

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

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

将公式2代入牛顿迭代公式, 可以得到:

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

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:

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

其中

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

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:

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

又如 [4] 提出的另一种三阶收敛的方法:

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

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。如最早由 [5] 提出的六阶收敛方法:

$$\begin{cases} w_n &= x_n - \frac{f(x_n)}{f'(x_n)}, \\ z_n &= w_n - \frac{f(w_n)}{f'(x_n)} \cdot \frac{f(x_n) - \frac{1}{2}f(w_n)}{f(x_n) - \frac{5}{2}f(w_n)}, \\ x_{n+1} &= z_n - \frac{f(z_n)}{f'(x_n)} \cdot \frac{f(x_n) - f(w_n)}{f(x_n) - 3f(w_n)} \end{cases} \quad (8)$$

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

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

又如 [8] 中提出的用前向差商代替导数的六阶收敛的方法:

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

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:

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

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 $\lim_{n \rightarrow \infty} \frac{y_n}{x_n} = 1$, 因此二阶导数可以使用一阶导计算前向差商近似为:

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

将公式12代入公式11, 使用 sympy¹ (代码见附录 1) 化简得:

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

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 阶泰勒展开的计算较为复杂, 因此我使用了 sympy¹ 进行符号计算和化简, 代码在附录 2 和 `code/convergence.py` 中, 本部分所推导的结果皆由该代码产生, 其正确性保证了本文推导结果的正确。由于篇幅有限, 部分泰勒展开用省略号表示。

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:

$$\begin{aligned} f(x_n) &= f(\alpha + e_n) \\ &= f(\alpha) + f'(\alpha)e_n + \frac{1}{2}f''(\alpha)e_n^2 + \frac{1}{3!}f'''(\alpha)e_n^3 + \cdots + O(e_n^7) \\ &= f'(\alpha) \left[e_n + \frac{f''(\alpha)e_n^2}{2f'(\alpha)} + \frac{f'''(\alpha)e_n^3}{3!f'(\alpha)} + \cdots + O(e_n^7) \right], \end{aligned} \quad (14)$$

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

$$\begin{aligned} f(x_n) &= f'(\alpha) [c_1e_n + c_2e_n^2 + c_3e_n^3 + c_4e_n^4 + c_5e_n^5 + c_6e_n^6 + O(e_n^7)] \\ &= f'(\alpha) \left[\sum_{k=1}^6 c_k e_n^k + O(e_n^7) \right] \end{aligned} \quad (15)$$

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

同理对 $f'(x_n)$ 在 α 处进行泰勒展开得:

$$\begin{aligned}
 f'(x_n) &= f'(\alpha + e_n) \\
 &= f'(\alpha) + f''(\alpha)e_n + \frac{1}{2}f'''(\alpha)e_n^2 + \cdots + O(e_n^6) \\
 &= f'(\alpha)[1 + 2c_2e_n + 3c_3e_n^2 + 4c_4e_n^3 + 5c_5e_n^4 + 6c_6e_n^5 + O(e_n^6)] \\
 &= f'(\alpha) \left[1 + \sum_{k=2}^6 kc_k e_n^{k-1} + O(e_n^6) \right]
 \end{aligned} \tag{16}$$

希望得到 y_n 的泰勒展开形式, 因此首先需要计算:

$$\frac{f(x_n)}{f'(x_n)} = \frac{\sum_{k=1}^6 c_k e_n^k + O(e_n^7)}{1 + \sum_{k=2}^6 kc_k e_n^{k-1} + O(e_n^6)}, \tag{17}$$

记 $\theta = 1 + \sum_{k=2}^6 kc_k e_n^{k-1} + O(e_n^6)$, 显然 $\lim_{n \rightarrow \infty} \theta = 0$, 因此在 $\theta = 0$ 处对分母进行泰勒展开有 (代码第 61 行):

$$\begin{aligned}
 \frac{1}{1+\theta} &= 1 - \theta - 2c_2e + \theta^2 - \theta^3 + \theta^4 - \theta^5 + O(\theta^6) \\
 &= 1 + e_n^2 \cdot (4c_2^2 - 3c_3) + e_n^3 (-8c_2^3 + 12c_2c_3 - 4c_4) \\
 &\quad + e_n^4 \cdot (16c_2^4 - 36c_2^2c_3 + 16c_2c_4 + 9c_3^2 - 5c_5) \\
 &\quad + e_n^5 (-32c_2^5 + 96c_2^3c_3 - 48c_2^2c_4 - 54c_2c_3^2 + 20c_2c_5 + 24c_3c_4 - 6c_6) + O(e_n^6),
 \end{aligned} \tag{18}$$

