第五章 非线性方程的数值解法

肖　明

微电子科学与技术学院

邮箱：xiaom37@mail.sysu.edu.cn

目录

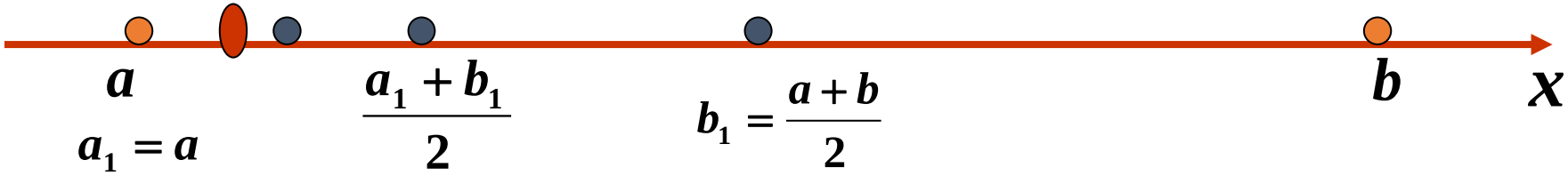
引言-回顾

本章主要讨论单变量非线性方程

$$f(x)=0$$

的求根问题

- 牛顿二分法是实现单实变量非线性方程零点可靠的求解方法，但存在**收敛速度慢**和对**有偶数重根**不适用等问题。



- 不动点迭代法可以实现快速求解方程的根，但是**迭代函数的选择**对求解结果的**收敛性**和**求解速率**有较大影响。要求迭代函数的导数满足条件 $|\varphi'(x)| < 1$

$$x_{k+1} = \varphi(x_k)$$

引言

艾萨克·牛顿（Isaac Newton，
1643年1月4日—1727年3月31日）

约瑟夫·拉弗森（Joseph Raphson，
1648年 - 1715年）

牛顿迭代法原理

基本思路：将非线性方程 $f(x)=0$ 线性化。

取x_0作为初始近似值，将 $f(x)$在x_0做Taylor展开:

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(\xi)}{2!}(x-x_0)^2$$

$$0=f(x^*)\approx f(x_0)+f'(x_0)(x^*-x_0)\Rightarrow \quad x^*\approx x_0-\frac{f(x_0)}{f'(x_0)}$$

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$ **作为第一次近似值**

重复上述过程 ⇒

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

Newton 迭代公式

牛顿迭代法原理

简单验证：

若解数列x_k是收敛的，且收敛到x*,则有解。

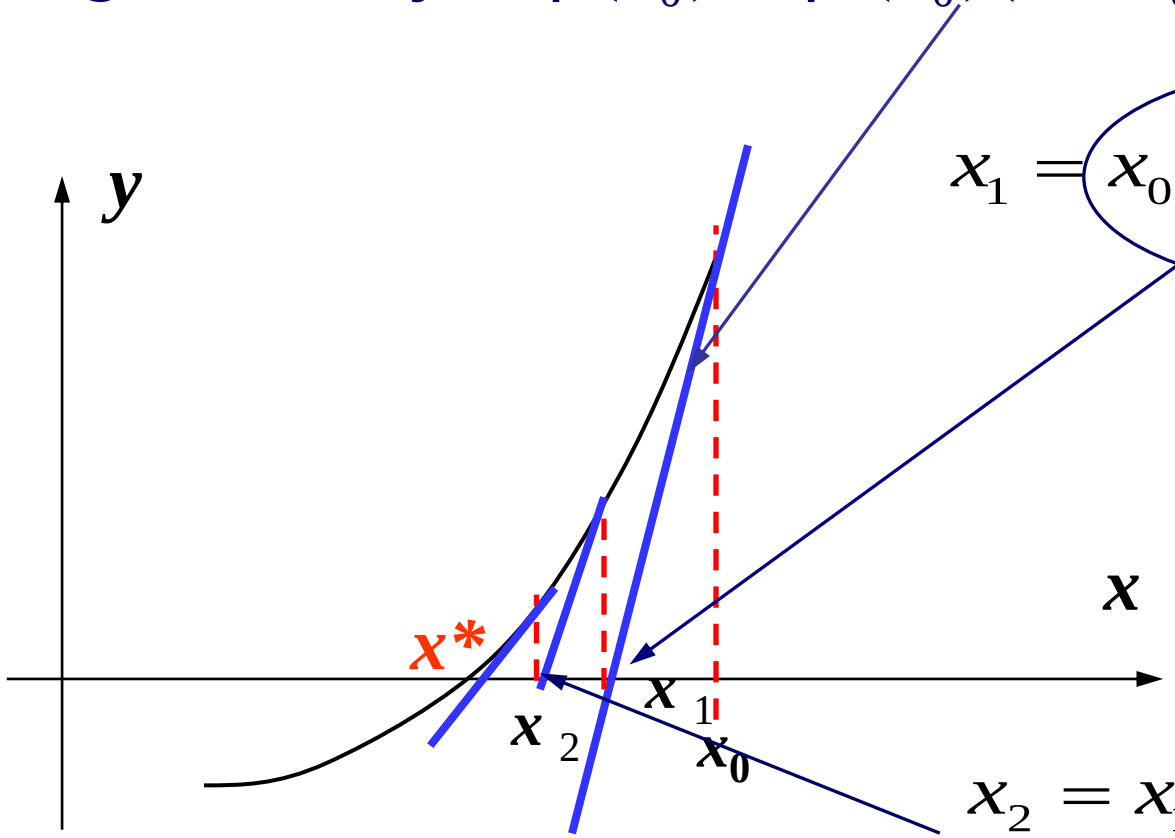
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \Rightarrow f(x_k) = f'(x_k)(x_k - x_{k+1})$$

