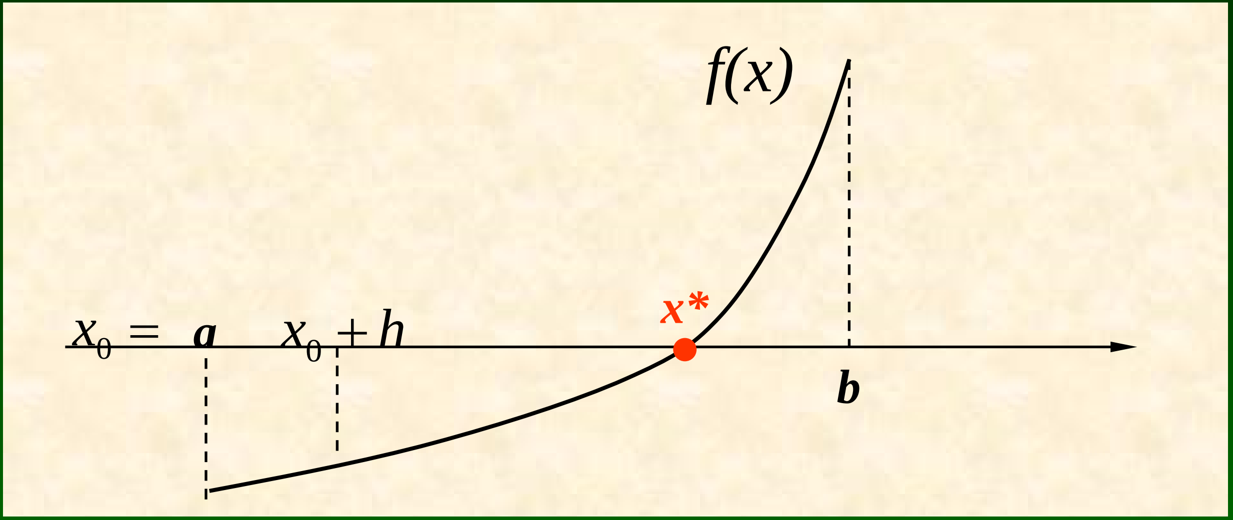
$$f(x) = xe^x - 2 = 0$$



2．逐步搜索法；

定理1：设函数 $f(x)$ 在区间[a, b]上连续，如果$f(a)\cdot f(b) < 0$，则方程 $f(x) = 0$ 在[a, b]内至少有一实根x^*。

2.2 逐步搜索法



从左端点$x = a$出发，按某个预先选定的步长h一步一步地向右跨，每跨一步都检验每步起点x_0和终点$x_0 + h$的函数值，若

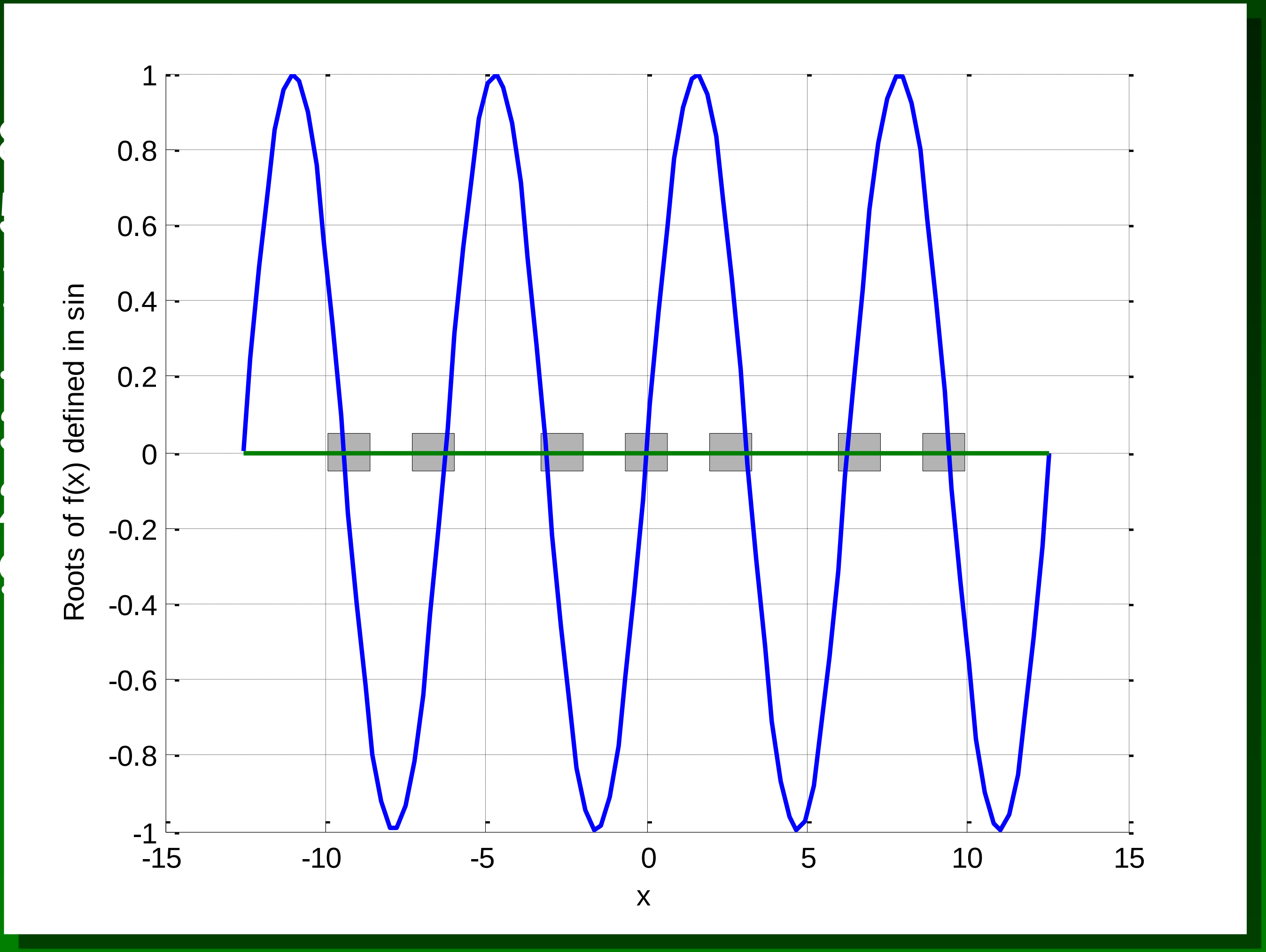
$$f(x_0)\cdot f(x_0+h)\le 0$$

那么所求的根x^*必在x_0与x_0+h之间，这里可取x_0或x_0+h作为根的初始近似。

分区间函数使用实例

```
>> Xb = brackPlot( 'sin' ,-4*pi,4*pi)
Xb =

-9.9208 -8
-7.2753 -5
-3.3069 -1
-0.6614 0.
1.9842 3.3
5.9525 7.2
8.5980 9.9
```



分区间函数

```
function Xb = brackPlot(fun,xmin,xmax,nx)
% brackPlot  Find subintervals on x that contain sign changes of f(x)
% Synopsis:  Xb = brackPlot(fun,xmin,xmax)
%            Xb = brackPlot(fun,xmin,xmax,nx)
% Input:     fun = (string) name of mfile function that evaluates f(x)
%  xmin,xmax = endpoints of interval to subdivide into brackets.
%   nx = (optional) number of samples along x axis used to test for
%   brackets.  The interval xmin <= x <= xmax is divided into
%   nx-1 subintervals.  Default:  nx = 20.
% Output:    Xb = two column matrix of bracket limits.  Xb(k,1) is the
% left(lower x value) bracket and Xb(k,2) is the right bracket for
% the k^th potential root.  If no brackets are found, Xb = [].
```

分区间函数

```
if nargin<4, nx=20; end
% --- Create data for a plot of f(x) on interval xmin <= x <= xmax
xp = linspace(xmin,xmax);  yp = feval(fun,xp);
% --- Save data used to draw boxes that indicate brackets
ytop = max(yp);  ybot = min(yp);          %  y coordinates of the box
ybox = 0.05*[ybot ytop ytop ybot ybot];  %  around a bracket
c = [0.7 0.7 0.7];                        %  RGB color used to fill the box

% --- Begin search for brackets
x = linspace(xmin,xmax,nx);    %  Vector of potential bracket limits
f = feval(fun,x);              %  Vector of f(x) values at potential brackets
nb = 0;  Xb = [];              %  Xb is null unless brackets are found
```

分区间函数

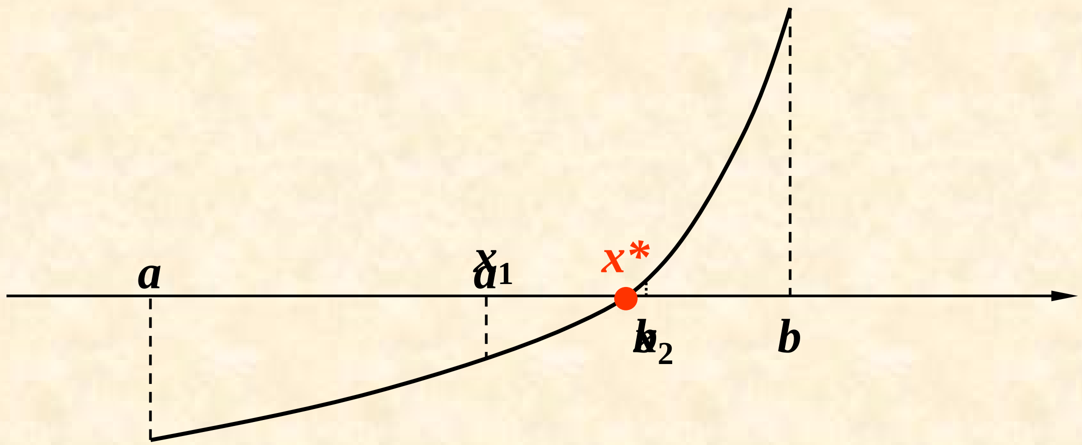
```
for k = 1:length(f)-1
  if sign(f(k))~=sign(f(k+1))
%  True if sign of f(x) changes in the interval
    nb = nb + 1;
    Xb(nb,:) = [x(k) x(k+1)];  %  Save left and right ends of the bracket
    hold on;  fill([x(k) x(k) x(k+1) x(k+1) x(k)],ybox,c); %  Add filled
box
  end
end
if isempty(Xb)   %  Free advice
 warning('No brackets found.  Check [xmin,xmax] or increase nx');
 return;         % return without drawing a plot
end
```

分区间函数

```
% --- Add plot here so that curve is on top of boxes used to indicate brackets
plot(xp,yp,[xmin xmax]);
grid on;  xlabel('x');
if isa(fun,'inline')
  ylabel(sprintf('Roots of f(x) = %s',formula(fun)));  %  label is formul in f(x)
else
  ylabel(sprintf('Roots of f(x) defined in %s',fun));  %  label is name of m-file
end
hold off
```