将公式18代入公式17并化简有 (代码第 65 行):

$$\begin{aligned}
 \frac{f(x_n)}{f'(x_n)} &= e - c_2e_n^2 + e_n^3 \cdot (2c_2^2 - 2c_3) + e_n^4 \cdot (-4c_2^3 + 7c_2c_3 - 3c_4) \\
 &\quad + e_n^5 \cdot (8c_2^4 - 20c_2^2c_3 + 10c_2c_4 + 6c_3^2 - 4c_5) \\
 &\quad + e_n^6 \cdot (-16c_2^5 + 52c_2^3c_3 - 28c_2^2c_4 - 33c_2c_3^2 + 13c_2c_5 + 17c_3c_4 - 5c_6) + O(e_n^7),
 \end{aligned} \tag{19}$$

因此求得 y_n 的泰勒展开 (代码第 70 行):

$$\begin{aligned}
 y_n &= x_n - \frac{f(x_n)}{f'(x_n)} \\
 &= \alpha + c_2e_n^2 + e_n^3 \cdot (-2c_2^2 + 2c_3) + e_n^4 \cdot (4c_2^3 - 7c_2c_3 + 3c_4) \\
 &\quad + e_n^5 \cdot (-8c_2^4 + 20c_2^2c_3 - 10c_2c_4 - 6c_3^2 + 4c_5) \\
 &\quad + e_n^6 \cdot (16c_2^5 - 52c_2^3c_3 + 28c_2^2c_4 + 33c_2c_3^2 - 13c_2c_5 - 17c_3c_4 + 5c_6) + O(e_n^7),
 \end{aligned} \tag{20}$$

公式20同时也证明了牛顿法是二阶收敛的, 截断误差为 $\frac{f''(\alpha)}{2f'(\alpha)}e_n^2 + O(e_n^3)$ 。继续对 $f'(y_n)$ 在 α

处做泰勒展开有（代码第 77 行）：

$$\begin{aligned} f'(y_n) &= f'(\alpha) + [c_2 e_n^2 + (2c_2^2 - 2c_3)e_n^3 + O(e_n^4)]f''(\alpha) + \cdots + O(e_n^6) \\ &= 1 + 2c_2^2 e_n^2 + e_n^3 (-4c_2^3 + 4c_2 c_3) + e_n^4 (8c_2^4 - 11c_2^2 c_3 + 6c_2 c_4) \\ &\quad + e_n^5 (-16c_2^5 + 28c_2^3 c_3 - 20c_2^2 c_4 + 8c_2 c_5) + O(e_n^6), \end{aligned} \quad (21)$$

将公式16与公式21相加得到 z_n 的分母部分的泰勒展开：

$$f'(x_n) + f'(y_n) = 2f'(\alpha) \left[1 + c_2 e_n + \left(c_2^2 + \frac{3}{2} c_3 \right) e_n^2 + \cdots + O(e_n^6) \right], \quad (22)$$

因此将公式22与公式14相除得到 z_n 的泰勒展开：

$$x_n - \frac{2f(x_n)}{f'(x_n) + f'(y_n)} = \alpha + e_n - \frac{2 \left[\sum_{k=1}^6 c_k e_n^k + O(e_n^7) \right]}{1 + c_2 e_n + \left(c_2^2 + \frac{3}{2} c_3 \right) e_n^2 + \cdots + O(e_n^6)}, \quad (23)$$

