程序设计 3

41704108 李达苇

(1)算法思想：

因为 Legendre 多项式在[-1,1]内有 n 个不同的实根，因此 n 次 Legendre 多项式可以表示成

$$P_n(x) = k(x-x_1)(x-x_2)\ldots(x-x_n)$$

在利用牛顿法求解出$P_n(x)$的第一个根x_1后，利用“压缩技术”，避免重复求根。记$P_{n-1}(x) = \frac{P_n(x)}{(x-x_1)}$，再次利用牛顿法求解$P_{n\text{-}1}(x)$的一个根，即得到$P_n(x)$的第二根。 由于$P'_{n-1}(x) = \frac{P'_n(x)}{x-x_1} - \frac{P_n(x)}{(x-x_1)^2}$，所以第二次迭代函数为

$$x_2 = x_0 - \frac{P_{n-1}(x_0)}{P'_{n-1}(x_0)} = x_0 - \frac{P_n(x_0)}{P'_n(x_0)} \frac{1}{1 - \frac{P_n(x_0)}{P'_n(x_0)} \frac{1}{x_0 - x_1}}$$

当我们已经求解出前 k 个根 $x_1, x_2, \ldots x_k$后,考虑

$$P_{n-k}(x) = \frac{P_n(x)}{(x-x_1)(x-x_2)\ldots(x-x_k)}$$

因为 $\frac{d}{dx}\left(\prod_{i=1}^{k}(x-x_i)^{-1}\right) = -\left(\prod(x-x_i)^{-1}\right)\left(\sum_{i=1}^{k}(x-x_i)^{-1}\right)$, 所以

$$P'_{n-k}(x) = \frac{P'_n(x)}{(x-x_1)(x-x_2)\ldots(x-x_k)} - \frac{P_n(x)}{(x-x_1)(x-x_2)\ldots(x-x_k)} \sum_{i=1}^{k} \frac{1}{x-x_i}$$

因此第 k+1 次迭代函数为

$$x_{k+1} = x_0 - \frac{P_n(x_0)}{P'_n(x_0)} \frac{1}{1 - \frac{P_n(x_0)}{P'_n(x_0)} \sum_{i=1}^{k} \frac{1}{x_0 - x_i}}$$

利用上面的迭代函数进行迭代,可以得到第 k+1 个根。如此循环下去，便可以得到 n 个根。

算法流程：

1. 利用 Legendre 多项式的三项递推公式

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), n = 1,2,\dots$$

$$P_0(x) = 1, P_1(x) = x$$

把$P_{30}(x)$与$P_{30}^{'}(x)$表示出来

2. 给定初始值x_0 ，计算$P_{30}(x_0)$与$P_{30}^{'}(x_0)$

3. 利用牛顿迭代法求出第一个根x_1，即迭代$x_1 = x_0 - \frac{P_{30}(x_0)}{P_{30}^{'}(x_0)}$ 。当 $|x_1 - x_0| < \varepsilon$(给定误差)时终止迭代，此时 x_1为所求的第一个根。

4. 循环：k =2，…，30. 迭代

$$x_k = x_0 - \frac{P_{30}(x_0)}{P_{30}^{'}(x_0)} \frac{1}{1 - \frac{P_{30}(x_0)}{P_{30}^{'}(x_0)} \sum_{i=1}^{k-1} \frac{1}{x_0 - x_i}}$$

当$|x_k - x_0| < \varepsilon$时终止迭代。 此时$x_k$为所求的第 k 个根

（2）Matlab 程序

```
function r=lratio(n,x)
p0=1; p0s=0; p1=x; p1s=1;
for k = 1:n-1
    p2 = (2*k+1)*x*p1/(k+1) - k*p0/(k+1);
    p2s = (2*k+1)*p1/(k+1) + (2*k+1)*x*p1s/(k+1) - k*p0s/(k+1);
    p0 = p1; p1 = p2;
    p0s = p1s; p1s = p2s;

    maxx = abs(p2)+abs(p2s);
    if maxx>1e20
        d=1e-20;
    elseif maxx<1e-20
        d=1e20;
    else
        d = 1;
    end
    p1=p1*d; p2=p2*d; p1s=p1s*d; p2s=p2s*d;
end
r=p2/p2s;
end
```

```
function root1 = Newton1(x0,tolerance)
x = x0 - lratio(30,x0);
while abs(x-x0) > tolerance
    x0 = x;
    x = x0 - lratio(30,x0);
end
x1 = x;
root1 = x1;
end
```

```
function Rroots = Newton2(x0,tolerance)
xi=[Newton1(x0,tolerance)];
for k = 2:30
    r = lratio(30,x0);
    s = sum(1./ (x0- xi(1:k-1)));
    x1 = x0 - r/(1-r*s);
    while abs(x1-x0) > tolerance
        x0 = x1;
        r = lratio(30,x0);
        s = sum(1./ (x0- xi(1:k-1)));
        x1 = x0 - r/(1-r*s);
    end
    xi(k) = x1;
    x0 = 1;
end
Rroots = xi;
end
```

```
Newton2(1, 0.00000005)
```

运行结果

在给定误差限为 0.00000005 的情况下

```
ans =

  1 至 3 列

   0.996893484074650   0.983668123279747   0.960021864968308

  4 至 6 列

   0.926200047429277   0.882560535792053   0.829565762382769

  7 至 9 列

   0.767777432104826   0.697850494793316   0.620526182989243

  10 至 12 列

   0.536624148142023   0.447033769538120   0.352704725530945

  13 至 15 列

   0.254636926167921   0.153869913608586   0.051471842555318

  16 至 18 列

  -0.051471842555318  -0.153869913608582  -0.254636926167890

  19 至 21 列

  -0.352704725530878  -0.447033769538089  -0.536624148142020

  22 至 24 列

  -0.620526182989243  -0.697850494793316  -0.767777432104806

  25 至 27 列

  -0.829565762382766  -0.882560535792053  -0.926200047429274

  28 至 30 列

  -0.960021864968307  -0.983668123279727  -0.996893484074650
```

所求得的 30 个根与下表相对应的值的误差不超过误差限

-0.996893484074650	-0.983668123279747	-0.960021864968308
-0.926200047429274	-0.882560535792053	-0.829565762382768
-0.767777432104826	-0.697850494793316	-0.620526182989243
-0.536624148142020	-0.447033769538089	-0.352704725530878
-0.254636926167890	-0.153869913608584	-0.0514718425553177
0.0514718425553177	0.153869913608584	0.254636926167890
0.352704725530878	0.447033769538089	0.536624148142020
0.620526182989243	0.697850494793316	0.767777432104826
0.829565762382768	0.882560535792053	0.926200047429274
0.960021864968308	0.983668123279747	0.996893484074650