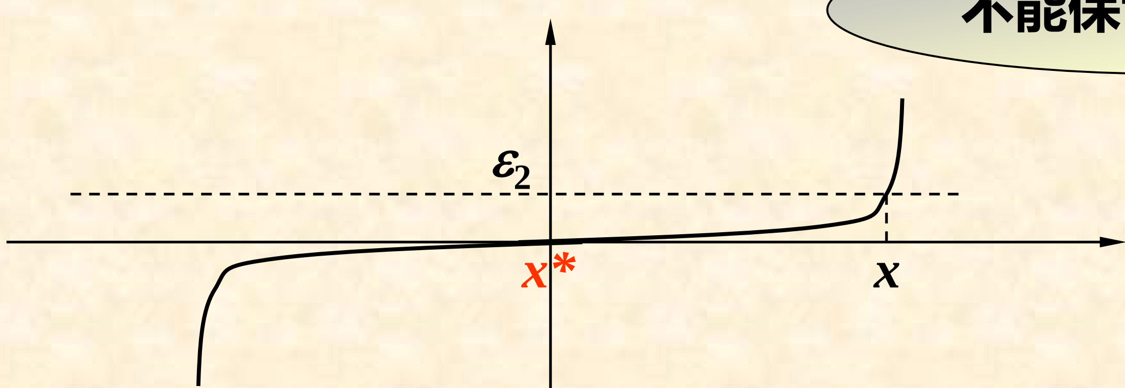
§2.3 二 分 法

问题：分区间法能够给出根的范围，能否利用这一方法缩减范围?

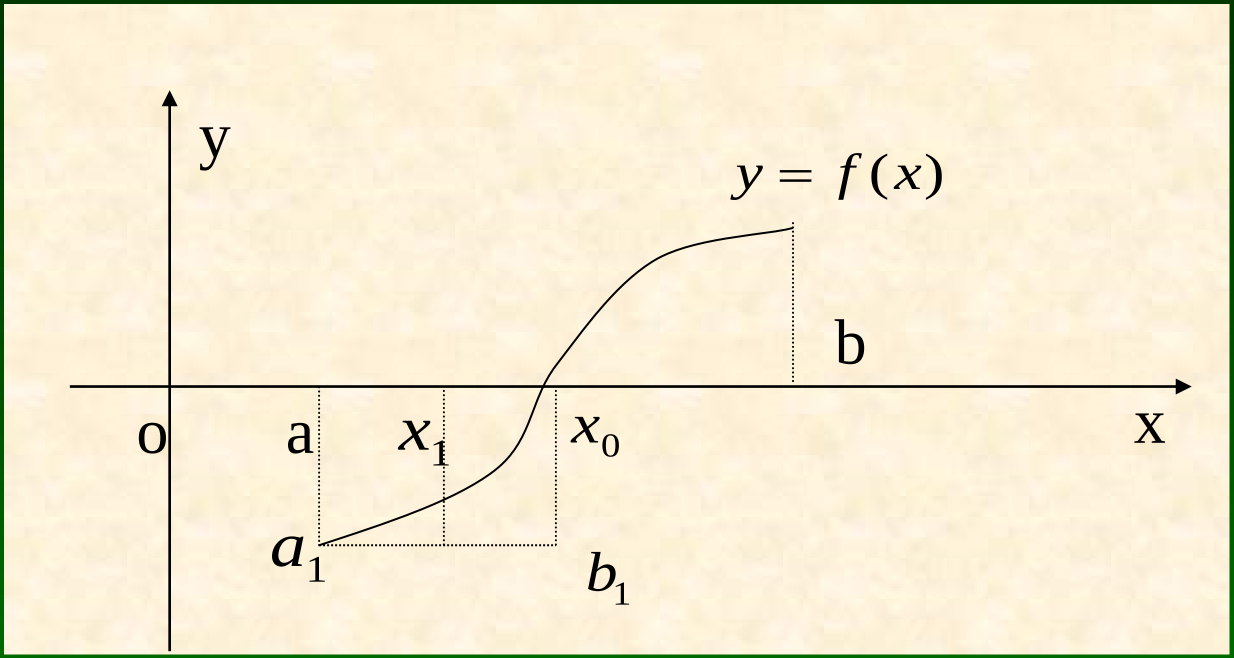


$$\left|x_{k+1}-x_k\right|<\varepsilon_1 \quad \text{或} \quad \left|f(x)\right|<\varepsilon_2$$

不能保证 x 的精度



二分法误差估计



若取 $x_n = \dfrac{b_n + a_n}{2}$ 作为根x^*的近似值，则误差为

$$\left| x^* - x_n \right| \le \frac{b_n - a_n}{2} = \frac{b - a}{2^{n+1}} \le \varepsilon$$

给定误差范围可计算出需要多少步：$n \ge \dfrac{\ln(b-a) - \ln(\varepsilon)}{\ln 2} - 1$

二分法

例4. 用二分法求方程$f(x)=x^3-x^2-2x+1=0$在区间[0,1]内的一个实根，要求有三位有效数字。

解：因为$f(0)=1>0$，$f(1)=-1<0$，且在区间[0,1]内，$f'(x)=3(x-1/3)^2-7/3<0$，有且只有一个实根。由

$$\frac{1}{2^{k+1}}(1-0)\le\frac{1}{2}\times10^{-3}$$

3位有效数字

解得 $k\ge\dfrac{3\ln10}{\ln2}\ge9.965$

故至少需要二分10次，才能满足解的精度要求。

```
function r = bisect(fun,xb,xtol,ftol,verbose)
% bisect  Use bisection to find a root of the scalar equation f(x) = 0
%% Synopsis:  r = bisect(fun,xb)
%           r = bisect(fun,xb,xtol)
%           r = bisect(fun,xb,xtol,ftol)
%           r = bisect(fun,xb,xtol,ftol,verbose)
%% Input: fun     = (string) name of function for which roots are sought
%       xb      = vector of bracket endpoints. xleft = xb(1), xright = xb(2)
%       xtol    = (optional) relative x tolerance.    Default:  xtol=5*eps
%       ftol    = (optional) relative f(x) tolerance. Default:  ftol=5*eps
%       verbose = (optional) print switch. Default: verbose=0, no printing
%% Output:  r = root (or singularity) of the function in xb(1) <= x <= xb(2)
if size(xb,1)>1, warning('Only first row of xb is used as bracket');  end
if nargin < 3,  xtol = 5*eps;  end
if nargin < 4,  ftol = 5*eps;  end
if nargin < 5,  verbose = 0;   end
```

```
xeps = max(xtol,5*eps);         %  Smallest tolerances are 5*eps
feps = max(ftol,5*eps);
a = xb(1,1); b = xb(1,2);       %  Use first row if xb is a matrix
xref = abs(b - a);              %  Use initial bracket in convergence test
fa = feval(fun,a);   fb = feval(fun,b);
fref = max([abs(fa) abs(fb)]);  %  Use max f in convergence test
if sign(fa)==sign(fb)           %  Verify sign change in the interval
  error(sprintf('Root not bracketed by [%f, %f]',a,b));
end

if verbose
  fprintf('\nBisection iterations for %s.m\n',fun);
  fprintf('   k       xm            fm\n');
end
```

```
k = 0;  maxit = 50;      %  Current and max number of iterations
while k < maxit
  k = k + 1;
  dx = b - a;
  xm = a + 0.5*dx;       %  Minimize roundoff in computing the midpoint
  fm = feval(fun,xm);
  if verbose, fprintf('%4d  %12.4e  %12.4e\n',k,xm,fm); end

  if (abs(fm)/fref < feps) | (abs(dx)/xref < xeps) % True when root is found
    r = xm;  return;
  end
  if sign(fm)==sign(fa)
    a = xm;  fa = fm;    %  Root lies in interval [xm,b], replace a and fa
  else
    b = xm;  fb = fm;    %  Root lies in interval [a,xm], replace b and fb
  end
end
warning(sprintf('root not within tolerance after %d iterations\n',k));
```

例2：　求方程

$$f(x) = x^3 - x - 1 = 0$$

k	a_k	b_k	x_k	$f(x_k)$的符号
0	1	1.5	1.25	-
1	1.25	1.5	1.375	+
2	1.25	1.375	1.3125	-
3	1.3125	1.375	1.3438	+
4	1.3125	1.3438	1.3281	+
5	1.3125	1.3281	1.3203	-
6	1.3203	1.3281	1.3242	-

§2.3 二 分 法

①简单，不论求根区间有多大;

② 对$f(x)$ 要求不高.

①无法求重根

② 收敛慢

2.4 简单迭代法(基本迭代法)

将非线性方程 $f(x)=0$ 化为一个同解方程

$$x = \varphi(x) \qquad \text{---------(2)}$$

并且假设 $\varphi(x)$ 为连续函数

任取一个初值 x_0，代入(2)的右端，得

$$x_1 = \varphi(x_0)$$

继续 $$x_2 = \varphi(x_1)$$

$$\cdots \quad \cdots \quad \cdots$$

$$x_{k+1} = \varphi(x_k) \qquad (k = 0,1,2,\cdots) \qquad \text{---------(3)}$$

称上述过程为求解非线性方程的简单迭代法

例1. 用迭代法求解方程$2x^3 - x - 1 = 0$

解: (1) 将原方程化为等价方程

$$x = 2x^3 - 1$$

如果取初值$x_0 = 0$,由迭代法(3),得

$$x_0 = 0$$

$$x_1 = 2x_0^3 - 1 = -1$$

$$x_2 = 2x_1^3 - 1 = -3$$

$$x_3 = 2x_2^3 - 1 = -55$$

$$\cdots\cdots\cdots$$

显然迭代法发散!

