六阶收敛的非线性方程组求根方法

课程：数值分析　　任课教师：黎卫兵/张雨浓

年级	2022	专业	计算机科学与技术
学号	22214373	姓名	林泽佳
班级	学硕 2 班	日期	2022 年 12 月 22 日

摘要

本文以张老师上课所介绍的构造 8 阶收敛的牛顿法，和 PPT 第 63 页简要介绍 Steffenson 和 Halley 方法为基础和启发，构造了 6 阶收敛的非线性方程组求根方法，并使用前向差商代替二阶导数，从而实现只需求一阶导数。本文使用泰勒展开证明了其收敛阶，充分的数值试验对比了本方法与牛顿法、Halley 法和若干现有的 6 阶方法。

1 引言

本节主要对张老师上课所介绍的构造 8 阶收敛的牛顿法，和 PPT 第 63 页简要介绍 Steffensen 和 Halley 方法做一个简要的回顾，作为本文的基础。

1.1 8 阶收敛的牛顿法

使用三次牛顿法反复代入，如公式1所示，可以得到序列 $\{y_n\}_{n=0\infty}$ 是 2 阶收敛，$\{z_n\}_{n=0\infty}$ 是 4 阶收敛，因此最终 $\{x_n\}_{n=0}^{\infty}$ 是 8 阶收敛的，这是一种简单的提高迭代公式阶数的方法。

$$
\begin{aligned}
y_n &= x_n - \frac{f(x_n)}{f'(x_n)} \\
z_n &= y_n - \frac{f(y_n)}{f'(y_n)} \\
x_{n+1} &= z_n - \frac{f(z_n)}{f'(z_n)}
\end{aligned}
\tag{1}
$$

1.2 Steffenson 方法

由于在零点附近 $\lim\limits_{n\to\infty} f(x_n)=0$，因此可以使用前向差商可以将零点附近的导数近似为：

$$f'(x) \approx \frac{f(x_n+f(x_n))}{f(x_n)} \tag{2}$$

将公式2代入牛顿迭代公式，可以得到：

$$x_{n+1}=x_n-\frac{f(x_n)^2}{f(x+f(x_n))-f(x_n)} \tag{3}$$

这就是 PPT 第 63 页所提到的 Steffensen 公式，它是也二阶收敛的 [1]，同时避免了求导。

1.3 Halley 方法

PPT 第 63 页同样提到了三阶收敛的 Halley 方法和 Chebyshev 方法，查阅文献 [2] 后发现，它们属于同一族方法，具有如下形式：

$$x_{n+1}=x_n-\left(1+\frac{1}{2}\frac{L_f(x_n)}{1-\beta L_f(x_n)}\right)\frac{f(x_n)}{f'(x_n)}, \tag{4}$$

其中

$$L_f(x_n)=\frac{1}{2}\frac{f''(x_n)f(x_n)}{[f'(x_n)]^2} \tag{5}$$

特别的，当 $\beta=0$ 时为 Chebyshev 方法，$\beta=\frac{1}{2}$ 时为 Halley 方法，$\beta=1$ 时为超 Halley 方法。不少研究对这族方法进行改进，但是它们都具有**较为复杂的迭代格式**，如 [3] 提出的一种无三阶收敛的方法：

$$\begin{cases} y_n &= x_n-\theta\dfrac{f(x_n)}{f'(x_n)}, \\ x_{n+1} &= x_n-\dfrac{\theta^2 f(x_n)}{(\theta^2-\theta+1)f(x_n)-f(y_n)}\dfrac{f(x_n)}{f'(x_n)}, \quad (\theta\in\mathbb{R},\theta\neq 0), \end{cases} \tag{6}$$

又如 [4] 提出的另一种三阶收敛的方法：

$$\begin{cases} w_n &= x_n-\dfrac{f(x_n)}{f'(x_n)}, \\ K_\alpha(x_n) &= \dfrac{2f(x_n)f(w_n)(1+\alpha f'^2(x_n))}{f^2(x_n)+\alpha f'^2(x_n)[f(w_n)-f(x_n)]^2}, \quad (\alpha\in\mathbb{R}), \\ x_n+1 &= x_n-\left(1+\dfrac{1}{2}\dfrac{K_\alpha(x_n)}{1-\beta K_\alpha(x_n)}\right)\dfrac{f(x_n)}{f'(x_n)}, \quad (\beta\in\mathbb{R}), \end{cases} \tag{7}$$