$$f(x^*) \quad f'(x^*) \quad (x^* - x^*)$$

即$f(x^*) = f'(x^*)(x^* - x^*) = 0$，则$x^*$是方程的根。

牛顿迭代法的几何解释

$$Tangent\ line: y = f(x_0) + f'(x_0)(x - x_0)$$



$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

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

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

其迭代函数为

$$\varphi(x) = x - \frac{f(x)}{f'(x)},$$

故：牛顿法也称为切线法

牛顿迭代法的计算步骤

Newton 法：

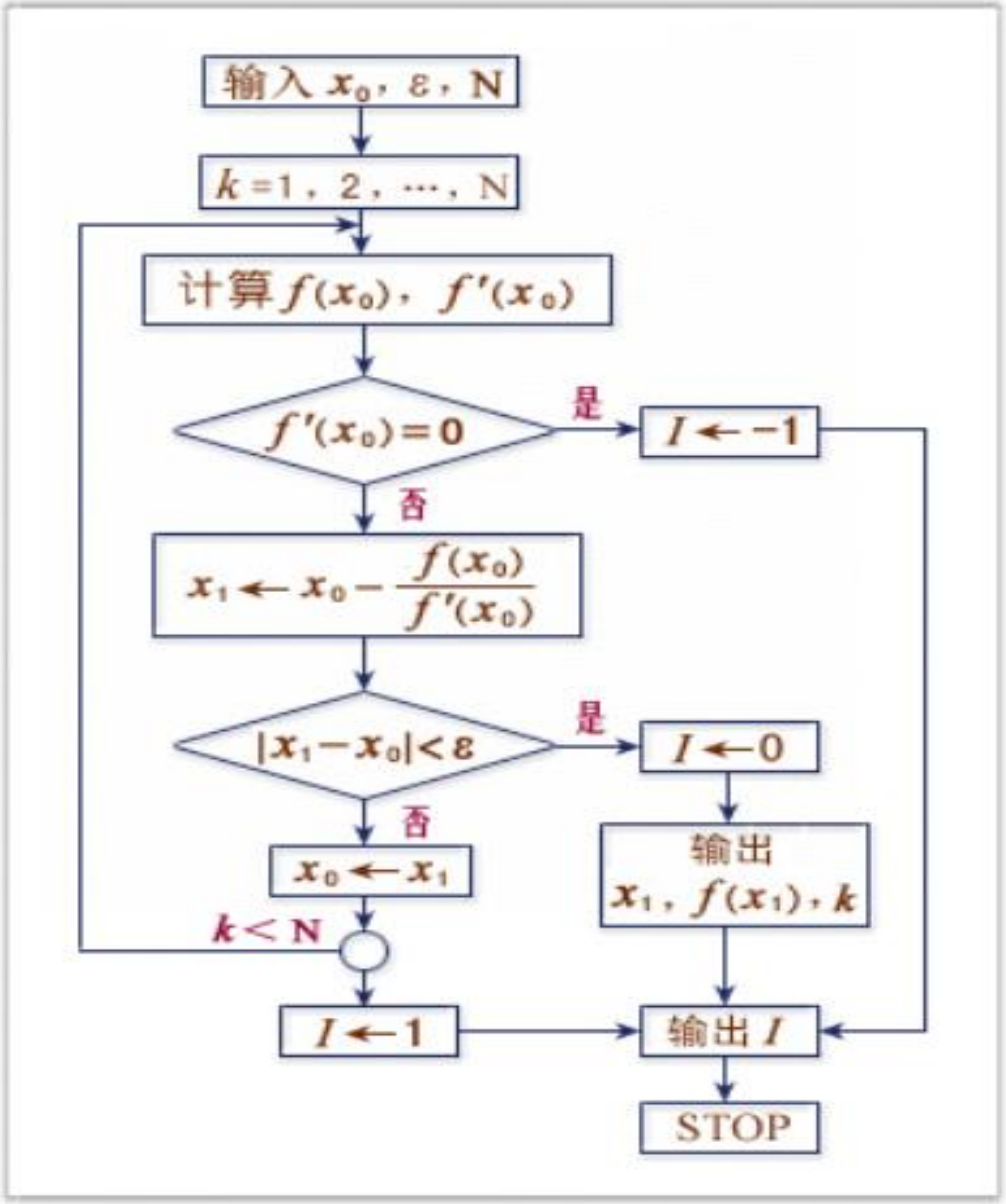
(1) 任取迭代初始值 x_0

(2) 计算 $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$

(3) 判断收敛性：如果 $|x_1 - x_0| < \varepsilon$ 或者 $|f(x_1)| < \varepsilon$ ，

则算法收敛，停止计算，输出近似解 x_1

(4) 令 $x_0 \leftarrow x_1$ ，返回第 (2) 步



牛顿迭代法的使用

例 1 根据牛顿法写出求解$\sqrt{a}$的计算公式，并计算$\sqrt{7}$。

解： 设$f(x)=x^2-a$，求$\sqrt{a}$即计算$f(x)=0$的根。

则由牛顿迭代公式可得：

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

$$=\frac{1}{2}\left(x_k+\frac{a}{x_k}\right).$$

令初始值$x_0=3$, 得

$$x_1=\frac{1}{2}\left(3+\frac{7}{3}\right)=2.667,\ x_2=\frac{1}{2}\left(2.667+\frac{7}{2.667}\right)=2.6458$$