(2) 如果将原方程化为等价方程

$$x = \sqrt[3]{\frac{x+1}{2}}$$

仍取初值 $x_0 = 0$

$$x_1 = \sqrt[3]{\frac{x_0 + 1}{2}} = \sqrt[3]{\frac{1}{2}} \approx 0.7937$$

$$x_2 = \sqrt[3]{\frac{x_1 + 1}{2}} = \sqrt[3]{\frac{1.7937}{2}} \approx 0.9644$$

依此类推,得

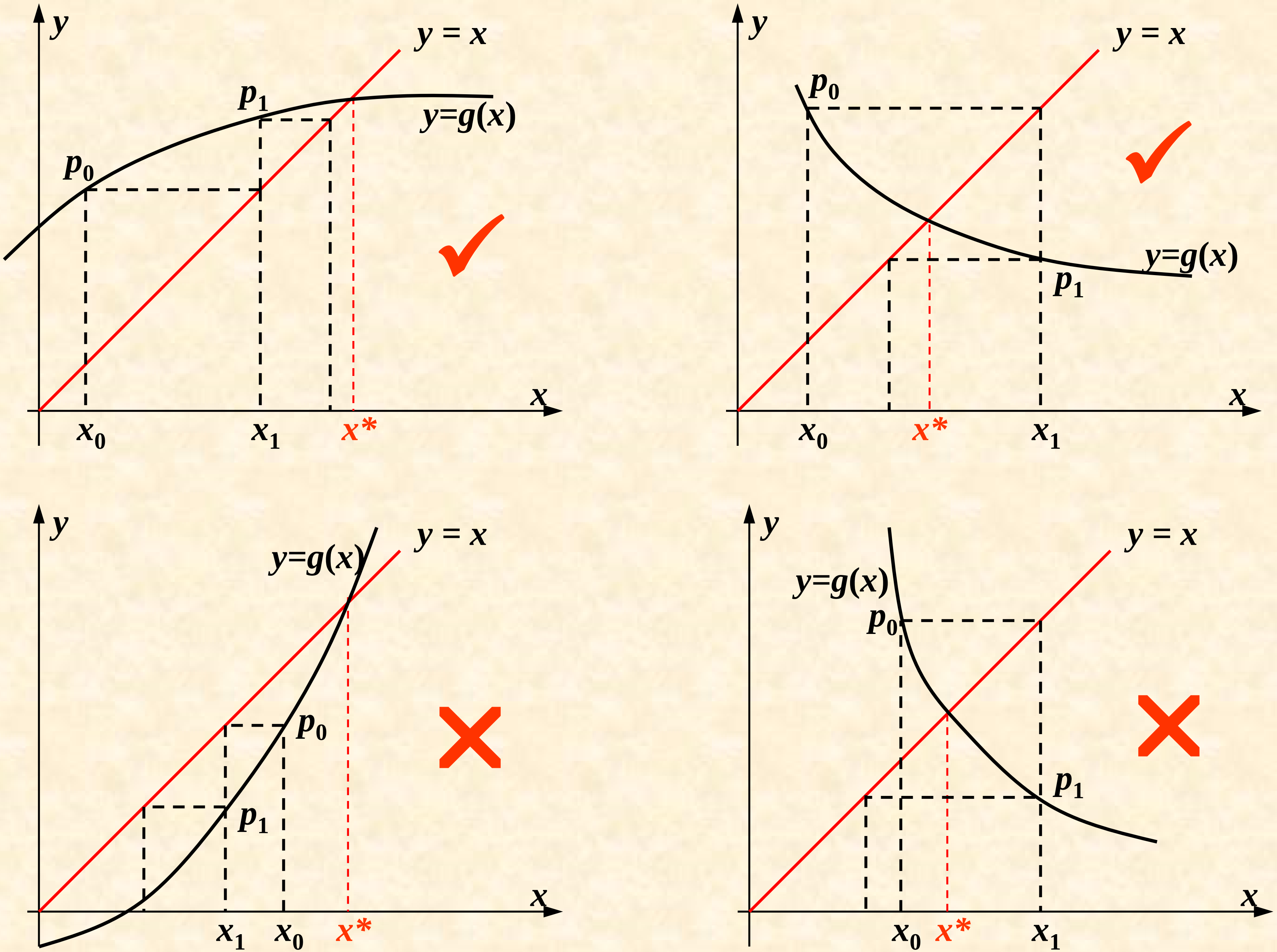
x2 = 0.9644
x3 = 0.9940
x4 = 0.9990
x5 = 0.9998
x6 = 1.0000
x7 = 1.0000

同样的方程不同的迭代格式有不同的结果。

已经收敛,故原方程的解为

$$x = 1.0000$$

什么形式的迭代法能够收敛呢?


y
y = x
y=g(x)
p0
p1
x
x0
x1
x*
y
y = x
p0
p1
y=g(x)
x
x0
x*
x1
y
y = x
y=g(x)
p0
p1
x
x1
x0
x*
y
y = x
y=g(x)
p0
p1
x
x0
x*
x1

定理一：假定函数$\varphi(x)$**满足下列条件：**

1、对任意$x \in [a,b]$**有**

$$a \le \varphi(x) \le b \text{；} \quad \mathbf{(1.1)}$$

2、存在正数 L<1，使对任意 $x_1, x_2 \in [a,b]$ **有**

$$|\varphi(x_1) - \varphi(x_2)| \le L|x_1 - x_2| \qquad 0 \le L < 1 \qquad \mathbf{(1.2)}$$

则方程在[a,b]内有唯一的根*x***，且对任何初值**$x_0 \in [a,b]$**,迭代序列** $x_{k+1} = \varphi(x_k)$ **均收敛于*x*****，且有如下的误差估计式：**

$$\left|x_k - x^*\right| \le \frac{L^k}{1-L}\left|x_1 - x_0\right| \qquad \mathbf{(1.3)}$$

定理1证明

根据条件2给出的不等式可知函数 $\varphi(x)$

在区间[a,b]上连续。

令 $f(x)=x-\varphi(x)$ 则 $f(x)$在[a,b]上也必然连续。

且有 … ≥ 0

矛盾！

由此可知满足上述条件的方程

在[a,b]范围内有唯一解。

根据连… $[a,b]$

使得 … $x=\varphi(x)$

在[a,b]内有解。

设x_1*，x_2*均为方程的解，则根据条件2可得

$$\left|x_1^*-x_2^*\right|=\left|\varphi(x_1^*)-\varphi(x_2^*)\right|\leq L\left|x_1^*-x_2^*\right|<\left|x_1^*-x_2^*\right|$$

定理1证明续

对任意的 $x_0 \in [a,b]$ **根据迭代公式** $x_{k+1} = \varphi(x_k)$

可知 $|x_{k+1} - x^*| = |\varphi(x_k) - \varphi(x^*)| \le L|x_k - x^*|$

据此反复递推有:

L<1

$$\left|x_k - x^*\right| \le L^k \left|x_0 - x^*\right|$$

故当 $k \to \infty$ **时迭代值** $x_k \to x^*$

即对任意的初值 $x_0 \in [a,b]$ **迭代序列{x_n}均收敛到方程的根*x****

定理1证明续

同理可知 $|x_{k+1}-x_k| = |\varphi(x_k)-\varphi(x_{k-1})| \le L|x_k - x_{k-1}|$

$$\Longrightarrow \quad |x_{k+1}-x_k| \le L^k |x_1 - x_0|$$

于是对任意正整数p有

$$|x_{k+p}-x_k| \le |x_{k+p}-x_{k+p-1}| + |x_{k+p-1}-x_{k+p-2}| + \cdots + |x_{k+1}-x_k|$$

$$\le (L^{p-1}+L^{p-2}+\cdots+1)|x_{k+1}-x_k|$$

$$\le \frac{(1-\ \ }{1-\ \ } \ \ \le \frac{(1-L^p)}{1-L}$$

误差事后估计

误差事前估计

令 $p \to \infty$

则 $|x^*-x_k| \le \dfrac{1}{1-L}|x_{k+1}-x_k| \le \dfrac{L^k}{1-L}|x_1-x_0|$

定理2. 设迭代函数$\varphi(x)$在$[a,b]$上连续,且满足

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

(2) 存在一正数L,满足$0<L<1$,且$\forall x\in[a,b]$,有

$$|\varphi'(x)|\le L \qquad \text{---------(5)}$$

则1^o. 方程$x=\varphi(x)$在$[a,b]$内有唯一解x^*

2^o.对于任意初值$x_0\in[a,b]$,迭代法$x_{k+1}=\varphi(x_k)$均收敛于x^*

$$3^o.\ |x_k-x^*|\le\frac{L}{1-L}|x_k-x_{k-1}| \qquad \text{---------(6)}$$

$$4^o.\ |x_k-x^*|\le\frac{L^k}{1-L}|x_1-x_0| \qquad \text{---------(7)}$$

$$5^o.\ \lim_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(x^*) \qquad \text{---------(8)}$$

证: 设$f(x)=x-\varphi(x)$, 则$f(x)$在$[a,b]$上连续可导

由条件(1) $f(a)=a-\varphi(a)\le 0$

$f(b)=b-\varphi(b)\ge 0$

由根的存在定理, 方程$f(x)=0$在$[a,b]$上至少有一个根

由 $|\varphi'(x)|\le L<1$

$$f'(x)=1-\varphi'(x)>0$$

则$f(x)$在$[a,b]$上单调递增, $f(x)=0$在$[a,b]$上仅有一个根

所以 1^o. 方程$x=\varphi(x)$在$[a,b]$内有唯一解x^*

2^o. 对于迭代法$x_{k+1}=\varphi(x_k)$，　由微分中值定理

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

$$x_{k+1}-x_k=\varphi(x_k)-\varphi(x_{k-1})=\varphi'(\bar{\xi})(x_k-x_{k-1})$$

由于 $|\varphi'(x)|\le L$

$$|x_{k+1}-x_k|\le L|x_k-x_{k-1}|$$

$$|x_{k+1}-x^*|\le L|x_k-x^*|=L|x_{k+1}-x^*-($$

$$\le L|x_{k+1}-x^*|+$$

误差事后估计

$$|x_{k+1}-x^*|\le\frac{L}{1-L}|x_{k+1}-x_k|$$

$$|x_k - x^*| \le \frac{L}{1-L}|x_k - x_{k-1}|$$

$$\le \frac{L^2}{1-L}|x_{k-1} - x_{k-2}|$$

$$\cdots \quad \cdots \quad \cdots$$

$$\le \frac{L^k}{1-L}|x_1 - x_0|$$

误差事前估计

由于 $L<1$, $\lim\limits_{k\to\infty}(x_k - x^*) = 0$

因此对任意初值 x_0, 迭代法 $x_{k+1} = \varphi(x_k)$ 均收敛于 x^*

$$|x_k - x^*| \le \frac{L}{1-L}|x_k - x_{k-1}| \le \frac{L^k}{1-L}|x_1 - x_0| \quad \text{证毕.}$$

对于预先给定的误差限ε 即要求$|x_k - x^*| < \varepsilon$

由(6)式,只要

$$\frac{L}{1-L}|x_k - x_{k-1}| < \varepsilon$$

因此,当 $|x_k - x_{k-1}| < \frac{1-L}{L}\varepsilon$ --------(8)

