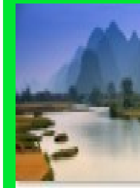
现代数值计算方法

第七章 非线性方程迭代解法



第七章 非线性方程迭代解法

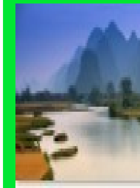
§7.3 牛顿型方法

§7.3.1 牛顿法的基本思想与算法

牛顿法是一种特殊形式的迭代法, 它是求解非线性方程最有效的方法之一. 其基本思想是: 利用泰勒公式将非线性函数在方程的某个近似根处展开, 然后截取其线性部分作为函数的一个近似, 通过解一个一元一次方程来获得原方程的一个新的近似根.

具体地说, 设当前点为 x_k, 将 $f(x)$ 在 x_k 处泰勒展开并截取线性部分得

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



Back
Close

令上式右端为 0, 解得

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots \tag{7.18}$$

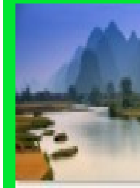
式 (7.18) 称为牛顿迭代公式.

根据导数的几何意义及上述推导过程可知, 牛顿法的几何上表现为: x_{k+1} 是函数 $f(x)$ 在点 $(x_k, f(x_k))$ 处的切线与 x 轴的交点. 因此, 牛顿法的本质是一个不断用切线来近似曲线的过程, 故牛顿法也称为切线法.

至于牛顿法的终止条件, 可以采用与简单迭代法相同的终止条件 (因牛顿法本身就是迭代法), 于是我们可以写出算法过程如下.

算法 7.5 (牛顿法)

步 1　取初始点 x_0, 最大迭代次数 N 和精度要求 ε, 置 $k := 0$;



步 2　计算

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

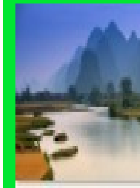
步 3　若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 4　若 $k = N$, 则停算; 否则, 置 $k := k + 1$, 转步 2.

根据算法 7.5, 编制 MATLAB 通用程序如下:

- 牛顿法 MATLAB 程序

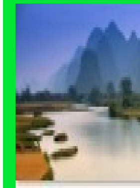
```
%manewton.m
function x=manewton(fun,dfun,x0,ep,N)
%用途: 用牛顿法求解非线性方程f(x)=0
%格式: x=manewton(fun,dfun,x0,ep,N)   fun和dfun分别示
%        f(x)的函数句柄, x0为迭代初值,ep为精度(默认1e-4),
%        N为最大迭代次数(默认为500),x返回近似根
```



```
if nargin<5,N=500;end
if nargin<4,ep=1e-4;end
k=0;
while k<N
    x=x0-feval(fun,x0)/feval(dfun,x0);
    if abs(x-x0)<ep  break; end
    x0=x; k=k+1;
end
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.9 用牛顿法程序 manewton.m 求方程 $f(x) = xe^x - 1 = 0$ 在 $[0, 1]$ 内的一个实根, 取初始点为 $x_0 = 0.5$, 精度为 10^{-5}.

解 在 MATLAB 命令窗口执行:



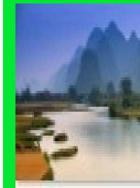
```
>> format long
>> fun=inline('x*exp(x)-1');
>> dfun=inline('(x+1)*exp(x)');
>> x=manewton(fun,dfun,0.5,1e-5)
```

得计算结果:

```
k=3
x =
   0.56714329040978
```

§7.3.2 牛顿法的收敛速度

现在我们来分析牛顿法的收敛性及收敛速度. 我们有



定理 7.5 设函数 $f(x)$ 二次连续可导, x^* 满足 $f(x^*)=0$ 及 $f'(x^*)\neq 0$, 则存在 $\delta>0$, 当 $x_0\in[x^*-\delta, x^*+\delta]$ 时, 牛顿法是收敛的, 且收敛阶至少是 2 (即至少是平方收敛的).

证 不难发现, 牛顿法本质相当于迭代函数为

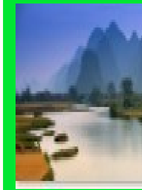
$$\varphi(x)=x-\frac{f(x)}{f'(x)}$$

的迭代法, 于是

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

由题设 $f(x^*)=0$ 及 $f'(x^*)\neq 0$ 可得 $\varphi'(x^*)=0$, 从而由定理 7.3 可知, 存在 $\delta>0$, 当 $x_0\in N(x^*,\delta)$ 时, 牛顿法收敛. 再由定理 7.4 可知, 其收敛阶至少是 2 阶的. □

由定理 7.5 的条件 ($f(x^*)=0$, $f'(x^*)\neq 0$) 可知, 当 x^* 是方程



$f(x)=0$ 的单根时, 收敛阶至少 2 阶的. 如果 x^* 是方程 $f(x)=0$ 的重根的情形会是怎么样的呢? 我们来做一些简单的分析.

设 x^* 是方程 $f(x)=0$ 的 m 重根, 即

$$f(x)=(x-x^*)^m g(x), \quad m\geq 2,$$

其中 $g(x)$ 有二阶导数且 $g(x^*)\neq 0$, 计算 $\varphi(x)=x-f(x)/f'(x)$ 的导数, 得

$$\varphi'(x)=\frac{\left(1-\dfrac{1}{m}\right)+(x-x^*)\dfrac{2g'(x)}{mg(x)}+(x-x^*)^2\dfrac{g''(x)}{m^2g(x)}}{\left[1+(x-x^*)\dfrac{2g'(x)}{mg(x)}\right]^2}$$

所以有

$$\varphi'(x^*)=1-\frac{1}{m}.$$

这样, 当 $m\geq 2$ 时, $\varphi'(x^*)\neq 0$, 且有 $|\varphi'(x)|<1$, 这样, 牛顿法就至多

只有线性收敛速度了. 这说明在重根的情形, 牛顿法失去了快速收敛的优点而变得不再实用.

为改善重根时牛顿法的收敛速度, 可以采用下面两种方法.

方法一: 当根的重数 $m \geq 2$ 时, 将迭代函数改为

$$\varphi(x) = x - \frac{mf(x)}{f'(x)}, \tag{7.19}$$

容易验证由上式定义的 $\varphi(x)$ 满足 $\varphi'(x^*) = 0$, 因此迭代公式

$$x_{k+1} = x_k - \frac{mf(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots \tag{7.20}$$

至少是二阶收敛的.

稍加思考我们便会发现上述加速方法并不适用, 因为事先并不知道根的重数 m, 故这一方法只具有理论上的意义, 下面的方法才是求重根时比较实用的加速方法.

方法二：若 x^* 是 $f(x)=0$ 的 m 重根, 则必为

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

的单根. 基于这个事实可以将牛顿迭代函数修改为

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

根据定理 7.5, 关于 $\mu(x)$ 的牛顿迭代公式

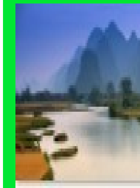
$$x_{k+1}=x_k-\frac{f(x_k)f'(x_k)}{[f'(x_k)]^2-f(x_k)f''(x_k)},\quad k=0,1,\cdots \tag{7.21}$$

至少是二阶收敛的. 公式 (7.21) 称为求重根的牛顿加速公式.

例 7.10 取初始点为 $x_0=1.5$, 分别用牛顿法和牛顿加速公式 (7.21) 计算方程

$$x^3-x^2-x+1=0$$

的根.



解 容易发现 $x=1$ 是二重根. 利用牛顿法的迭代公式为

$$x_{k+1} = x_k - \frac{x_k^2-1}{3x_k+1}, \tag{7.22}$$

而利用牛顿加速公式 (7.21) 的迭代公式为

$$x_{k+1} = \frac{x_k^2+6x_k+1}{3x_k^2+2x_k+3}. \tag{7.23}$$

利用公式 (7.22), 迭代 13 次得近似解为：$x_{13}=1.0001$. 而利用公式 (7.23) 迭代 3 次即可得到十分精确的结果：$x_1=0.96078$, $x_2=0.9996$, $x_3=1.0000$. 由此可见, 牛顿迭代加速公式 (7.21) 是有效的.

例 7.11 构造计算 $\sqrt[3]{c}$ 的牛顿迭代公式, 并用此公式计算 $\sqrt[3]{412}$ 的近似值, 精确到 10^{-8}.

解 令 $x=\sqrt[3]{c}$, 即求等价方程 $x^3-c=0$ 的根. 取 $f(x)=x^3-c$,

则 $f'(x)=3x^2$. 代入牛顿迭代公式有

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

计算 $\sqrt[3]{412}$ 时, 因 $f(7)<0$, $f(8)>0$, 故 $x^*\in[7,8]$. 取 $x_0=7$, 利用上述迭代公式进行计算, 计算结果下表.

k	x_k	k	x_k
0	7	3	7.441018862
1	7.469387755	4	7.441018861
2	7.441126470		

可以看出, $\sqrt[3]{412}$ 精确到 10^{-8} 的值为 7.44101886.

作业： P168: 7.14; P169: 7.18.