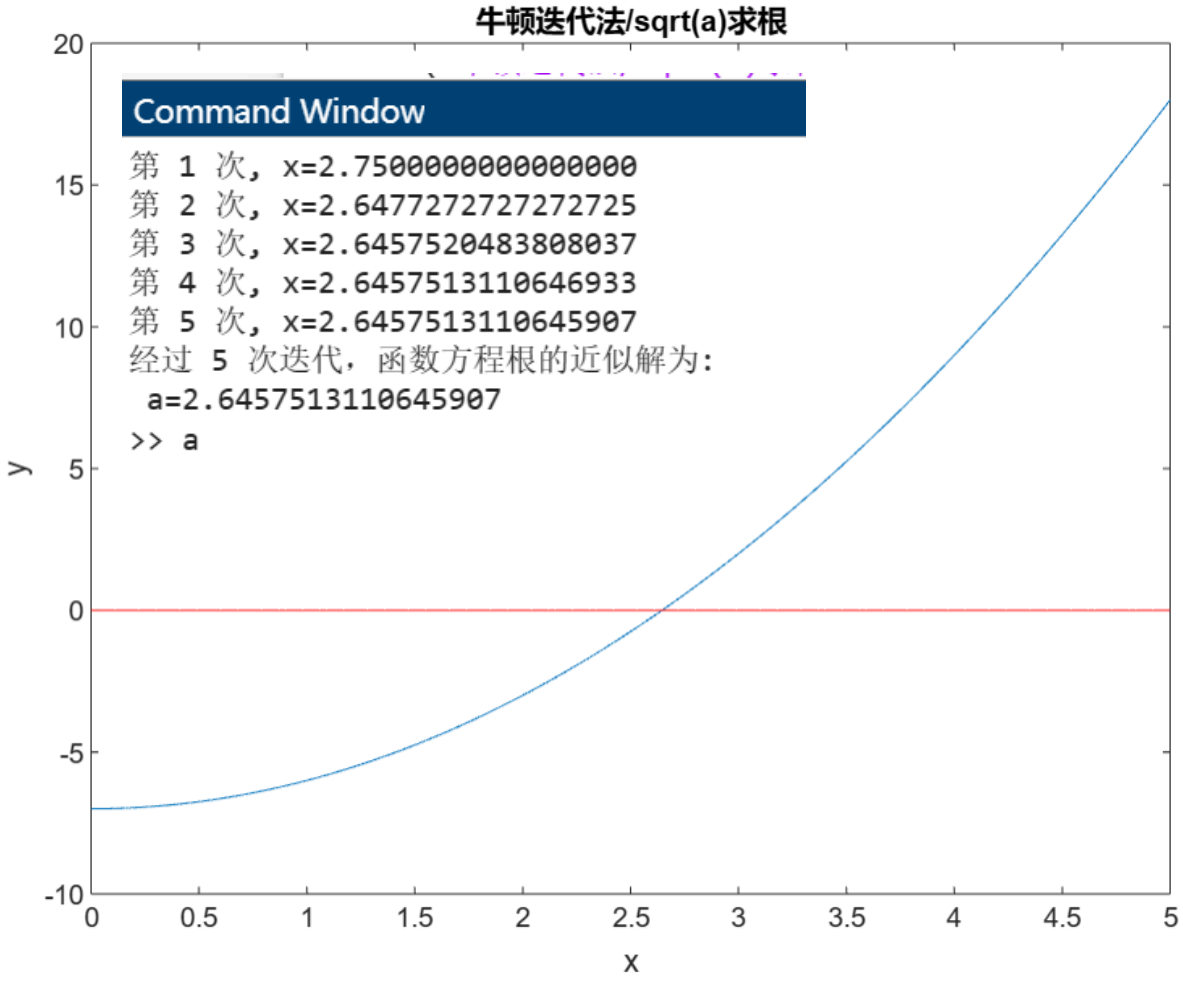
$$\sqrt{7}=2.6457513.$$

牛顿迭代法的使用

例 1

根据牛顿法写出求解$\sqrt{a}$的计算公式，并计算$\sqrt{7}$。

```
clear
clear all;
clc;
b=7;
f=@(x) x.^2-b;
% x=(x+1)^(1/3);
a=2; % 初始值
esp=1e-10; % 迭代终止条件
N=100; % 最大迭代次数
y=zeros(N, 1); % 暂存x变量的空间
for t=1:N
    a=fun(a,b);
    y(t)=a;
    fprintf('第 %d 次, x=%.16f\n', t, a);
    if t>1
        if abs(y(t)-y(t-1))<esp
            break;
        end
    end
end
fprintf('经过 %d 次迭代，函数方程根的近似解为:\n a=%.16f\n',t,a)%输出迭代过程
% 画出函数曲线
xx=0:0.01:5;
yy=xx.^2-b;
figure(1)
plot(xx, real(yy));
hold on
z=0*ones(1, length(xx));
plot(xx, z, 'r');
xlabel('x');
ylabel('y');
title('牛顿迭代法/sqrt(a)求根');
%saveas(gcf,sprintf('不动点迭代法.jpg'),'bmp');
function x=fun(x,b)
%b为所需要的目标值
x=(x+b./x)/2;
end
```



牛顿迭代法的使用

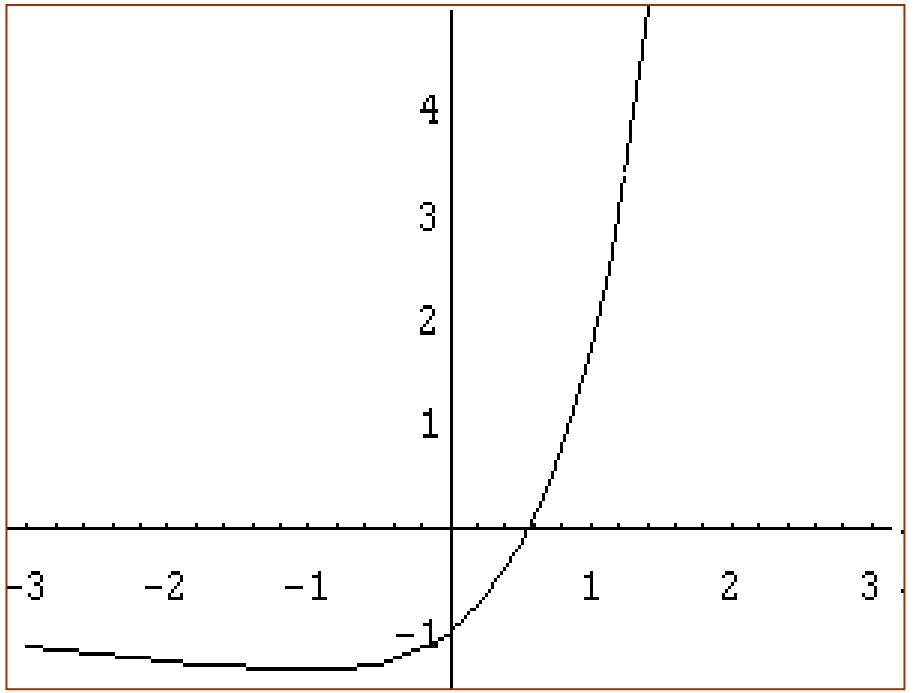
例2. 用牛顿法解方程 $xe^x - 1 = 0.$

解 $f(x) = xe^x - 1,\ f'(x) = (1+x)e^x$

$f(0) = -1 < 0,\ f(1) = e - 1 > 0,$

且$x \in [0,1],\ f'(x) > 0,$

$\therefore\ f(x) = 0$有唯一根$x^* \in (0,1)$



取$x_0 = 0.5$ $\quad x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)} = x_k - \dfrac{x_k e^{x_k} - 1}{e^{x_k}(1+x_k)}$