迭代就可以终止, x_k可以作为方程的近似解

$$f(x) = x^2 - 2x - 3 = 0$$

例1. 在区间[2,4]上考虑如下2个迭代格式的收敛性：

(1) $x_{k+1} = \sqrt{2x_k + 3} \qquad k = 0,1,2,\ldots$

(2) $x_{k+1} = \frac{1}{2}\left(x_k^2 - 3\right) \qquad k = 0,1,2,\ldots$

解 **(1)** $\varphi(x) = \sqrt{2x+3}, \qquad \varphi'(x) = \frac{1}{\sqrt{2x+3}}$

$$\left.\begin{aligned}&\varphi(x) \in \left[\varphi(2), \varphi(4)\right] = \left[\sqrt{7}, \sqrt{11}\right] \subset [2,4] \\ &\left|\varphi'(x)\right| \le \left|\varphi'(2)\right| \le \frac{1}{\sqrt{7}} < 1\end{aligned}\right\} \Longrightarrow \text{收敛}$$

(2) $\varphi(x) = \frac{1}{2}\left(x^2 - 3\right), \qquad \varphi'(x) = x$

当 $x \in [2,4]$ 时，$\left|\varphi'(x)\right| \ge 2 > 1 \Longrightarrow$ 发散

例2. 在区间[0,1]内用迭代法求方程$f(x)=x(x+1)^2-1=0$的一个实根，精确至4位有效数字。

解 方程等价形式为 $x=\dfrac{1}{(x+1)^2}$

即 $\varphi(x)=\dfrac{1}{(x+1)^2}$ $\quad \varphi'(x)=-\dfrac{2}{(x+1)^3}$

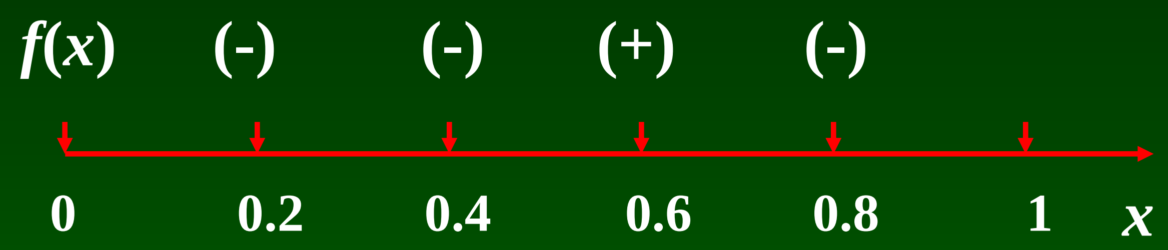
在区间[0,1]内， $\varphi(x)\in\left[\varphi(1),\varphi(0)\right]=[0.25,1]\subset[0,1]$

满足定理2.1的条件1。

而$\left|\varphi'(0)\right|=2>1$，不满足定理2.1的条件2。

$\Longrightarrow$ 在区间[0,1]上不能直接进行迭代求解？

将区间细分：



所以方程在区间[0.4,0.6]内有一个实根。在区间[0.4,0.6]上

$$\left|\varphi'(x)\right| \le \left|\varphi'(0.4)\right| = 0.7289 < 1$$

满足定理2.1的条件2。

但是 $\varphi(x) \in \left[\varphi(0.6), \varphi(0.4)\right] = [0.3906, 0.5102] \not\subset [0.4, 0.6]$

不满足定理2.1的条件1。

在区间[0.4,0.6]上也不能直接进行迭代求解？

再将区间细分，考虑区间[0.4,0.55]

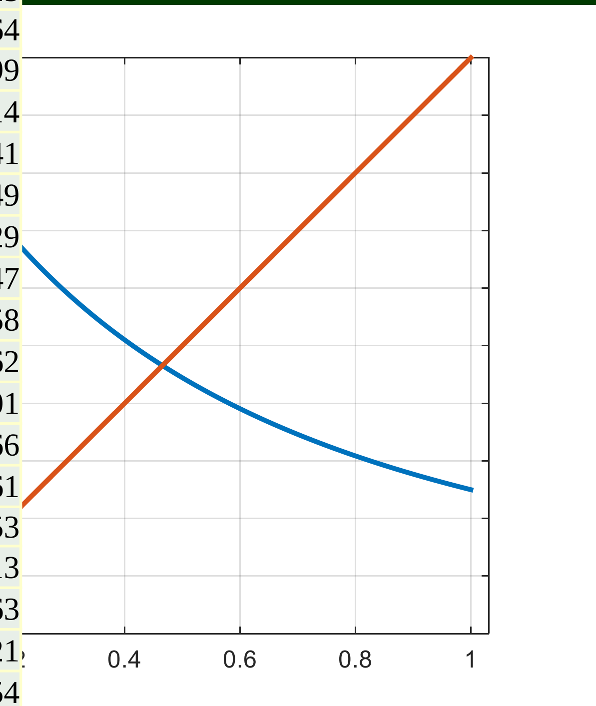
$$\left.\begin{array}{l}\varphi(x)\in\left[\varphi(0.55),\varphi(0.4)\right]=\left[0.4612,0.5102\right]\subset\left[0.4,0.55\right]\\ \left|\varphi'(x)\right|\le\left|\varphi'(0.4)\right|=0.7289<1\end{array}\right\}$$

⇒ **满足定理2.1。**

⇒ 对任意 $x_0\in\left[0.4,0.55\right]$，迭代格式

$$x_{k+1}=\frac{1}{\left(x_k+1\right)^2}$$ 收敛

0
1
0.25
0.64
0.371802499
0.531394214
0.426408641
0.491487049
0.449532429
0.475931147
0.459058258
0.469736962
0.462935801
0.467250166
0.464506361
0.46624853
0.465141213
0.465844563
0.465397621
0.465681554
0.465501147
0.465615763
0.465542941
0.465589207
0.465559812
0.465578488
0.465566622
0.465574161
0.465569371
0.465572414



1

k	x_k
0	0.4
1	0.510204
2	0.438459
3	0.483287
4	0.454516
5	0.472675
6	0.461090
…	…
17	0.465602
18	0.465552
19	0.465584
20	0.465563

$$|x_{18} - x_{17}| = 0.5 \times 10^{-4}$$

$$\Rightarrow \quad x^* \approx 0.4656$$

定理2.2 设方程 $x=\varphi(x)$ 在区间[*a*,*b*]内有根*x**，且当 $x\in[a,b]$ 时，$|\varphi'(x)|\geq 1$，则对任意初始值 $x_0\in[a,b]$ 且 $x_0\neq x^*$，迭代公式发散。

证　由 $x_0\in[a,b]$ 知

$$\left|x_1-x^*\right|=\left|\varphi(x_0)-\varphi(x^*)\right|=\left|\varphi'(\zeta_0)(x_0-x^*)\right|\geq\left|x_0-x^*\right|>0$$

若 $x_1\in[a,b]$，则有

$$\left|x_2-x^*\right|=\left|\varphi'(\zeta_1)(x_1-x^*)\right|\geq\left|x_1-x^*\right|\geq\left|x_0-x^*\right|$$

如此继续下去，或者 $x_k\notin[a,b]$，或者 $\left|x_k-x^*\right|\geq\left|x_0-x^*\right|$，即发散。

例3.　用迭代法求方程的近似解,精确到小数点后6位

$$e^x + 10x - 2 = 0$$

解:　　由于$e^x > 0$,　则$10x - 2 < 0$　　$x < 0.2$

$x < 0$时，$0 < e^x < 1$，$10x - 2 < -2$，无解

因此[0,0.2]为有根区间

直观上看有两种简单构造形式

$$x = \varphi_1(x) = \frac{2 - e^x}{10} \qquad x = \varphi_2(x) = \ln(2 - 10x)$$

由于 $|\varphi_1'(x)| = \dfrac{e^x}{10} < \dfrac{e^{0.2}}{10} < 1$　$|\varphi_2'(x)| = \dfrac{10}{2 - 10x} \geq 5$

因此采用迭代函数　$x = \varphi_1(x) = \dfrac{2 - e^x}{10}$

取初值$x_0 = 0$

$$x_1 = \frac{2 - e^{x_0}}{10} = 0.1$$

x1 = 0.1000000	d1 = 0.1000000
x2 = 0.0894829	d2 = -0.0105171
x3 = 0.0906391	d3 = 0.1156e-002
x4 = 0.0905126	d4 = -0.1265e-003
x5 = 0.0905265	d5 = 0.1390e-004
x6 = 0.0905250	d6 = -0.1500e-005
x7 = 0.0905251	d7 = 0.1000e-006

由于|d7| =0.1000e-006<1e-6

因此原方程的解为 $x^* \approx$ x7 = 0.090525

定理3：

如果函数$\psi(x)$在x^*的一邻域$O(x^*,\delta^*)$内可导连续，x^*为方程$x=\psi(x)$的根，且$|\psi'(x^*)|<1$，则存在正数δ，$(\delta<\delta^*)$，使得对任意$x_0 \in \left[x^* - \delta, x^* + \delta \right]$，迭代序列$x_{n+1}=\psi(x_n)(n=0,1,2\ldots)$收敛于$x^*$。

证明：$\psi'(x)$在$O(x^*,\delta^*)$内连续，且$|\psi'(x^*)|<1$，则存在正数$L<1,\delta<\delta^*$，使得对任意 $x \in \left[x^* - \delta, x^* + \delta \right]$，

有$\left|\varphi'(x)\right| \le L < 1$。

$$\because \left|\varphi(x) - x^*\right| = \left|\varphi(x) - \varphi(x^*)\right| \le L\left|x - x^*\right| < \delta$$

$\therefore \varphi(x) \in [x^* - \delta, x^* + \delta]$ 由定理2可知序列收敛。

定理2. 设迭代函数$\varphi(x)$在$[a,b]$上连续,且满足

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

(2) 存在一正数L,满足$0<L<1$,且$\forall x\in[a,b]$,有

$$|\varphi'(x)|\le L \qquad \text{---------(5)}$$

则1^o. 方程$x=\varphi(x)$在$[a,b]$内有唯一解x^*

2^o.对于任意初值$x_0\in[a,b]$,迭代法$x_{k+1}=\varphi(x_k)$均收敛于x^*

$$3^o.\ |x_k-x^*|\le\frac{L}{1-L}|x_k-x_{k-1}| \qquad \text{---------(6)}$$

$$4^o.\ |x_k-x^*|\le\frac{L^k}{1-L}|x_1-x_0| \qquad \text{---------(7)}$$

$$5^o.\ \lim_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(x^*) \qquad \text{---------(8)}$$

迭代过程的收敛速度

定义：设由某方法确定的序列$\{x_k\}$收敛于方程的根x^*，如果存在正实数p，使得

$$\lim_{k\to\infty}\frac{\left|x^*-x_{k+1}\right|}{\left|x^*-x_k\right|^p}=C$$ （C为非零常数）

则称序列$\{x_k\}$收敛于x^*的收敛速度是p阶的，或称该方法具有p 阶敛速。当$p = 1$时，称该方法为线性（一次）收敛；当$p = 2$时，称方法为平方（二次）收敛；当$1< p< 2$时，称方法为超线性收敛。

例4. 设两个迭代分别是线性收敛和平方收敛的：

(1) $\frac{e_{k+1}}{e_k}=\frac{1}{2}(k=0,1,2,\ldots)$ **(2)** $\frac{\tilde{e}_{k+1}}{\tilde{e}_k^2}=\frac{1}{2}(k=0,1,2,\ldots)$

其中$e_k=x_k-x^*$，已知 $e_0=\tilde{e}_0=1/3$ **，若取计算精度10^{-10}，**
分别估计二者所需迭代次数。

解 (1) $e_k=\frac{1}{2}e_{k-1}=\cdots=\frac{1}{2^k}e_0=\frac{1}{2^k}\frac{1}{3}$

要使 $|e_k|\le 10^{-10}$ ⟹ 只要 $\frac{1}{2^k}\frac{1}{3}\le 10^{-10}$

解得 $k\ge 31.63$

⟹ 需要迭代**32**次才能满足精度要求。

$$(2)\quad \tilde{e}_k = \frac{1}{2}\tilde{e}_{k-1}^2 = \cdots = \left(\frac{1}{2}\right)^{2^k-1}\tilde{e}_0^{2^k} = \left(\frac{1}{2}\right)^{2^k-1}\left(\frac{1}{3}\right)^{2^k} = 2\left(\frac{1}{6}\right)^{2^k}$$

要使 $\left|\tilde{e}_k\right| \le 10^{-10}$ $\Longrightarrow$ 只要 $2\left(\frac{1}{6}\right)^{2^k} \le 10^{-10}$

解得 $k \ge 3.73$

$\Longrightarrow$ 需要迭代4次就能满足精度要求。

如何判断迭代函数的收敛速度呢?

