科学计算作业 9

张文涛 517030910425

2018 年 12 月 10 日

1. 试确定常数 p, q, r，使迭代公式

$$x_{k+1} = px_k + \frac{qa}{x_k^2} + \frac{ra^2}{x_k^5}$$

产生的序列收敛到 $\sqrt[3]{a}$，并使收敛阶尽量高.

观察其有三个自由度，可以三界收敛

为了使产生的序列满足条件，可以列出以下方程求解：

$$\begin{cases} p + q + r = 1 \\ p - 2q - 5r = 0 \\ q + 5r = 0 \end{cases}$$

解得：

$$\begin{cases} p = \frac{5}{9} \\ q = \frac{5}{9} \\ r = -\frac{1}{9} \end{cases}$$

2. 设 x_* 是 $f(x) = 0$ 的单根, $f \in C^2(\mathbb{R})$，给定迭代法

$$\begin{cases} y_k = x_k - [f'(x_k)]^{-1} f(x_k) \\ x_{k+1} = y_k - [f'(x_k)]^{-1} f(y_k). \qquad k = 0, 1, \cdots \end{cases}$$

证明迭代法产生的序列 x_k 在 $x = x^*$ 处至少三阶局部收敛.

记 e_1, e_2, e_3 分别为 x_k, y_k, x_{k+1} 的误差。则原方程转化为：

$$\begin{cases} e_2 = e_1 - \frac{f(x_k)}{f'(x_k)} \\ e_3 = e_2 - \frac{f(y_k)}{f'(x_k)} \end{cases}$$

对 $f(x)$ 在 x^* 处 Taylor 展开，并计算 $f(x_k), f(y_k)$：

$$f(x_k) = f'(x^*)e_1 + \frac{f''(\xi_1)e_1^2}{2}$$
$$f(y_k) = f'(x^*)e_2 + \frac{f''(\xi_2)e_2^2}{2}$$

代入得：

$$e_2 = e_1 - \frac{f'(x^*)e_1 + \frac{f''(\xi_1)e_1^2}{2}}{f'(x_k)}$$

$$e_3 = e_2 - \frac{f'(x^*)e_2 + \frac{f''(\xi_2)e_2^2}{2}}{f'(x_k)}$$

消去 $f'(x_k)$，并解出 e_3 得到：

$$e_3 = \frac{f''(\xi_1)e_1^2e_2 + 2f'(x^*)e_2^2 - f''(\xi_2)e_2^2 - f''(\xi_2)e_2^2e_1 + f''(\xi_2)e_2^3}{2f'(x^*)e_1 + f''(\xi_1)e_1^2}$$

在 k 足够大时 $e_2 \approx Ae_1^2$，其中 $A = \frac{f''(x^*)}{f'(x^*)}$，代入得到：

$$e_3 = \frac{f''(\xi_1)Ae_1^3 + 2f'(x^*)A^2e_1^3 - f''(\xi_2)A^2e_1^3 - f''(\xi_2)A^2e_1^4 + f''(\xi_2)A^3e_1^5}{2f'(x^*) + f''(\xi_1)e_1}$$

容易看出：

$$\lim_{k\to\infty}\left|\frac{e_3}{e_1^2}\right| = 0$$

且：

$$\lim_{k\to\infty}\left|\frac{e_3}{e_1^3}\right| \neq 0$$

所以至少是三阶局部收敛。

3. 对 $\phi(x) = x + x^3\,, x = 0$ 为 ϕ 的一个不动点，验证 $x_{k+1} = \phi(x_k)$ 迭代对 $x_0 \neq 0$ 不收敛，但改用 Steffensen 方法则是收敛的.

$\forall x_p \neq 0$ ，代入得到 $x_p + x_p^3$，总有 $|x_p + x_p^3| > |x_p^3|$
因此显然不收敛。由 Steffensen 迭代法迭代公式得到：

$$\psi(x) = x - \frac{x^6}{(x+x^3)^3 - x^3} = x - \frac{x^3}{(x^2+1)^3 - 1} = x - \frac{x}{x^4 + 3x^2 + 3}$$

则求导得：

$$\psi'(x) = 1 + \frac{3(x^4 + x^2 - 1)}{(x^4 + 3x^2 + 3)^2}$$

可以计算得 $\psi'(0) = \frac{2}{3}$，因此存在 δ 在 $O(0,\delta)$ 中收敛。

4. Newton 迭代法中需要计算 $[f'(x)]^{-1}$，但数值微分是一个病态问题且高维情况计算代价较高. 若考虑

$$f'(x_k) \approx \frac{f(x_k + f(x_k)) - f(x_k - f(x_k))}{2f(x_k)},$$

将其代入 Newton 迭代法中，可得新的迭代法：

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

求证：若 x_* 是方程 $f(x) = 0$ 的单实根，且 $f'(x)$ 在 x_* 的某领域内连续，则以上迭代法在 x_* 附近至少是二阶收敛的.