牛顿迭代法的使用

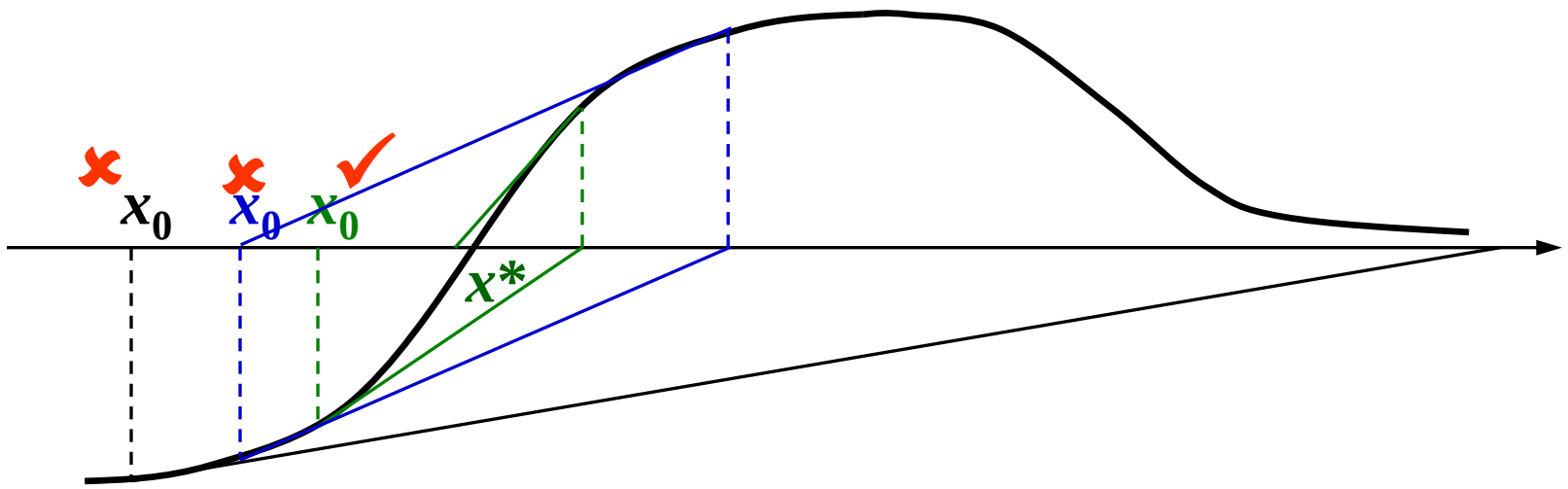
例2. 用牛顿法解方程 $xe^x - 1 = 0.$

```
23  function [p,k,Y]=NTM(f,p0,tol,maxK)
24  %p0表示迭代初始值
25  %f表示要求解的方程
26  %maxK表示规定的最大迭代次数
27  %tolr表示允许误差
28  %k表示最终迭代的次数
29  %p表示最终迭代的值
30      syms x;
31      P(1)=p0;
32      k=2;
33      df=diff(f);      %利用diff()函数计算f(x)的导数
34
35      P(k)=P(k-1)-subs(f,x,P(k-1))/subs(df,x,P(k-1)); %第二次迭代的结果
36      fprintf('\n第 %d 次, x=%.16f', k, P(k));
37      fprintf('    误差值err=%.16f\n',abs(P(k)-P(k-1)));
38      while k<=maxK
39          err=abs(P(k)-P(k-1));     %err表示相邻的迭代值的差值
40          if(err<tol)
41              fprintf('\n迭代%d次即可满足允许误差值退出\n',k-1);
42              break;
43          end
44          k=k+1;
45          P(k)=P(k-1)-subs(f,x,P(k-1))/subs(df,x,P(k-1));  %迭代
46          fprintf('\n第 %d 次, x=%.16f', k, P(k));
47          fprintf('    误差值err=%.16f\n', abs(P(k)-P(k-1)));
48
49      end       %共迭代了k-1次
50      if(k-1==maxK)
51          disp("超过最大迭代次数！");
52      end
53      p=P(k);
54      k=k-1;
```

```
第 2 次, x=0.5710204398084223    误差值err=0.0710204398084223

第 3 次, x=0.5671555687441144    误差值err=0.0038648710643079

第 4 次, x=0.5671432905332610    误差值err=0.0000122782108535

第 5 次, x=0.5671432904097838    误差值err=0.0000000001234771

第 6 次, x=0.5671432904097838    误差值err=0.0000000000000000

迭代5次即可满足允许误差值退出
使用牛顿迭代法法迭代 5 次，计算 x*exp(x) - 1=0 以 0.5 为迭代初始值的解为：0.567143
```

牛顿迭代法的收敛性问题

疑问： Newton法是否永远收敛呢？如果不是，那么与谁有直接关系呢？



结论： Newton法的收敛性依赖于x_0 的选取。

牛顿迭代法的收敛性问题

迭代函数

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

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

$$\varphi'(x_*) = 0, \quad \varphi''(x_*) = \frac{f''(x_*)}{f'(x_*)}$$

牛顿法至少二阶局部收敛

$$\lim_{k\to\infty} \frac{x_{k+1} - x_*}{(x_k - x_*)^2} = \frac{\varphi''(x_*)}{2!} = \frac{f''(x_*)}{2f'(x_*)}$$

