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

§7.3.3 阻尼牛顿法

一般来说, 牛顿法的收敛性依赖于初值 x_0 的选取, 如果 x_0 偏离 x^* 较远, 则牛顿法可能收敛缓慢甚至发散. 例如, 用牛顿法求方程 $x^3 - x - 1 = 0$ 的近似根, 如果取 $x_0 = 1.5$, 用牛顿迭代公式

$$x_{k+1} = x_k - \frac{x_k^3 - x_k - 1}{3x_k^2 - 1} \tag{7.24}$$

迭代 3 次可得结果: $x_1 = 1.3478$, $x_2 = 1.3252$, $x_3 = 1.3247$,其误差小于 10^{-5}. 但如果取 $x_0 = -2.0$, 则要得到同样精度的解需要迭代 65 次!

因此, 为了保证当 x_0 远离 x^* 时, 迭代仍然收敛, 可在牛顿迭代公式中增加一个参数 α, 改为

$$x_{k+1} = x_k - \alpha_k \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots \tag{7.25}$$

其中 α_k 的选择保证

$$|f(x_{k+1})| < |f(x_k)|. \tag{7.26}$$

公式 (7.25) 和 (7.26) 合起来称为阻尼牛顿法或牛顿下降法.

如何选择 α_k, 通常采用简单后退准则, 即取 $\rho = 0.5$, 记 m_k 是使下面不等式成立的最小非负整数 m:

$$\left| f\left(x_k - \rho^m f(x_k)/f'(x_k)\right)\right| < |f(x_k)|, \tag{7.27}$$

然后令 $\alpha_k = \rho^{m_k}$ 即可.

现写出阻尼牛顿法具体的算法步骤如下:

Back
Close

算法 7.6 (阻尼牛顿法)

步 1 取初始点 x_0, $\rho = 0.5$, 最大迭代次数 N 和精度要求 ε, 置 $k := 0$;

步 2 计算 $f(x_k)$ 及 $f'(x_k)$;

步 3 对于 $m = 0, 1, \cdots$, 检验下面的不等式:

$$\left|f\left(x_k - \rho^m f(x_k)/f'(x_k)\right)\right| < |f(x_k)|,$$

记 m_k 为使上述不等式成立的最小非负整数 m;

步 4 置

$$\alpha_k = \rho^{m_k}, \quad x_{k+1} = x_k - \alpha_k f(x_k)/f'(x_k);$$

步 5 若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 6 若 $k = N$, 则停算; 否则, 置 $k := k + 1$, 转步 2.

Back
Close

§7.3.4 离散牛顿法

用牛顿法或阻尼牛顿法解方程 $f(x) = 0$ 的优点是收敛速度快, 但牛顿法有一个明显的缺点：每次迭代除需计算函数值 $f(x_k)$, 还需计算导数 $f'(x_k)$ 的值, 如果 $f(x)$ 比较复杂, 计算 $f'(x_k)$ 就可能十分麻烦. 尤其当 $|f'(x_k)|$ 很小时, 计算需十分精确, 否则会产生较大的误差.

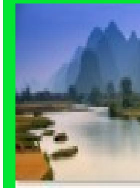
为避开计算导数, 可以改用差商 (离散形式) 代替导数 (连续形式), 即

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

得到牛顿迭代公式 (7.5) 的离散化形式

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}). \tag{7.28}$$

迭代公式 (7.28) 称为离散牛顿法或割线法. 可以证明下面的收敛定理.



定理 7.6 设函数 $f(x)$ 在其零点 x^* 的某个邻域 $S=\{x||x-x^*|\le\delta\}$ 内有二阶连续导数, 且对任意 $x\in S$, 有 $f'(x)\ne 0$, 则当 $\delta>0$ 充分小时, 对 S 中任意 x_0,x_1, 由离散牛顿迭代 (7.28) 产生的序列 $\{x_k\}$ 收敛到方程 $f(x)=0$ 的根 x^*, 且具有超线性收敛速度, 其收敛阶 $p\approx 1.618$.

证明可参见文献 [3] 第 348-350 页.

由于离散牛顿法不需要计算导数, 虽然收敛阶低于牛顿法, 但高于简单迭代法. 因此, 离散牛顿法在非线性方程的求解中得到广泛的应用, 也是工程计算中的常用方法之一.

综上所述, 离散牛顿法的计算步骤可归纳如下.

算法 7.7 (离散牛顿法)

步 1 取初始点 x_0,x_1, 最大迭代次数 N 和精度要求 ε, 置 $k:=0$;

步 2 计算 $f(x_k)$ 及 $f(x_{k-1})$;

步 3 置

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

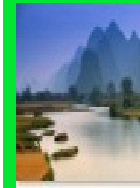
步 4 若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 5 若 $k = N$, 则停算; 否则, 置 $x_{k-1} := x_k$, $x_k := x_{k+1}$, $k := k+1$, 转步 2.

根据算法 7.7, 编制 MATLAB 通用程序如下:

- 离散牛顿法 MATLAB 程序

```
%maqnewt.m
function x=maqnewt(fun,x0,x1,ep,N)
%用途: 用离散牛顿法求解非线性方程f(x)=0
```



Back
Close

```
%格式: x=maqnewt(fun,,x0,x1,ep,N)  fun为表示f(x)的函数
%      句柄,x0,x1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
  temp=feval(fun,x1)-feval(fun,x0)
  x=x1-(x1-x0)*feval(fun,x1)/temp;
  if abs(x-x1)<ep
    break;
  end
  x0=x1; x1=x; k=k+1;
```

Back
Close

```
end
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.10 用离散牛顿法程序 maqnewt.m, 求方程 $f(x) = xe^x - 1 = 0$ 在 $[0,1]$ 内的一个实根, 取初始点为 $x_0 = 0.4$, $x_1 = 0.6$, 精度为 10^{-5}.

解 在 MATLAB 命令窗口执行:

```
>> x=maqnewt(inline('x*exp(x)-1'),0.4,0.6,1e-5)
```

得计算结果:

```
k=3
x =
   0.56714329035989
```

作业： P168: 7.11; 7.16.