在等式两边减去 x^*，得到：

$$e_{k+1} = e_k - \frac{2f^2(x_k)}{f(x_k + f(x_k)) - f(x_k - f(x_k))}$$

对 $f(x)$ 利用带 Peano 余项的 Taylor 展开得到：

$$e_{k+1} = e_k - \frac{2(f'(x^*))^2 e_k^2 + 4f'(x^*)e_k O(e_k^2) + (O(e_k^2))^2}{2(f'(x))^2 e_k + (f'(x^*) + 1)O(e_k^2)}$$

可以通过在分子中凑系数得到：

$$e_{k+1} = e_k - e_k + \frac{(3f'(x^*) - 1)e_k O(e_k^2) + (O(e_k^2))^2}{2(f'(x))^2 e_k + (f'(x^*) + 1)O(e_k^2)}$$

则有

$$\lim_{k\to\infty} \left| \frac{e_{k+1}}{e_k^2} \right| = \left| \frac{A}{f'(x^*) + 1} \right| \neq 0$$

且显然：

$$\lim_{k\to\infty} \left| \frac{e_{k+1}}{e_k^2} \right| = 0$$

因此至少二阶收敛。

5. 编写 MATLAB 程序求解非线性方程组：

$$f(x) = x^4 - 4x^2 + 4 = 0$$

给定容许误差为 $\epsilon = 10^{-8}$，初值 $x_0 = 1$.

(1) 利用 Newton 迭代法求解并记录每次迭代结果, 观察序列是否收敛, 若序列收敛, 指出收敛阶.

(2) 将 Newton 迭代法看作一种特定的不动点迭代方法, 利用 Aitken Δ^2 加速方法加速 Newton 迭代法, 得到 Steffensen 方法, 使用 Steffensen 方法求解并记录相关结果.

我们容易构造牛顿迭代方法：

$$\psi(x) = x - \frac{x^2 - 2}{4x}$$

可以验证：

$$\psi'(\sqrt{2}) = 1 - \frac{1}{2} = \frac{1}{2} \neq 0$$

因此是线性收敛的。

编写 MATLAB 程序进行牛顿迭代：

```
1  function res = Newton()
2  eps = 1e-8;
3  xk = 1;
4  cnt = 0;
```

```
5          xk1 = 0;
6          while(1)
7          xk1 = xk - (xk.^4 - 4.*xk.^2 + 4)/(4.*xk.^3 - 8.*xk);
8          if(abs(xk - xk1)< eps)
9          break;
10         end
11         cnt = cnt +1;
12         fprintf("result is %.9f after %d iteration.\n", xk1, cnt);
13         xk = xk1;
14         end
15         fprintf("iterate %d times, result is %.9f\n",cnt, xk1);
16         res = xk;
17         end
```

得到结果：

result is 1.250000000 after 1 iteration. result is 1.337500000 after 2 iteration.
result is 1.376956776 after 3 iteration. result is 1.395837186 after 4 iteration.
result is 1.405085856 after 5 iteration. result is 1.409664533 after 6 iteration.
result is 1.411942718 after 7 iteration. result is 1.413079053 after 8 iteration.
result is 1.413646535 after 9 iteration. result is 1.413930106 after 10 iteration.
result is 1.414071848 after 11 iteration. result is 1.414142709 after 12 iteration.
result is 1.414178137 after 13 iteration. result is 1.414195850 after 14 iteration.
result is 1.414204706 after 15 iteration. result is 1.414209134 after 16 iteration.
result is 1.414211348 after 17 iteration. result is 1.414212455 after 18 iteration.
result is 1.414213009 after 19 iteration. result is 1.414213286 after 20 iteration.
result is 1.414213424 after 21 iteration. result is 1.414213493 after 22 iteration.
result is 1.414213527 after 23 iteration. result is 1.414213546 after 24 iteration.
iterate 24 times, result is 1.414213552

利用 Setffensen 方法加速，编写程序得到：

```
1          function res = Setffensen()
2          eps = 1e-8;
3          xk = 1;
4          yk = 0;
5          zk = 0;
6          xk1 = 0;
7          cnt = 0;
8          while(1)
9          yk = xk - (xk.^4 - 4.*xk.^2 + 4)/(4.*xk.^3 - 8.*xk);
10         zk = yk - (yk.^4 - 4.*yk.^2 + 4)/(4.*yk.^3 - 8.*yk);
11         xk1 = xk - (yk - xk).^2/(zk -2.*yk+xk);
12         if(abs(xk1 - xk) < eps )
```

```
13          break;
14          end
15          cnt = cnt + 1;
16          fprintf('result is %.9f after %d iteration.\n', xk1, cnt);
17          xk = xk1;
18          end
19          fprintf('iterate %d times, result is %.9f\n',cnt, xk1);
20          res = xk1;
21          end
```

得到结果：

result is 1.384615385 after 1 iteration.

result is 1.414057935 after 2 iteration.

result is 1.414213558 after 3 iteration.

iterate 3 times, result is 1.414213556