牛顿迭代法的特点

优点: 收敛快，是目前求解非线性方程的主要方法

缺点: (1) 每一步都要计算$f(x_k)$和$f'(x_k)$，
程序常发生中断

(2) 初始近似x_0只在根x^*附近才能保证收敛

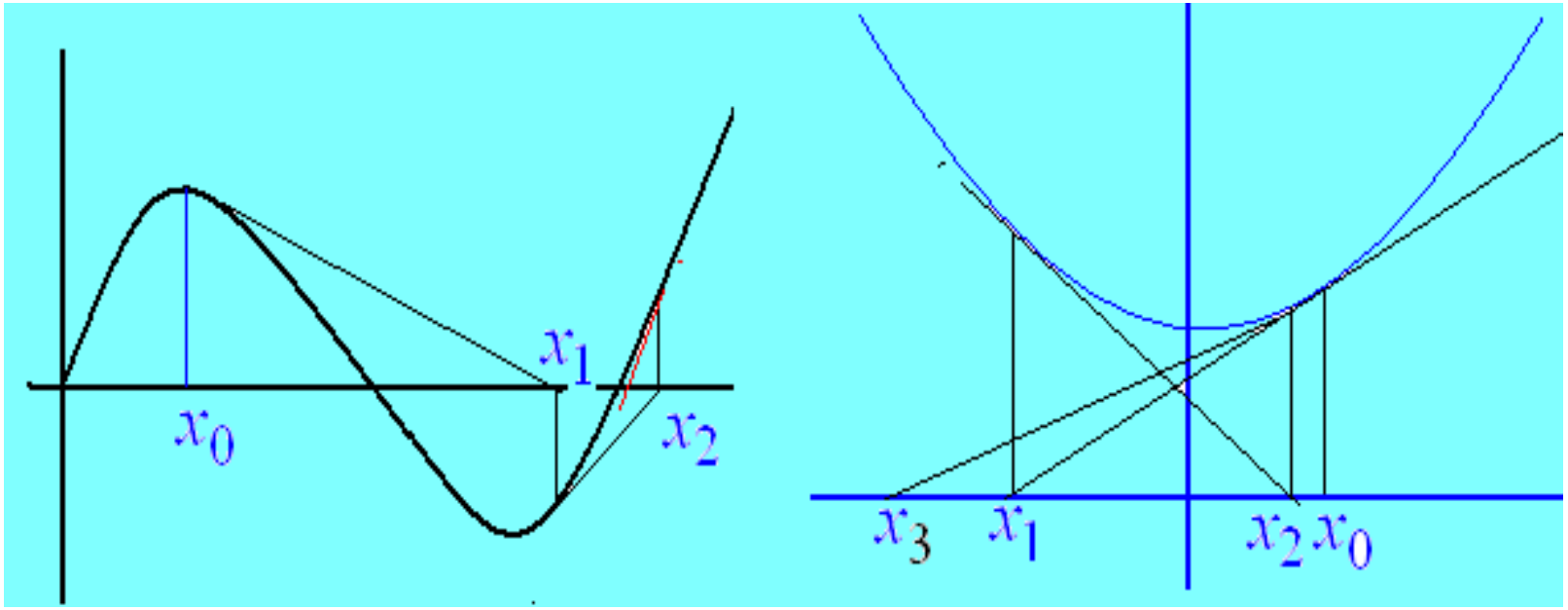
牛顿迭代法的不足

牛顿法的不足：

1）牛顿迭代法存在从一个根跳到另一个根的情况。

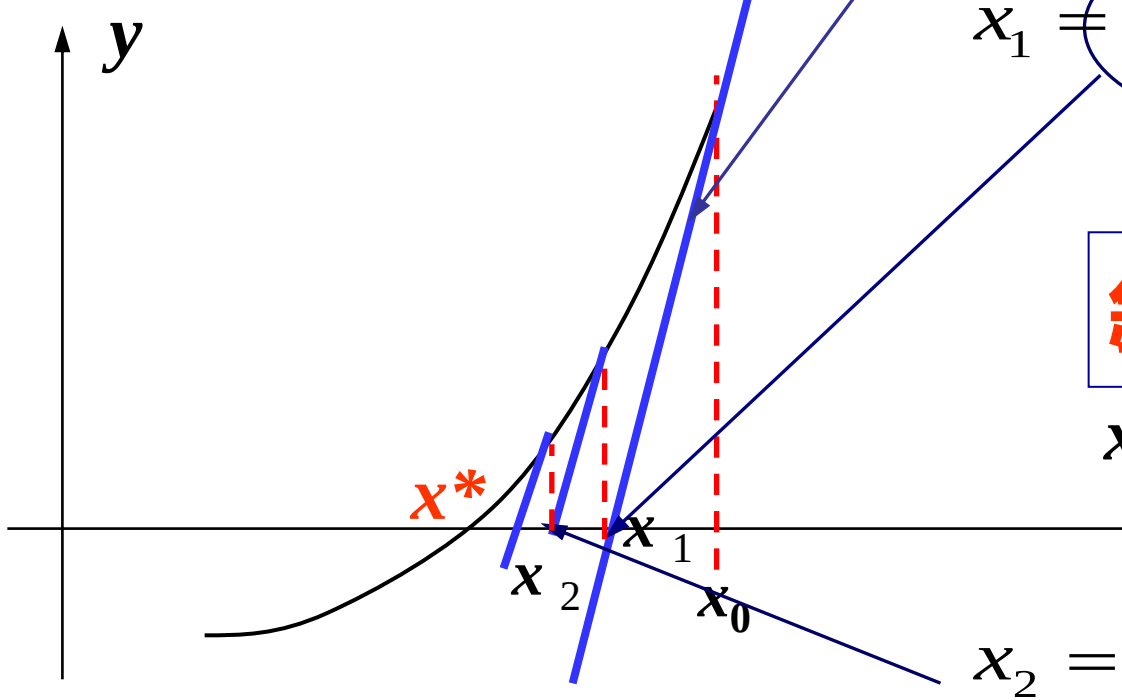
2）如果$f(x)=0$没有实根，则牛顿迭代序列不收敛。

3）每次迭代都要求导，程序易中断。



如何改进牛顿迭代法---简化的牛顿迭代法

$$Tangent\ line: y = f(x_0) + f'(x_0)(x - x_0)$$



$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

斜率恒定不变

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_0)}$$

简化牛顿法

- 出发点：减少导数计算次数
- 基本思想：用 $f'(x_0)$ 替代所有的 $f'(x_k)$

迭代格式

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)} \qquad k = 0, 1, 2, \ldots \ldots$$

- 好处：只需要计算一次导数
- 缺点：只有线性收敛速度（假定方法是收敛的）

简化牛顿法

例2. 用简化牛顿法解方程 $xe^x - 1 = 0.$

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

```
23  function [p,k,Y]=NTM(f,p0,tol,maxK)
24  %p0表示迭代初始值
25  %f表示要求解的方程
26  %maxK表示规定的最大迭代次数
27  %tolr表示允许误差
28  %k表示最终迭代的次数
29  %p表示最终迭代的值
30      syms x;
31      P(1)=p0;
32      k=2;
33      df=diff(f);   | %利用diff()函数计算f(x)的导数
34
35      P(k)=P(k-1)-subs(f,x,P(k-1))/subs(df,x,P(1)); %第二次迭代的结果
36
37      fprintf('\n第 %d 次, x=%.16f', k, P(k));
38      fprintf('    误差值err=%.16f\n',abs(P(k)-P(k-1)));
39      while k<=maxK
40          err=abs(P(k)-P(k-1));     %err表示相邻的迭代值的差值
41          if(err<tol)
42              fprintf('\n迭代%d次即可满足允许误差值退出\n',k-1);
43              break;
44          end
45          k=k+1;
46          P(k)=P(k-1)-subs(f,x,P(k-1))/subs(df,x,P(1));  %迭代
47
48          fprintf('\n第 %d 次, x=%.16f', k, P(k));
49          fprintf('    误差值err=%.16f\n', abs(P(k)-P(k-1)));
50
51      end      %共迭代了k-1次
52      if(k-1==maxK)
53          disp("超过最大迭代次数! ");
54      end
55      p=P(k);
56      k=k-1;
57      Y=P;
58  end
```

命令行窗口

```
第 2 次, x=0.5710204398084223    误差值err=0.0710204398084223

第 3 次, x=0.5666746434510392    误差值err=0.0043457963573831

第 4 次, x=0.5671980709245896    误差值err=0.0005234274735504

第 5 次, x=0.5671368608365182    误差值err=0.0000612100880715

第 6 次, x=0.5671440446867818    误差值err=0.0000071838502637

第 7 次, x=0.5671432019178033    误差值err=0.0000008427689785

第 8 次, x=0.5671433007916186    误差值err=0.0000000988738152

第 9 次, x=0.5671432891917912    误差值err=0.0000000115998273

第 10 次, x=0.5671432905526782    误差值err=0.0000000013608870

第 11 次, x=0.5671432903930196    误差值err=0.0000000001596586

第 12 次, x=0.5671432904117506    误差值err=0.0000000000187310

迭代11次即可满足允许误差值退出
使用简化牛顿迭代法法迭代 11 次，计算 x*exp(x) - 1=0 以 0.5 为迭代初始值的解为：0.567143

迭代值如下:
    0.5000    0.5710    0.5667    0.5672    0.5671    0.5671    0.5671    0.5671    0.5671    0.5671    0.5671    0.5671
```