定理4:

设迭代函数$\psi(x)$在x^*的邻域有r阶连续导数$(r\geq 2)$，且$x^*=\psi(x^*)$，$\psi^{(k)}(x^*)=0(k=1,\ldots,r-1)$, $\psi^{(r)}(x^*)\neq 0$，则迭代公式所产生的序列$\{x_n\}$是r阶收敛的。若$0<|\psi'(x^*)|<1$，则迭代序列是线性收敛的。

证明： $\because |\psi'(x^*)|<1$ $\therefore$满足定理3的条件，在x^*的某一邻域内，$\{x_n\}$是收敛于x^*。

$$x_{n+1}=\varphi(x_n)=\varphi(x^*)+\varphi'(x^*)(x_n-x^*)+\dots+\frac{\varphi^{r-1}(x^*)}{(r-1)!}(x_n-x^*)^{r-1}$$

$$+\frac{\varphi^{r}(\xi)}{r!}(x_n-x^*)^{r}$$

ξ在x^*与x_n之间

$$=x^*+\frac{\varphi^{r}(\xi)}{r!}(x_n-x^*)^{r}$$

$$\Rightarrow\quad \frac{x_{n+1}-x^*}{(x_n-x^*)^{r}}=\frac{\varphi^{r}(\xi)}{r!}$$

$$\Rightarrow\quad \lim_{n\to\infty}\frac{\left|x_{n+1}-x^*\right|}{\left|(x_n-x^*)\right|^{r}}=\lim_{n\to\infty}\left|\frac{\varphi^{r}(\xi)}{r!}\right|=\left|\frac{\varphi^{r}(x^*)}{r!}\right|$$

$\varphi^{r}(x^*)\neq 0$ **则迭代公式所产生的序列$\{x_n\}$是r阶收敛的**

$$0<\left|\varphi'(x^*)\right|<1\qquad \lim_{n\to\infty}\frac{\left|x_{n+1}-x^*\right|}{\left|(x_n-x^*)\right|}=\lim_{n\to\infty}\left|\frac{\varphi(x_n)-\varphi(x^*)}{x_n-x^*}\right|=\left|\varphi'(x^*)\right|$$

设迭代格式 $x_{k+1}=\varphi(x_k)$是收敛的，由定理2.1，有

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\varphi'(x^*)$$

当k适当大的时候， $$\frac{x_{k+2}-x^*}{x_{k+1}-x^*}\approx\frac{x_{k+1}-x^*}{x_k-x^*}$$

解出 $$x^*\approx\frac{x_k x_{k+2}-x_{k+1}^2}{x_{k+2}-2x_{k+1}+x_k}$$

$$\left.\begin{aligned}x_{k+2}&=\varphi(x_{k+1})\\x_{k+1}&=\varphi(x_k)\end{aligned}\right\}\Rightarrow\left.\begin{aligned}x_{k+2}&=\varphi(\varphi(x_k))\\x_{k+1}&=\varphi(x_k)\end{aligned}\right\}$$ 代入上式

得到 $$x^*\approx\boxed{\frac{x_k\varphi(\varphi(x_k))-\varphi(x_k)^2}{\varphi(\varphi(x_k))-2\varphi(x_k)+x_k}}$$

$\phi(x_k)$

得到一个新的迭代格式（**即埃特金加速法**）

$$x_{k+1}=\phi(x_k)\qquad k=0,1,2,\ldots$$

设x*是 $x=\varphi(x)$ 的根，且 $\varphi(x)$ 在x*的附近存在一阶连续导数，那么 $\lim\limits_{x\to x^*}\phi(x)=x^*$ ？

证　由罗必塔法则

$$\lim_{x\to x^*}\phi(x)=\lim_{x\to x^*}\frac{\varphi(\varphi(x))+x\varphi'(\varphi(x))\varphi'(x)-2\varphi(x)\varphi'(x)}{\varphi'(\varphi(x))\varphi'(x)-2\varphi'(x)+1}$$

$$=\frac{\varphi(\varphi(x^*))+x^*\varphi'(\varphi(x^*))\varphi'(x^*)-2\varphi(x^*)\varphi'(x^*)}{\varphi'(\varphi(x^*))\varphi'(x^*)-2\varphi'(x^*)+1}$$

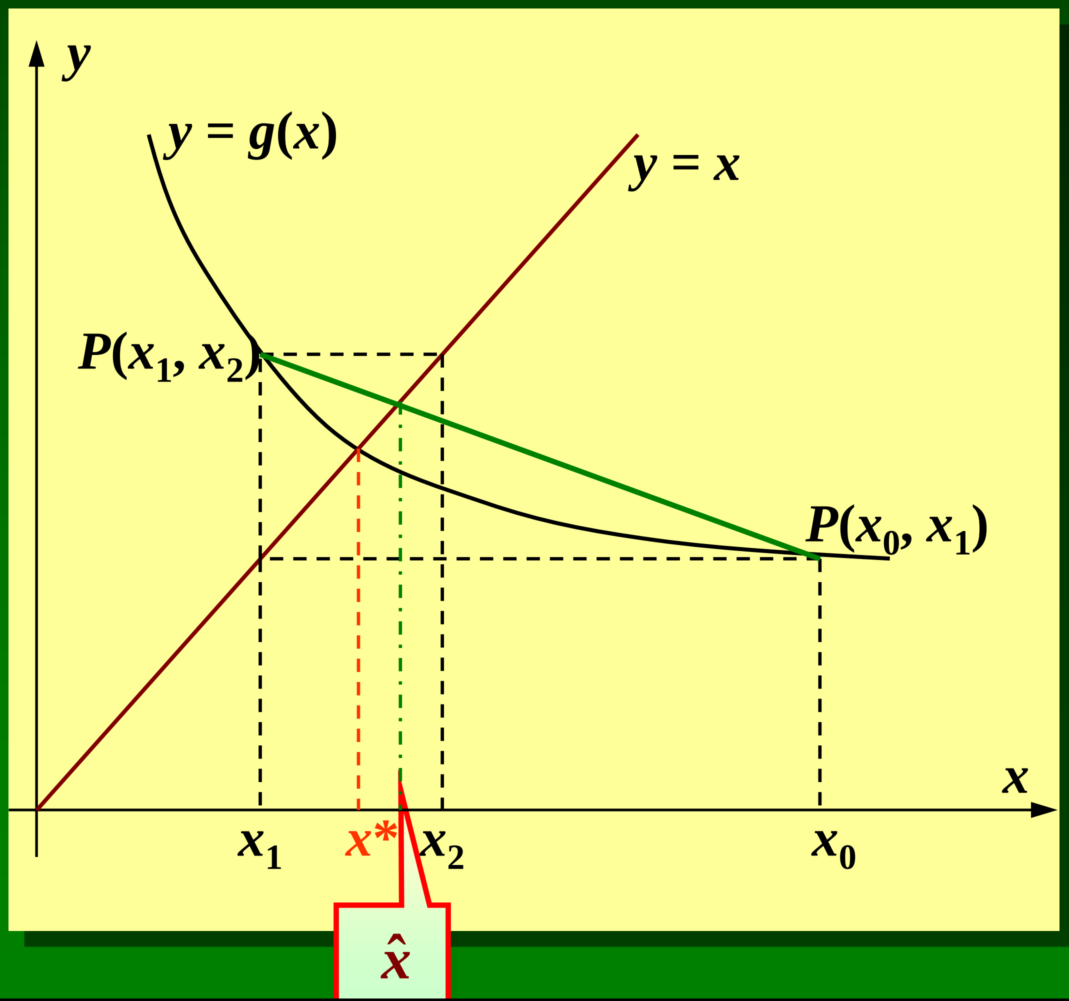
利用

$x^*=\varphi(x^*)$

$$=\frac{x^*+x^*(\varphi'(x^*))^2-2x^*\varphi'(x^*)}{(\varphi'(x^*))^2-2\varphi'(x^*)+1}=x^*$$

Atkin's method from another point of view

$$\frac{x_2 - x^*}{x_1 - x^*} \approx \frac{x_1 - x^*}{x_0 - x^*}$$



定理5:

设在x^*附近$\varphi(x)$有$(p+1)$ 阶连续导数,则对一个充分靠近x^*的初始值x_0:

（1）迭代格式$\varphi(x)$是线性收敛的，则迭代格式$\Phi(x)$是二阶收敛的。

（2）如果迭代格式$\varphi(x)$是p（$p\geq 2$）阶收敛的，则迭代格式$\varphi(x)$是（$2p-1$)阶收敛的。

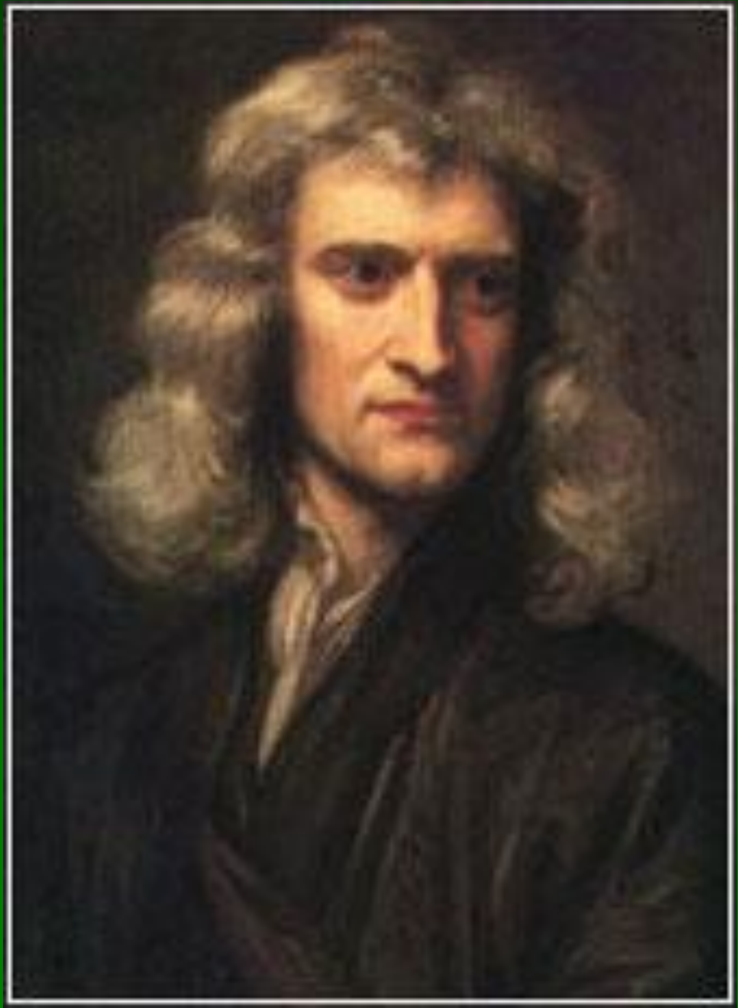
证明见参考文献:

Szidarovszky F，Yakowitz S著，施明光，潘仲雄译。《数值分析的原理及过程》，上海科学技术文献出版社，1982。

牛 顿 法

牛顿（Newton）迭代法，是牛顿在17世纪提出的一种在实数域和复数域上近似求解方程的方法。方法使用函数$f(x)$的泰勒级数的前面几项来寻找方程$f(x) = 0$的根。

牛顿迭代法是求方程根的重要方法之一，其最大优点是在方程$f(x) = 0$的单根附近具有平方收敛，而且该法还可以用来求方程的重根、复根。



牛顿法

设x_k是$f(x)=0$的一个近似根，把$f(x)$在x_k处作一阶泰勒展开

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

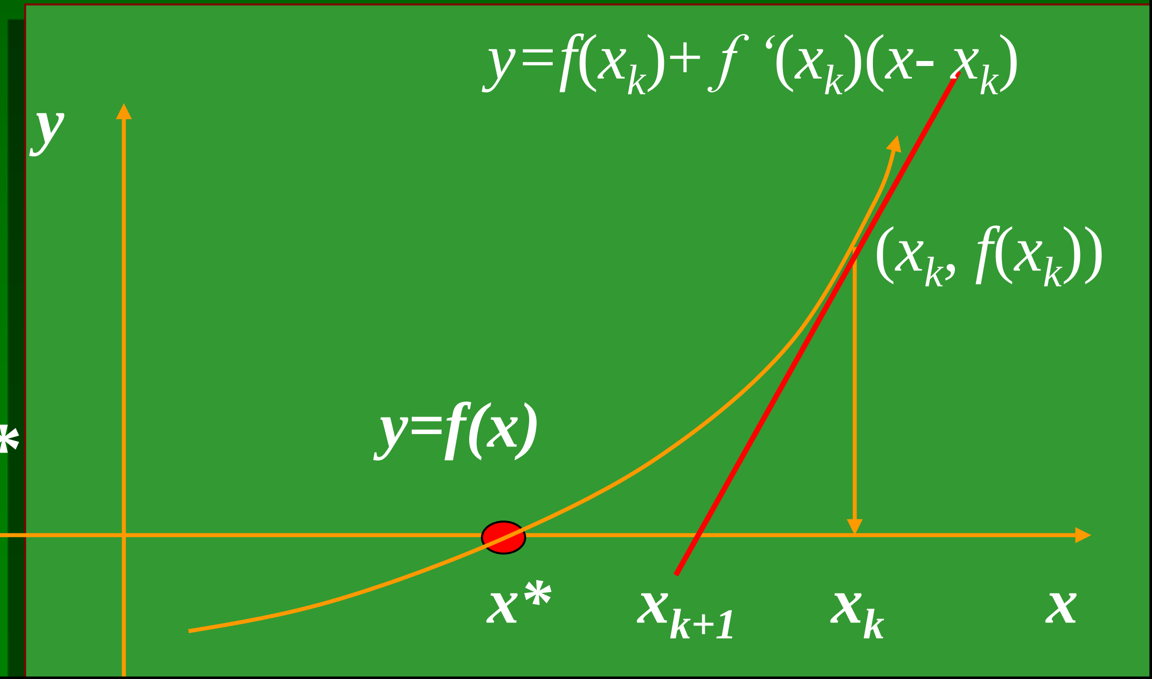
得到

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

设 $f'(x_k) \neq 0$，牛顿迭代公式为

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

x_{k+1}比x_k更接近于x^*



例4. 设$f(a)=0$,且$f'(a)\neq 0$,证明迭代法

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$ 至少是平方收敛的

证明: 令 $$\varphi(x)=x-\frac{f(x)}{f'(x)}$$

则 $$\varphi'(x)=1-\frac{[f'(x)]^2-f(x)f''(x)}{[f'(x)]^2}=\frac{f(x)f''(x)}{[f'(x)]^2}$$

所以 $\varphi'(a)=0$

由定理5 该迭代法至少是平方收敛的

牛顿法

例5 用牛顿法求方程$f(x)=x(x+1)^2-1=0$在$x_0=0.4$附近的根，精确到4位有效数字。

解　对$f(x)$求导得

$$f'(x)=(x+1)(3x+1)$$

牛顿迭代格式为：

$$\begin{cases} x_{k+1}=x_k-\dfrac{f(x_k)}{f'(x_k)}=x_k-\dfrac{x_k\left(x_k+1\right)^2-1}{\left(x_k+1\right)\left(3x_k+1\right)} & k=0,1,2,\ldots \\ x_0=0.4 \end{cases}$$

牛顿法

```
f=inline('x*(x+1)^2-1');
f1=inline('(x+1)*(3*x+1)');
x0=0.4;
er=1;
k=0;
while er>0.5e-4
   x=x0-f(x0)/f1(x0);
   er=abs(x-x0)
   x0=x;
   k=k+1
end
```