这些方法与同为三阶收敛的其它方法相比，或多或少能减少一些迭代次数，但代价是较为复杂的迭代格式。

1.4 六阶收敛的方法

在三阶收敛方法的基础上，再使用一次牛顿迭代或其它二阶迭代法，即可实现六阶收敛的方法。如开山鼻祖 [5] 中提出的六阶收敛的一族方法：

$$\begin{cases} w_n &= x_n - \dfrac{f(x_n)}{f'(x_n)}, \\ z_n &= w_n - \dfrac{f(w_n)}{f'(x_n)} \dfrac{f(x_n)+\beta f(w_n)}{f(x_n)+(\beta-2)f(w_n)}, \\ x_{n+1} &= z_n - \dfrac{f(w_n)}{f'(x_n)} \dfrac{f(x_n)-f(w_n)+\gamma f(z_n)}{f(x_n)-3f(w_n)+\lambda f(z_n)} \end{cases} \tag{8}$$

又如 [6] 中对 Ostrowski 方法 [7] 改进提出的六阶收敛的方法：

$$\begin{cases} y_n &= x_n - \dfrac{f(x_n)}{f'(x_n)}, \\ z_n &= y_n - \dfrac{y_n - x_n}{2f(y_n)-f(x_n)} f(y_n), \\ x_{n+1} &= z_n - \dfrac{y_n - x_n}{2f(y_n)-f(x_n)} f(z_n), \end{cases} \tag{9}$$

又如 [8] 中提出的用前向差商代替导数的六阶收敛的方法：

$$\begin{cases} w_n &= x_n + f(x_n), \\ y_n &= x_n - \dfrac{f(x_n)}{f[x_n, w_n]}, \\ z_n &= x_n - \dfrac{f(x_n)}{f[x_n, w_n]} \left(1 + \dfrac{f(y_n)}{f(x_n)} \left(1 + 2\dfrac{f(y_n)}{f(x_n)}\right)\right), \\ x_{n+1} &= z_n - \dfrac{f(z_n)}{f[y_n, z_n]} \left(1 - \dfrac{1+f[x_n, w_n]f(z_n)}{f[x_n, w_n]f(w_n)}\right) \end{cases} \tag{10}$$

除此之外，[9][10][11] 等均提出了不同形式的六阶收敛的方法，虽然它们的计算效率较高，但均具有较为复杂的迭代格式。本文希望能建立一种较为简单的迭代格式对 Halley 方法进行改造，在避免求二阶导数的基础上实现六阶收敛的方法。**诚信声明，在我所了解到的文献里没有与本文方法相同的，虽然推导和证明过程均有参考它们，也有相应的引用，但皆为原创和独立完成；如有雷同，是我查找资料不周全，没有恶意抄袭的意思。**

2 方法设计

2.1 方法推导

在三阶的 Halley 方法的基础上再进行一次牛顿法迭代得到的序列，显然是六阶收敛的：

$$\begin{cases} z_n &= x_n - \dfrac{2f(x_n)f'(x_n)}{2[f'(x_n)]^2 - f(x_n)f''(x_n)}, \\ x_{n+1} &= x_n - \dfrac{f(z_n)}{f'(z_n)} \end{cases} \tag{11}$$

参考 [12] 中的方法,希望消去 $\{z_n\}_{n=0}^{\infty}$ 中的二阶导数,如果记 $y_n = x_n - \frac{f(x)}{f'(x)}$,则有 $\lim_{n\to\infty} \frac{y_n}{x_n} = 1$，因此二阶导数可以使用一阶导计算前向差商近似为:

$$
\begin{aligned}
f''(x_n) &= \lim_{n\to\infty} \frac{f'(y_n) - f'(x_n)}{y_n - x_n} \\
&\approx \frac{f'(y_n) - f'(x_n)}{y_n - x_n},
\end{aligned} \tag{12}
$$

将公式12代入公式11，使用 sympy[1]（代码见附录 1）化简得:

$$
\begin{cases}
y_n &= x_n - \dfrac{f(x)}{f'(x)} \\
z_n &= x_n - \dfrac{2f(x_n)}{f'(x_n) + f'(y_n)} \\
x_{n+1} &= z_n - \dfrac{f(z_n)}{f'(z_n)}
\end{cases} \tag{13}
$$

公式13即为本文所提出的方法，由于它使用了差商代替二阶导数，因此需要对收敛阶进行证明。

2.2 收敛阶证明

我参考了 [3] 和 [12] 使用的技巧对公式13的收敛阶进行证明，由于涉及到 6 阶泰勒展开，计算较为复杂，因此我使用了 sympy 进行符号计算和化简，代码在附录 2 中。

假设函数 $f(x)$ 的单根是 α 且 $f'(\alpha) \neq 0$，记截断误差 $e_n = \alpha - x_n$，对 $f(x_n)$ 在 α 处进行泰勒展开得:

$$
\begin{aligned}
f(x_n) &= f(\alpha + e_n) \\
&= f(\alpha) + f'(\alpha)e_n + \frac{1}{2}f''(\alpha)e_n^2 + \frac{1}{3!}f'''(\alpha)e_n^3 + O(e_n^4) \\
&= f'(\alpha)\left[e_n + \frac{f''(\alpha)e_n^2}{2f'(\alpha)} + \frac{f'''(\alpha)e_n^3}{3!f'(\alpha)} + O(e_n^4)\right],
\end{aligned} \tag{14}
$$