牛顿重根法

如果x_0为$f(x)$的k重根，牛顿迭代公式修改为：

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

若三种方法都收敛，则一般重根法收敛最快，牛顿法次之，简化迭代法最慢。

牛顿迭代法的使用与比较

例 3

对方程$f(x)=(x-1)^3(x-2)=0$

1）取$x_0=0.9$, 用牛顿法计算原方程;

2）取$x_0=0.9$, 用牛顿重根法计算原方程;

3）取$x_0=0.9$, 用简化的牛顿法计算原方程。

解 三种迭代公式分别为 1) $x_{k+1}=x_k-\dfrac{f(x_k)}{f'(x_k)}$

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

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

牛顿迭代法的使用与比较

例 3

对方程$f(x)=(x-1)^3(x-2)=0$

三种迭代公式迭代结果分别为:

命令行窗口

```
第 5 次, x=0.9795042416467699    误差值err=0.0100979968195362

第 6 次, x=0.9862907275673437    误差值err=0.0067864859205738

第 7 次, x=0.9908399772245536    误差值err=0.0045492496572099

第 8 次, x=0.9938841077488724    误差值err=0.0030441305243187

第 9 次, x=0.9959186161001105    误差值err=0.0020345083512381

第 10 次, x=0.9972772365626771    误差值err=0.0013586204625666

第 11 次, x=0.9981840036390410    误差值err=0.0009067670763639

第 12 次, x=0.9987889702174818    误差值err=0.0006049665784408

第 13 次, x=0.9991924841195619    误差值err=0.0004035139020800

第 14 次, x=0.9994615837040898    误差值err=0.0002690995845279

第 15 次, x=0.9996410236155991    误差值err=0.0001794399115094

第 16 次, x=0.9997606680990222    误差值err=0.0001196444834231

第 17 次, x=0.9998404390369607    误差值err=0.0000797709379385

迭代16次即可满足允许误差值退出
使用牛顿迭代法法迭代 16 次，计算 (x - 1)^3*(x - 2)=0 以 0.9 为迭代初
```

牛顿迭代法

命令行窗口

```
第 2 次, x=0.9970588235294118    误差值err=0.0970588235294118

第 3 次, x=0.9999971277573529    误差值err=0.0029383042279412

第 4 次, x=0.9999999999972501    误差值err=0.0000028722398971

迭代3次即可满足允许误差值退出
使用牛顿重根迭代法法迭代 3 次，计算 (x - 1)^3*(x - 2)=0 以 0.9 为迭代初始值的解为：1

迭代值如下:
    0.9000    0.9971    1.0000    1.0000
```

牛顿重根法

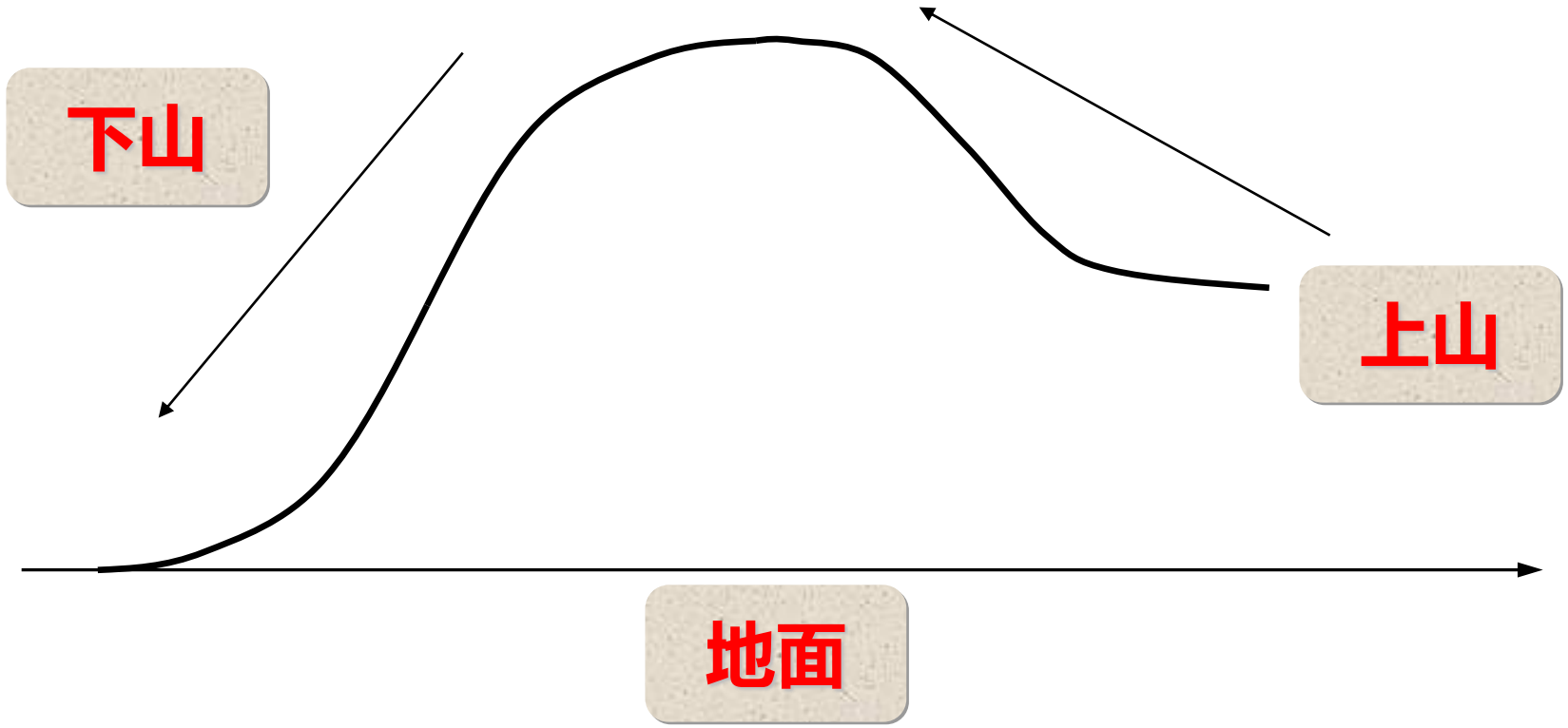
命令行窗口

```
第 65 次, x=0.9844930596453282    误差值err=0.0001138576774621

第 66 次, x=0.9846044332633520    误差值err=0.0001113736180238

第 67 次, x=0.9847134124062190    误差值err=0.0001089791428670

第 68 次, x=0.9848200821813514    误差值err=0.0001066697751324

第 69 次, x=0.9849245235037537    误差值err=0.0001044413224023

第 70 次, x=0.9850268133586236    误差值err=0.0001022898548699

第 71 次, x=0.9851270250440876    误差值err=0.0001002116854639

第 72 次, x=0.9852252283958186    误差值err=0.0000982033517311

迭代71次即可满足允许误差值退出
使用简化牛顿迭代法法迭代 71 次，计算 (x - 1)^3*(x - 2)=0 以 0.9 为迭代初始值的解为：0.985225

迭代值如下:
  列 1 至 28
```