k	x_k
0	0.4
1	0.47013
2	0.46559
3	0.46557

$\Rightarrow$ $x^* \approx 0.4656$

例6 用牛顿法求方程$f(x)=x^2+1=0$的根。

Columns 1 through 4

5.0000 - 2.0000i   2.4138 - 1.0345i   1.0319 - 0.5922i   0.1515 - 0.5053i

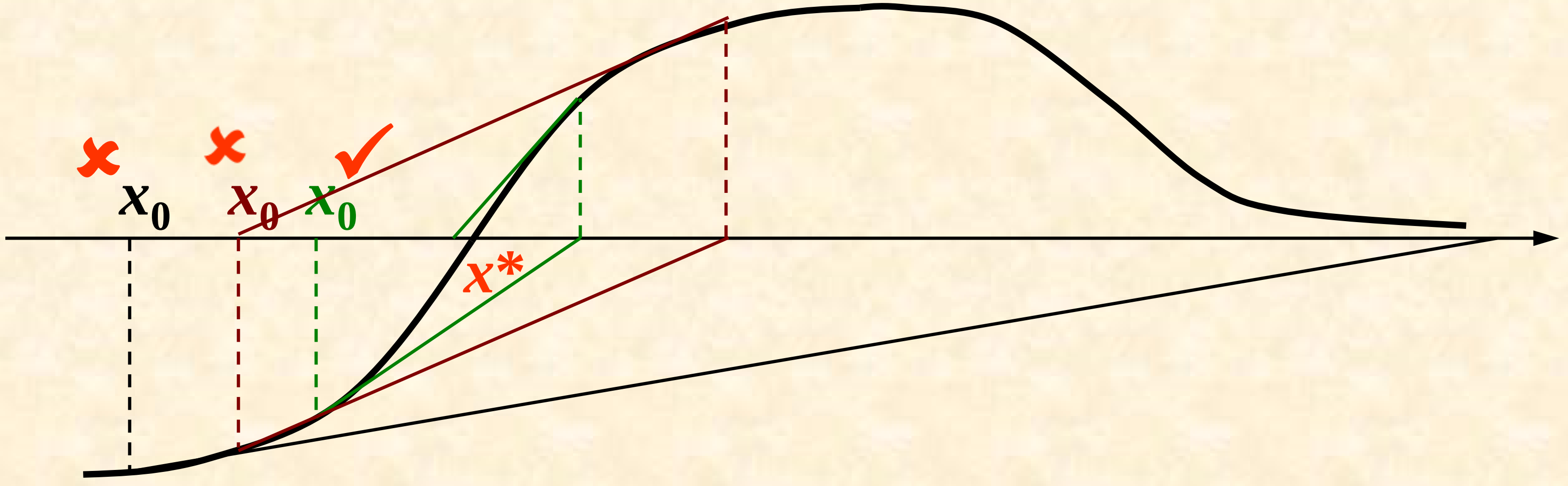
Columns 5 through 8

-0.1964 - 1.1606i -0.0273 - 0.9991i   0.0000 - 0.9996i -0.0000 - 1.0000i

Columns 9 through 11

-0.0000 - 1.0000i -0.0000 - 1.0000i   0.0000 - 1.0000i

牛顿法收敛性示意图



注：Newton法的收敛性依赖于x_0 的选取。

- 与二分法不同，牛顿法一般不在x轴的有限范围内求根，因此其二次收敛是有限制的，在最坏的情况下会出现迭代发散现象，一般用于求解良性函数。

牛顿法

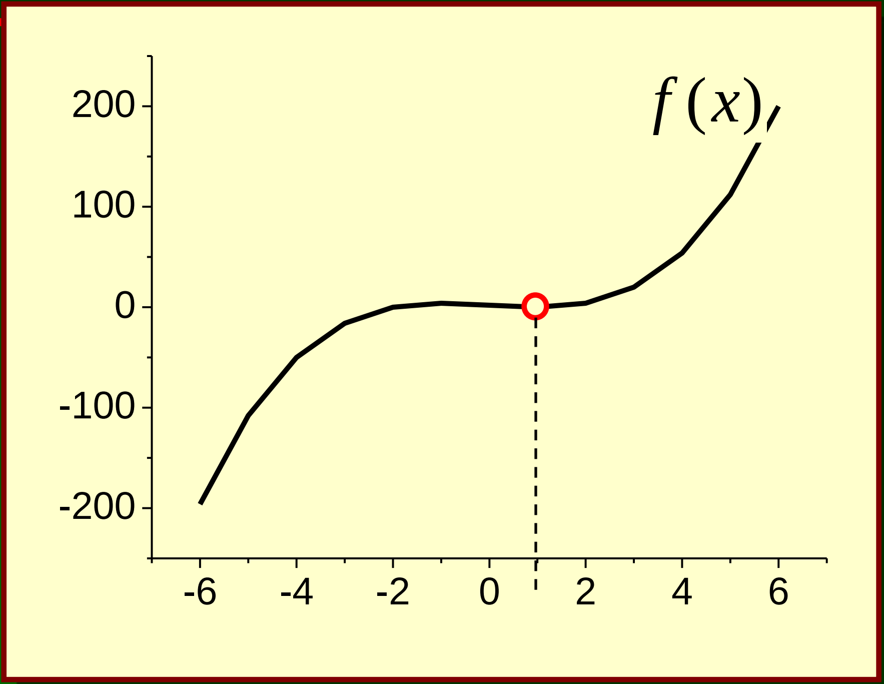
Newton迭代法的缺陷：

1.被零除错误

方程: $f(x)=x^3-3x+2=0$

在重根$x^*=1$附近,

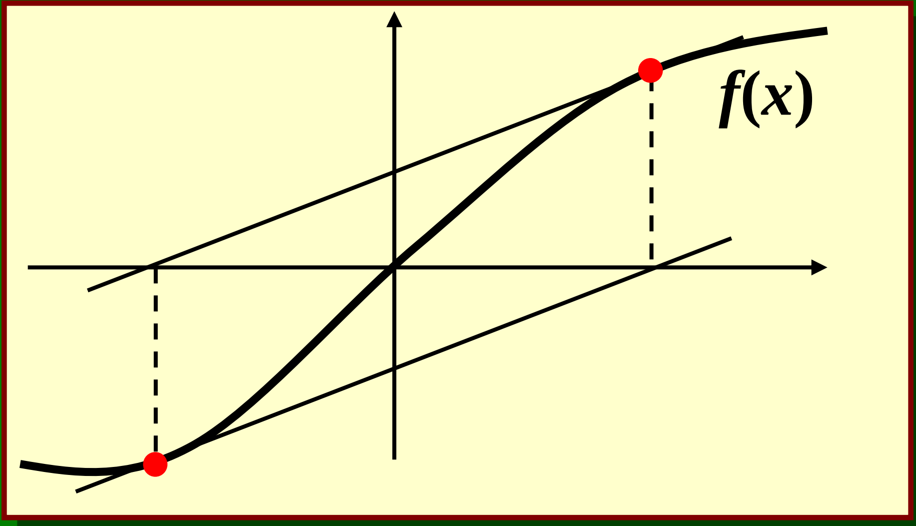
$f'(x)$近似为零



2.程序死循环

方程： $f(x)=\arctan x=0$

当 x_0取1.3917时, Newton迭代法陷入死循环



牛顿法

牛顿迭代法的大范围收敛性

定理2.6　设 $f(x)$在区间$[a, b]$上存在二阶连续导数，且满足条件：

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

(2) $x\in[a,b]$时，$f'(x)\neq 0$；

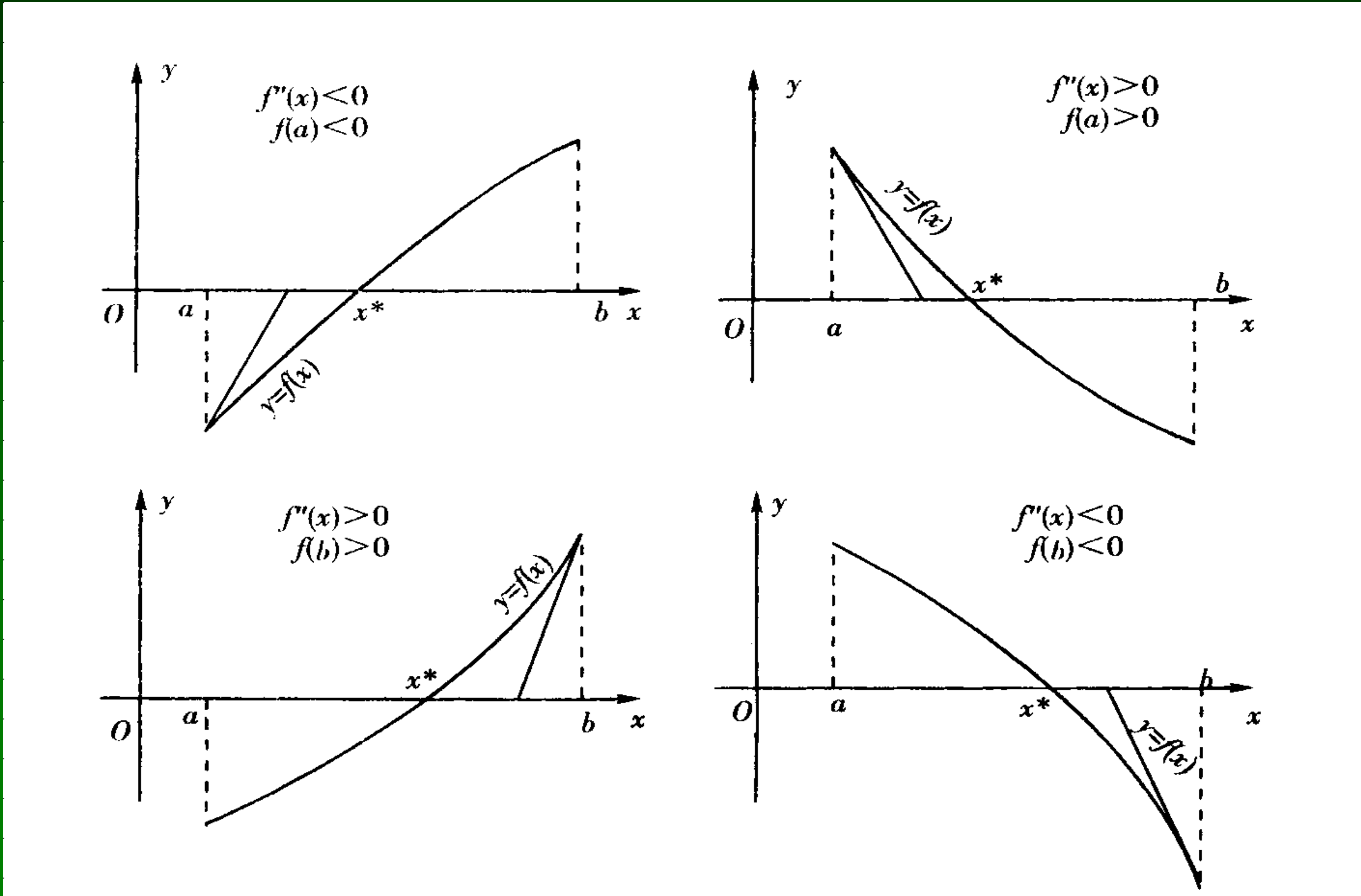
(3) $x\in[a,b]$时，$f''(x)$保号；

(4) $a-\dfrac{f(a)}{f'(a)}\le b,\quad b-\dfrac{f(b)}{f'(b)}\le a$

则对任意初值，Newton迭代序列二阶收敛.

$$初始值x_0\in[a,b]使得f''(x\ )f(x_0)>0$$

牛顿法收敛性示意图



$f''(x)<0$
$f(a)<0$
y
O
a
x*
b
x
y=f(x)
$f''(x)>0$
$f(a)>0$
y
O
a
x*
b
x
y=f(x)
$f''(x)>0$
$f(b)>0$
y
O
a
x*
b
x
y=f(x)
$f''(x)<0$
$f(b)<0$
y
O
a
x*
b
x
y=f(x)

```
function r = newton(fun,x0,xtol,ftol,verbose)
% newton     Newton's method to find a root of the scalar equation f(x) = 0
%% Synopsis:  r = newton(fun,x0)
%            r = newton(fun,x0,xtol)
%            r = newton(fun,x0,xtol,ftol)
%            r = newton(fun,x0,xtol,ftol,verbose)
%% Input:  fun     = (string) name of mfile that returns f(x) and f'(x).
%         x0      = initial guess
%         xtol    = (optional) absolute tolerance on x.    Default: xtol=5*eps
%         ftol    = (optional) absolute tolerance on f(x). Default: ftol=5*eps
%         verbose = (optional) flag.  Default: verbose=0, no printing.
%%  Output:  r = the root of the function
if nargin < 3,  xtol = 5*eps;  end
if nargin < 4,  ftol = 5*eps;  end
if nargin < 5,  verbose = 0;   end
xeps = max(xtol,5*eps);  feps = max(ftol,5*eps);  %  Smallest tols are 5*eps
```

```
if verbose
  fprintf('\nNewton iterations for %s.m\n',fun);
  fprintf('  k      f(x)        dfdx        x(k+1)\n');
end

x = x0;  k = 0;  maxit = 15;     %  Initial guess, current and max iterations
while k <= maxit
  k = k + 1;
  [f,dfdx] = feval(fun,x);      %   Returns f( x(k-1) ) and f'( x(k-1) )
  dx = f/dfdx;
  x = x - dx;
  if verbose, fprintf('%3d %12.3e %12.3e %18.14f\n',k,f,dfdx,x);  end

  if ( abs(f) < feps ) | ( abs(dx) < xeps ),  r = x;  return;  end
end
warning(sprintf('root not found within tolerance after %d iterations\n',k));
```

例4. 设x^*是方程$f(x)=0$的$m(\geq 2)$重根，证明迭代法

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

为线性收敛

证明: 因为x^*是方程$f(x)=0$的m重根，故

$$f(x)=(x-x^*)^m g(x) \qquad 且 g(x^*)\neq 0, m\geq 2$$

所以 $$f'(x)=m(x-x^*)^{m-1}g(x)+(x-x^*)^m g'(x)$$

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

$$=x_k-\frac{(x_k-x^*)g(x_k)}{mg(x_k)+(x_k-x^*)g'(x_k)}$$

$$x_{k+1} - x^* = (x_k - x^*)\left(1 - \frac{g(x_k)}{mg(x_k) + (x_k - x^*)g'(x_k)}\right)$$

$$\lim_{x_k \to x^*} \frac{x_{k+1} - x^*}{x_k - x^*} = \lim_{x_k \to x^*}\left(1 - \frac{g(x_k)}{mg(x_k) + (x_k - x^*)g'(x_k)}\right) = 1 - \frac{1}{m}$$