注意到 $f(\alpha) = 0$，所以公式中将它消去了，记 $c_k = \frac{f^{(k)}(\alpha)}{f'(\alpha)k!}$，则有

$$
f(x_n) = f'(\alpha)\left[c_1 e_n + c_2 e_n^2 + c_3 e_n^3 + O(e_n^4)\right], \tag{15}
$$

同理对 $f'(x_n)$ 在 α 处进行泰勒展开得:

$$
\begin{aligned}
f'(x_n) &= f'(\alpha + e_n) \\
&= f'(\alpha) + f''(\alpha)e_n + \frac{1}{2}f'''(\alpha)e_n^2 + O(e_n^3), \\
&= f'(\alpha)[1 + 2c_2 e_n + 3c_3 e_n^2 + O(e_n^3)]
\end{aligned} \tag{16}
$$

[1]sympy 是 Python 的符号计算库，https://www.sympy.org/en/index.html

希望得到 y_n 的泰勒展开形式，因此首先需要计算：

$$\frac{f(x_n)}{f'(x_n)} = \frac{e_n + c_2e_n^2 + c_3e_n^3 + O(e_n^4)}{1 + 2c_2e_n + 3c_3e_n^2 + O(e_n^3)}, \tag{17}$$

记 $\theta = 2c_2e_n + 3c_3e_n^2 + O(e_n^3)$，显然 $\lim\limits_{n\to\infty} \theta = 0$，因此在 $\theta = 0$ 处对分母进行泰勒展开有：

$$\begin{aligned}
\frac{1}{1+\theta} &= 1 - \theta + \theta^2 - \theta^3 + O(\theta^4) \\
&= 1 - [2c_2e_n + 3c_3e_n^2 + O(e_n^3)] + [2c_2e_n + 3c_3e_n^2 + O(e_n^3)]^2 \\
&\qquad + [2c_2e_n + 3c_3e_n^2 + O(e_n^3)]^3 + O(e_n^4) \\
&= 1 - (2c_2e_n + 3c_3e_n^2) + 4c_2^2e_n^2 + 8c_2^3e_n^3 + O(e_n^4) \\
&= 1 - 2c_2e_n + (4c_2^2 - 3c_3)e_n^2 + 8c_2^3e_n^3 + O(e_n^4)
\end{aligned} \tag{18}$$

将公式18代入公式17并化简，由于化简过程较为复杂，为了保证准确性，再次使用 sympy 进行验证（代码见附录 2）：

$$\begin{aligned}
\frac{f(x_n)}{f'(x_n)} &= [e_n + c_2e_n^2 + c_3e_n^3 + O(e_n^4)] \cdot [1 - 2c_2e_n + (4c_2^2 - 3c_3)e_n^2 + 8c_2^3e_n^3 + O(e_n^4)] \\
&= e_n - c_2e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)
\end{aligned} \tag{19}$$

因此求得 y_n 的泰勒展开：

$$\begin{aligned}
y_n &= x_n - \frac{f(x_n)}{f'(x_n)} \\
&= \alpha + e_n - [e_n - c_2e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)] \\
&= \alpha + c_2e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)
\end{aligned} \tag{20}$$

于是 y_n 的截断误差为 $c_2e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)$，对 $f'(y_n)$ 在 α 处做泰勒展开有：

$$\begin{aligned}
f'(y_n) &= f'(\alpha) + [c_2e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)]f''(\alpha) + O(e_n^4) \\
&= f'(\alpha)\left[1 + \frac{2c_2e_n^2 + 4(c_3 - c_2^2)e_n^3 + O(e_n^4)}{2f'(\alpha)} f''(\alpha)\right] \\
&= f'(\alpha)[1 + 2c_2^2e_n^2 + 4c_2(c_3 - c_2^2)e_n^3 + O(e_n^4)],
\end{aligned} \tag{21}$$

将公式16与21相加得到 z_n 的分母部分的泰勒展开：

$$f'(x_n) + f'(y_n) = 2f'(\alpha)\left[1 + c_2e_n + \left(c_2^2 + \frac{3}{2}c_3\right)e_n^2 + O(e_n^3)\right], \tag{22}$$

因此将公式22与14相除得到 z_n 的泰勒展开：

$$x_n - \frac{2f(x_n)}{f'(x_n) + f'(y_n)} = \alpha + e_n - \frac{e_n + c_2e_n^2 + c_3e_n^3 + O(e_n^4)}{1 + c_2e_n + \left(c_2^2 + \frac{3}{2}c_3\right)e_n^2 + O(e_n^3)} \tag{23}$$