简化牛顿迭代法

牛顿下山法

如果不收敛，能否继续改进牛顿法呢？



牛顿下山法

为了防止迭代发散，要求迭代满足单调性：

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

满足这项要求的算法称下山法.

思路

将牛顿法与下山法结合起来使用，即在下山法保证函数值稳定下降的前提下，用牛顿法加快收敛速度

为此，将牛顿法的计算结果

$$\bar{x}_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

与前一步的近似值x_k适当加权平均作为新的改进值

$$x_{k+1} = \lambda \bar{x}_{k+1} + (1-\lambda)x_k \ ,$$

牛顿下山法

迭代格式

$$x_{k+1} = x_k - \lambda \frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, 2, \ldots\ldots$$

- 其中 λ 是下山因子：从 λ=1 开始，逐次减半，直到满足下降条件为止

$$\text{取} \quad \lambda \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots \frac{1}{2^m}\}$$

直到满足:

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

牛顿下山法的程序实现

例 4

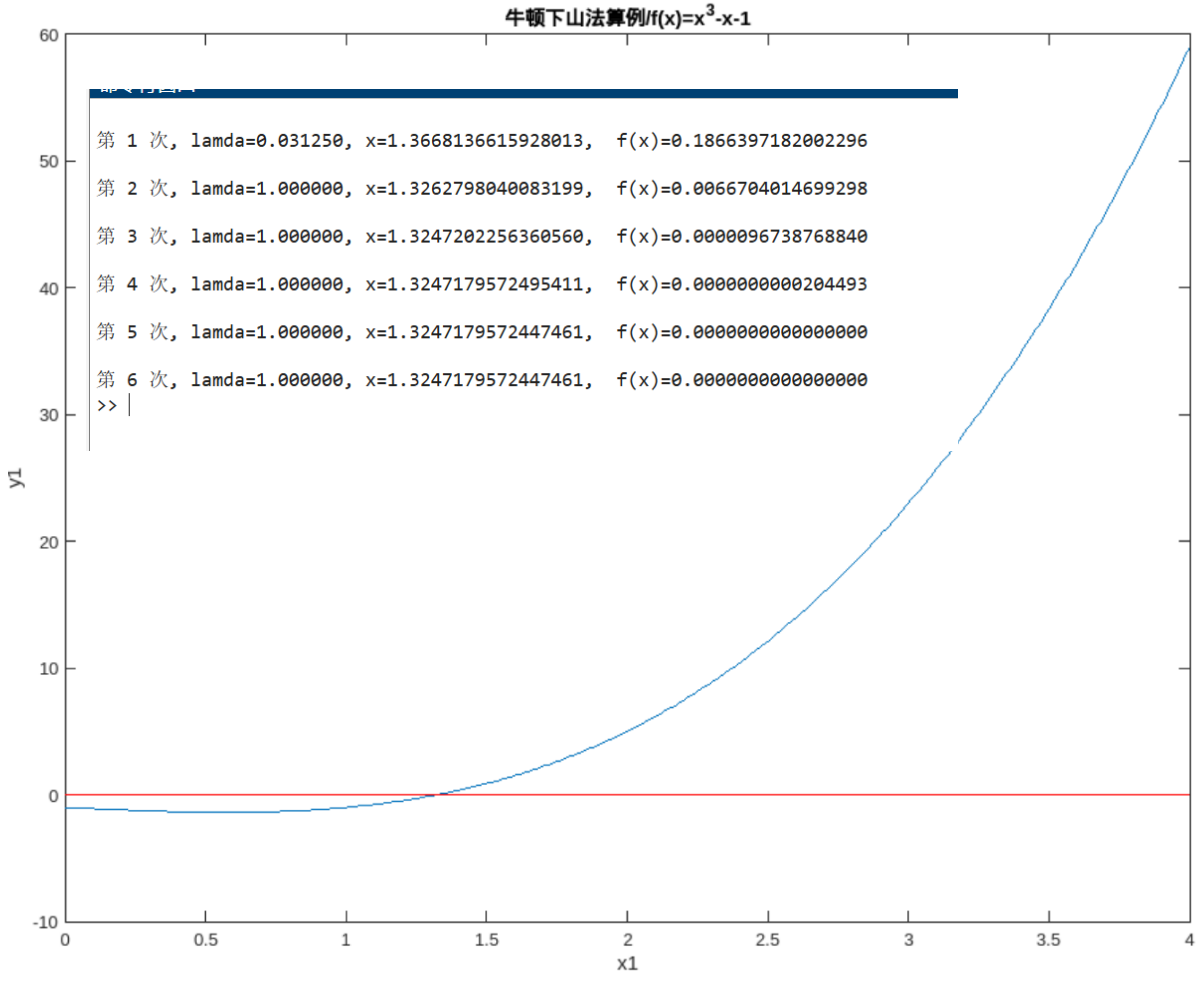
求方程 $f(x)=x^3-x-1=0$ 在$x_0=1.5$附近的根x^*.