$m \geq 2$时，$1 - \frac{1}{m} > 0$ 由定义1

该迭代法对$m(\geq 2)$重根是线性收敛的

已知x^*是$f(x)=0$的m重根，则可用下面迭代法

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

如果未知x^*是$f(x)=0$的几重根，则可用下面迭代法

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

均可实现至少二阶收敛

Newton迭代法的变形

Newton迭代法 $$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$ --------(12)

需要求每个迭代点处的导数 $f'(x_k)$ 复杂！

用 x_0 近似替代 $f'(x_k)$ 中的 x_k，得

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_0)}$$ --------(13)

这种格式称为简化Newton迭代法 精度稍低

如果用数值导数代替 $f'(x_k)$ $$f'(x_k)\approx\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$$

则Newton迭代法变为

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}) \quad \text{---------}(14)$$

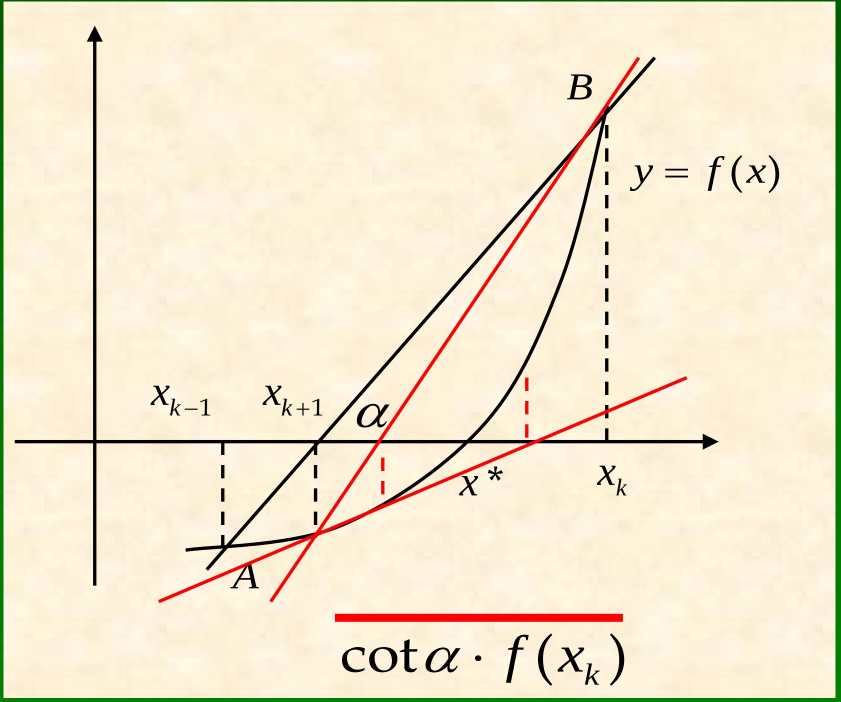
这种格式称为弦截法。

如图, AB的斜率为$K_{AB} = \dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$

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

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

$$= x_k - \cot\alpha \cdot f(x_k)$$



几何意义

例6. 用简化Newton法和弦截法解例(5)中方程的根，并和Newton 迭代法比较 $x^3-3x+1=0$

解: 设 $f(x)=x^3-3x+1$ $\quad f'(x)=3x^2-3$

由简化Newton法

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_0)}=x_k-\frac{x_k^3-3x_k+1}{3x_0^2-3}$$

由弦截法

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

简化Newton法

x0=0.5
x1= 0.3333333333
x2 = 0.3497942387
x3 = 0.3468683325
x4 = 0.3473702799
x5 = 0.3472836048
x6 = 0.3472985550
x7 = 0.3472959759
x8 = 0.3472964208
x9 = 0.3472963440
x10 = 0.3472963572
x11 = 0.3472963553

由弦截法

x0=0.5;
x1=0.4;
x2 = 0.3430962343
x3 = 0.3473897274
x4 = 0.3472965093
x5 = 0.3472963553
x6 = 0.3472963553

要达到精度10^{-8}

简化Newton法迭代11次

弦截法迭代5次

Newton迭代法迭代4次

无论前面哪种迭代法：

Newton迭代法　简化Newton法　弦截法

是否收敛均与初值的位置有关

如　$f(x)=\arctan(x)=0$　精确解为 $x=0$

Newton迭代法　$x_{k+1}=x_k-\arctan x_k\cdot(1+x_k^2)$

取初值 $x_0=1$

x0 =1
x1 = -0.5708
x2 = 0.1169
x3 = -0.0011
x4 = 7.9631e-010
x5 = 0

收敛

若取初值 $x_0=2$

x0 = 2
x1 = -3.54
x2 = 13.95
x3 = -279.34
x4 = 122017

发散

如果在构造迭代法时加入要求：$|f(x_{k+1})|<|f(x_k)|$

因此考虑引入一因子λ,建立迭代法

$$x_{k+1} = x_k - \lambda \frac{f(x_k)}{f'(x_k)} \qquad \text{---------(15)}$$

在迭代时,选择一个λ,使得

$$|f(x_{k+1})|<|f(x_k)|$$

这种方法称为Newton下山法，λ称为下山因子

λ的选取方式　按$\lambda = 1, \frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \cdots$的顺序

直到$|f(x_{k+1})|<|f(x_k)|$成立为止

例7. 求解方程$f(x)=\frac{x^3}{3}-x$,取初值$x_0=-0.99$

$|x_n-x_{n-1}|\le 10^{-5}$

解: 1.先用Newton迭代法 $f'(x)=x^2-1$

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

$$x_1=x_0-\frac{x_0^3-3x_0}{3(x_0^2-1)}=-32.50598$$

$$x_2=x_1-\frac{x_1^3-3x_1}{3(x_1^2-1)}=-21.69118$$

$$x_3=x_2-\frac{x_2^3-3x_2}{3(x_2^2-1)}=15.15689$$

x4 =	9.70724
x5 =	6.54091
x6 =	4.46497
x7 =	3.13384
x8 =	2.32607
x9 =	1.90230
x10=	1.75248
x11=	1.73240
x12=	1.73205
x13=	1.73205

迭代13次才达到精度要求

2.用Newton下山法,结果如下

k	下山因子	x_k	$f(x_k)$
k=0		x0 =-0.99	fx0 =0.666567
k = 1		x1 =32.505829	f(x) = 11416.4
	w =0.5	x1 =15.757915	f(x) = 1288.5
	w =0.25	x1 =7.383958	f(x) =126.8
	w =0.125	x1 =3.196979	f(x) =7.69
	w = 0.0625	x1 =1.103489	f(x)=-0.655
k = 2		x2 = 4.115071	f(x) =19.1
	w = 0.5	x2 = 2.60928	f(x)=3.31
	w =0.25	x2 =1.85638	f(x)=0.27
k = 3		x3 =1.74352	f(x)=0.023
k = 4		x4 = 1.73216	f(x)=0.00024
k = 5		x5 = 1.73205	f(x)=0.00000
k = 6		x6 = 1.73205	f(x)=0.000000

§ 6 劈因子法 /* Splitting Method */

求多项式的根

从 $f(x)$中分离出一个2 次因子。即：

$$f(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$
$$= (x^2 + u^* x + v^*)(b_0^* x^{n-2} + b_1^* x^{n-3} + \cdots + b_{n-3}^* x + b_{n-2}^*)$$

通过 $x^2 + u^* x + v^* = 0$可解出一对共轭复根。

思路 从一对初值 (u, v) 出发，则有

$$f(x) = (x^2 + u x + v) P(x) + (r x + s)$$

其中(r, s)取决于u 和 v，可以看作是(u, v)的函数，即 $r = r(u, v)$，$s = s(u, v)$。

目标： $r = r(u^*, v^*) = 0$，$s = s(u^*, v^*) = 0$。

将r 和 s 在初值点(u, v)做一阶Taylor展开，并代入(u^*, v^*)：

$$\begin{cases} 0 = r(u^*,v^*) \approx r(u,v) + \dfrac{\partial r}{\partial u}\underline{(u^*-u)} + \dfrac{\partial r}{\partial v}\underline{(v^*-v)} \\ 0 = s(u^*,v^*) \approx s(u,v) + \dfrac{\partial s}{\partial u}\underline{(u^*-u)} + \dfrac{\partial s}{\partial v}\underline{(v^*-v)} \end{cases}$$

从中解出$\Delta u = u^* - u$, $\Delta v = v^* - v$**，以**$u+\Delta u$**，**$v+\Delta v$ **更新** u **和** v **再迭代，直到** r **和** s **充分接近0。**

$$f(x) = (x^2 + ux + v)P(x) + (rx + s)$$

每步迭代须计算 r，s，$\dfrac{\partial r}{\partial u}$，$\dfrac{\partial r}{\partial v}$，$\dfrac{\partial s}{\partial u}$，$\dfrac{\partial s}{\partial v}$.

➢ **计算** $\dfrac{\partial r}{\partial v}, \dfrac{\partial s}{\partial v}$**：**

$$f(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$
$$= (x^2 + ux + v) P(x) + rx + v$$

$$\Rightarrow \quad \frac{\partial f}{\partial v} = 0$$
$$= P(x) + (x^2 + ux + v) \cdot \frac{\partial P}{\partial v} + \frac{\partial r}{\partial v} x + \frac{\partial s}{\partial v}$$

$$\Rightarrow \quad -P(x) = (x^2 + ux + v) \frac{\partial P}{\partial v} + \frac{\partial r}{\partial v} x + \frac{\partial s}{\partial v}$$

n–2 **阶多项式**

n–4 **阶多项式**

与前一步同理，可导出 $\dfrac{\partial r}{\partial v}, \dfrac{\partial s}{\partial v}$ **和** $\dfrac{\partial P}{\partial v}$ **的公式。**

➢ **计算** $\dfrac{\partial r}{\partial u}, \dfrac{\partial s}{\partial u}$**:**

$$f(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$
$$= (x^2 + ux + v)P(x) + rx + v$$

➡ $$\frac{\partial f}{\partial u} = 0$$
$$= xP(x) + (x^2 + ux + v) \cdot \frac{\partial P}{\partial u} + \frac{\partial r}{\partial u} x + \frac{\partial s}{\partial u}$$

➡ $$-xP(x) = (x^2 + ux + v)\frac{\partial P}{\partial u} + \frac{\partial r}{\partial u} x + \frac{\partial s}{\partial u}$$

而前一步得到 $-P(x) = (x^2 + ux + v)\dfrac{\partial P}{\partial v} + \dfrac{\partial r}{\partial v} x + \dfrac{\partial s}{\partial v}$

$$\begin{cases} 0 = r(u^*, v^*) \approx r(u, v) + \dfrac{\partial r}{\partial u}(u^* - u) + \dfrac{\partial r}{\partial v}(v^* - v) \\ 0 = s(u^*, v^*) \approx s(u, v) + \dfrac{\partial s}{\partial u}(u^* - u) + \dfrac{\partial s}{\partial v}(v^* - v) \end{cases}$$