使用与公式18相似的技巧，对分母部分进行泰勒展开得：

$$
\begin{aligned}
x_n - \frac{2f(x_n)}{f'(x_n)+f'(y_n)} &= \alpha + e_n - [e_n + c_2e_n^2 + c_3e_n^3 + O(e_n^4)] \\
&\quad \left\{1 - \left[c_2e_n + \left(c_2^2 + \frac{3}{2}c_3\right)e_n^2 + O(e_n^3)\right]\right. \\
&\quad \left. + \left[c_2e_n + \left(c_2^2 + \frac{3}{2}c_3\right)e_n^2 + O(e_n^3)\right]^2 + O(e_n^4)\right\} \\
&= \alpha + e_n - [e_n + c_2e_n^2 + c_3e_n^3 + O(e_n^4)]\left[1 - c_2e_n - \frac{3}{2}c_3e_n^2 + O(e_n^3)\right] \\
&= \alpha + \left(c_2^2 + \frac{1}{2}c_3\right)e_n^3 + O(e_n^4)
\end{aligned} \tag{24}
$$

可以得到最终 z_n 的误差：

$$
z_n - \alpha = \left(c_2^2 + \frac{1}{2}c_3\right)e_n^3 + O(e_n^4) \tag{25}
$$

因此 z_n 是三阶收敛的，由张老师上课所介绍的 8 阶牛顿法的技巧可知，由于 x_{n+1} 是在 z_n 的基础上使用一次牛顿迭代，因此 x_n 是六阶收敛的。

2.3 方法的扩展

3 实验

4 总结

参考文献

[1] Pankaj Jain. Steffensen type methods for solving non-linear equations. *Applied Mathematics and Computation*, 194(2):527–533, 2007.

[2] Jose Manuel Gutierrez and Miguel A Hernández. A family of chebyshev-halley type methods in banach spaces. *Bulletin of the Australian Mathematical Society*, 55(1):113–130, 1997.

[3] Jisheng Kou, Yitian Li, and Xiuhua Wang. Modified halley's method free from second derivative. *Applied Mathematics and Computation*, 183(1):704–708, 2006.

[4] Changbum Chun. Some second-derivative-free variants of chebyshev–halley methods. *Applied Mathematics and Computation*, 191(2):410–414, 2007.

[5] Beny Neta. A sixth-order family of methods for nonlinear equations. *Int. J. Comput. Math*, 7(1997):157–161, 1979.

[6] Miquel Grau and José Luis Díaz-Barrero. An improvement to ostrowski root-finding method. *Applied Mathematics and Computation*, 173(1):450–456, 2006.

[7] AS Householder. Solution of equations and systems of equations (am ostrowski). *SIAM Review*, 9(3):608, 1967.

[8] F Soleymani and V Hosseinabadi. New third-and sixth-order derivative-free techniques for nonlinear equations. *Journal of Mathematics Research*, 3(2):107, 2011.

[9] Changbum Chun and Beny Neta. A new sixth-order scheme for nonlinear equations. *Applied mathematics letters*, 25(2):185–189, 2012.

[10] Obadah Said Solaiman and Ishak Hashim. Two new efficient sixth order iterative methods for solving nonlinear equations. *Journal of King Saud University-Science*, 31(4):701–705, 2019.

[11] Mona Narang, Saurabh Bhatia, and Vinay Kanwar. New two-parameter chebyshev–halley-like family of fourth and sixth-order methods for systems of nonlinear equations. *Applied Mathematics and Computation*, 275:394–403, 2016.

[12] Tahereh Eftekhari. A new sixth-order steffensen-type iterative method for solving nonlinear equations. *International Journal of Analysis*, 2014, 2014.

附录

附录 1

```python
import sympy as sp


fx = sp.symbols('fx')        # f(x)
dx = sp.symbols('dx')        # f'(x)
x = sp.symbols('x')          # x
dy = sp.symbols('dy')        # f'(y)


y = x - fx / dx              # y
ddx = (dy - dx) / (y - x)    # f''(x)
halley = x - (2 * fx * dx) / (2 * dx * dx - fx * ddx)


print(sp.simplify(halley - x))
```

附录 2

```python
import sympy as sp


e = sp.symbols('e')
e1 = e
e2 = e ** 2
e3 = e ** 3
e4 = e ** 4
c2, c3 = sp.symbols('c2 c3')


div = (e1 + c2*e2 + c3*e3 + e4) * (1 - 2*c2*e1 + (4*(c2**2) - 3*c3)*e2 + 8*(c2**3)*e3 + e4)


print(sp.latex(sp.simplify(sp.expand(div))))
```