/MATLAB Drive/Newton_downhill.m

```
clc
clear all;
syms x
f(x) = x.^3 - x - 1;
df(x) = 3*x.^2 - 1;
x1 = 0:0.01:4;
y1=x1.^3-x1-1;
plot(x1,y1);
xlabel('x1');%设置横轴坐标
ylabel('y1');%设置纵轴坐标
title('牛顿下山法算例/f(x)=x^3-x-1');
hold on
z=0*ones(1, length(x1));
plot(x1, z, 'r');
%p0=input("请输入牛顿迭代法的初始值p_0:   \n");
p0=0.6;
%计算精度
%tol=input("请输入精度E:   \n");
tol=1e-12;
x=Newton_downhill_1(f,df,p0,tol);

function x=Newton_downhill_1(fun, dfun, x0, EPS)
%fun 为迭代函数，dfun为迭代函数的一阶导数，x0为初始值，eps为精度
f=fcnchk(fun);
df=fcnchk(dfun);
x1=x0-f(x0)/df(x0);
d=norm(x1-x0);
k=1;
r=1;
while d>=EPS
    while abs(f(x1))>=abs(f(x0)) %迭代过程具有单调性检测
        r=r/2;
        x2=x0-r*f(x0)/df(x0);
        x1=x2;
    end
    x0=x1;
    x1=x0-f(x0)/df(x0);
    fprintf('\n第 %d 次, lamda=%f, x=%.16f,  f(x)=%.16f\n', k, r, x1,f(x1));
    d=norm(x1-x0);
    r=1;
    k=k+1;
end
if k ==1000
    x="iterative divergence";
else
    x=x1;
end
end
```

命令行窗口

```
第 1 次, lamda=0.031250, x=1.3668136615928013,  f(x)=0.1866397182002296

第 2 次, lamda=1.000000, x=1.3262798040083199,  f(x)=0.0066704014699298

第 3 次, lamda=1.000000, x=1.3247202256360560,  f(x)=0.0000096738768840

第 4 次, lamda=1.000000, x=1.3247179572495411,  f(x)=0.0000000000204493

第 5 次, lamda=1.000000, x=1.3247179572447461,  f(x)=0.0000000000000000

第 6 次, lamda=1.000000, x=1.3247179572447461,  f(x)=0.0000000000000000
>>
```



牛顿下山法的程序实现

例 4

求方程 $f(x) = x^3 - 5x + 1$ 的根。

初始值的选择的重要性

X0=-4 → X*=-2.330

X0=-2 → X*=-2.330

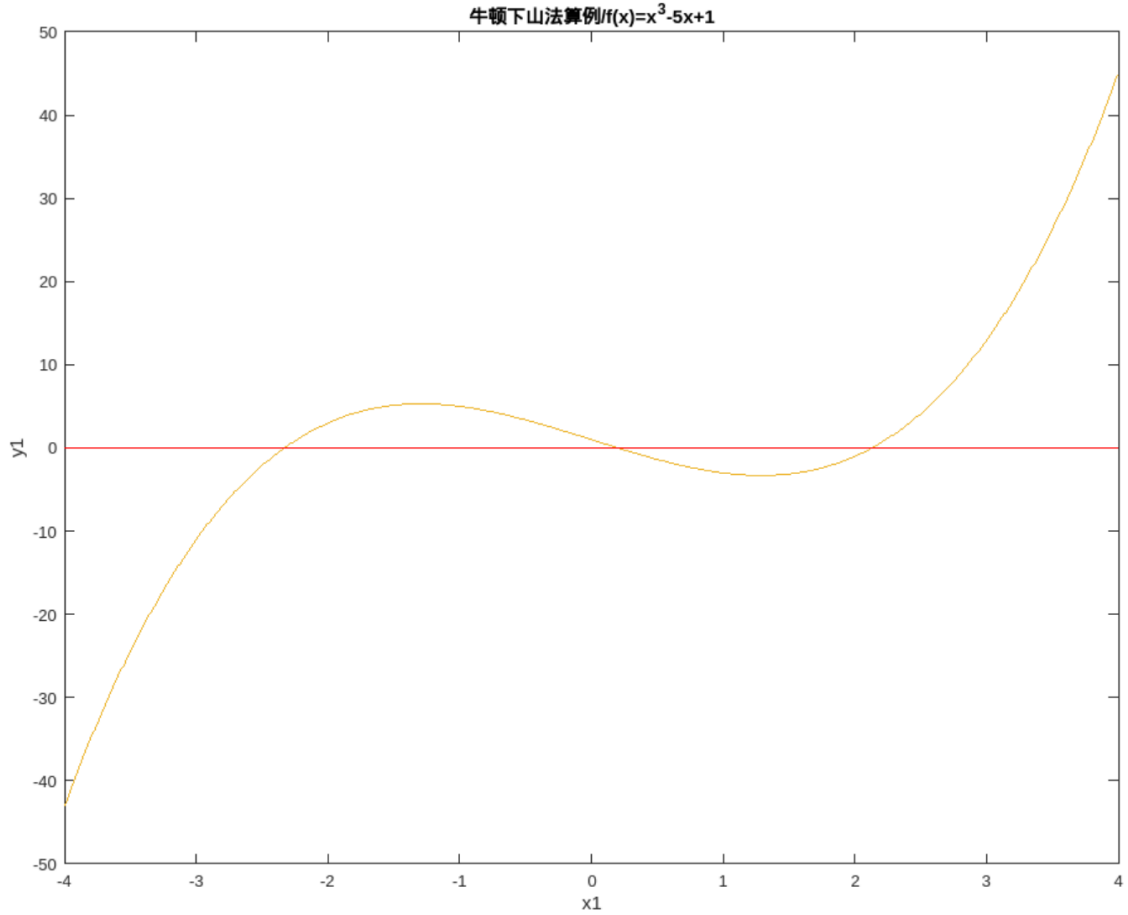
X0=-1 → X*=2.12

X0=0 → X*=0.20

X0=1 → X*=0.20

X0=1.5 → X*=2.12

X0=3 → X*=2.12



总结

- 牛顿迭代法：计算非线性方程根的重要方法，具有收敛速度快的优点，但是存在需要反复计算导数、对初值的选择依赖性较大的问题；$x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)}$
- 简化的牛顿迭代法：简化导数运算 $x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_0)}$
- 牛顿重根法：针对重根的计算，收敛性最快

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

- 牛顿下山法：确保**函数单调下降**，放宽了初值的选取范围

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

取 $\lambda \in \{1, \frac{1}{2}, \frac{1}{4}, \cdots \frac{1}{2^m}\}$

本章总结

- 非线性方程的求根问题
  - 根的存在判定（**零点定理**）
  - 求根方法的选择（注意每个方法的适用性、计算速率和收敛性）
    - 二分法
    - 不动点迭代法
    - 牛顿迭代法（含简化的牛顿迭代法和牛顿重根法）
    - 牛顿下山法
- 在实际应用牛顿迭代求解方程的根，往往分两步完成：先用**二分方法**得到足够准确的近似根；然后，再用**牛顿迭代法**进一步精确化。

谢谢大家！

肖　明

微电子科学与技术学院

邮箱：xiaom37@mail.sysu.edu.cn

电话：1781722706

办公室：中山大学珠海校区公共实验楼2楼A214室