使用与公式18相似的技巧，对分母部分进行泰勒展开得（代码第 82 行）：

$$\begin{aligned} x_n - \frac{2f(x_n)}{f'(x_n) + f'(y_n)} &= \alpha + e_n^3 \cdot \left(c_2^2 + \frac{c_3}{2} \right) + e_n^4 \cdot \left(-3c_2^3 + \frac{3c_2 c_3}{2} + c_4 \right) \\ &\quad + e_n^5 \cdot \left(6c_2^4 - 9c_2^2 c_3 + 2c_2 c_4 - \frac{3c_3^2}{4} + \frac{3c_5}{2} \right) \\ &\quad + e_n^6 \cdot \left(-9c_2^5 + 25c_2^3 c_3 - 15c_2^2 c_4 - \frac{5c_2 c_3^2}{2} + \frac{5c_2 c_5}{2} - \frac{5c_3 c_4}{2} + 2c_6 \right) \\ &\quad + O(e_n^7), \end{aligned} \quad (24)$$

公式24同时也证明了序列 $\{z_n\}_{n=0}^\infty$ 是三阶收敛的，即使用一阶导数的前向差商来计算二阶导数的 Halley 方法是仍是三阶收敛的。继续计算 $f(z_n)$ 和 $f'(z_n)$ 的泰勒展开，使用与公式14-20相同的方法，可以得到 x_{n+1} 的泰勒展开（代码第 88 行）：

$$x_{n+1} = \alpha + e^6 \cdot \left(c_2^5 + c_2^3 c_3 + \frac{c_2 c_3^2}{4} \right) + O(e^7), \quad (25)$$

因此证明了本方法是六阶收敛的，截断误差为 $e^6 \cdot \left(c_2^5 + c_2^3 c_3 + \frac{c_2 c_3^2}{4} \right) + O(e^7)$

2.3 一些有趣的发现

由于使用 sympy 可以很方便地进行泰勒展开来对收敛阶进行推导，我又尝试了一下修改迭代格式，使用与 $\{z_n\}_{n=0}^\infty$ 相似的格式来代替最后一步的牛顿法：

$$\begin{cases} y_n &= x_n - \frac{f(x)}{f'(x)} \\ z_n &= x_n - \frac{2f(x_n)}{f'(x_n) + f'(y_n)}, \\ x_{n+1} &= z_n - \frac{2f(z_n)}{f'(y_n) + f'(z_n)} \end{cases} \quad (26)$$

附录 2 的代码第 94 行展示了收敛性证明的计算过程, 使用与公式21-25类似的方法, 可以得到 x_{n+1} 的泰勒展开:

$$x_{n+1} = e^5 \cdot \left(c_2^4 + \frac{c_2^2 c_3}{2} \right) + e^6 \cdot \left(-5c_2^5 + \frac{5c_2^3 c_3}{2} + c_2^2 c_4 + c_2 c_3^2 \right) + \alpha + O(e^7), \quad (27)$$

因此公式26只有 5 阶收敛, 但它的计算量比牛顿法还多 (同样需要计算 $f(z_n)$ 和 $f'(z_n)$, 且需要额外计算一次加法和乘法), 进一步让我感受到了牛顿法无可比拟的优势。

2.4 计算效率讨论

在 [13] 中定义了迭代法的效率指数 (Efficiency index) 为:

$$EI = q^{1/\omega}, \quad (28)$$

其中 q 是收敛阶, ω 是每次迭代需要计算的函数的次数 (包括其导数)。例如牛顿法二阶收敛, 需要计算一次 $f(x_n)$ 和一次 $f'(x_n)$, 因此效率指数为 $EI = 2^{1/2}$ 。

表1列举了本方法和部分方法的效率指数, 和方法中额外引入的乘除法次数, 这些方法也将会在实验中进行验证。可以看到, 本方法具有相对简单的迭代格式, 尽管效率指数略低于现有方法, 但引入额外的乘除法次数较少, 且减少了一次计算 $f(x)$, 仍然具有一定的实用价值。

表 1: 一些迭代法的计算效率

方法	收敛阶	计算次数			效率指数	额外乘除法次数
		$f(x)$	$f'(x)$	$f''(x)$		
牛顿法	2	1	1	0	$2^{1/2} \approx 1.414$	1
Halley 法 (公式11)	3	1	1	1	$3^{1/3} \approx 1.442$	4
NM[5] (公式8)	6	3	1	0	$6^{1/4} \approx 1.567$	7
GM[6] (公式9)	6	3	1	0	$6^{1/4} \approx 1.567$	5
本方法 (公式13)	6	2	3	0	$6^{1/5} \approx 1.431$	3

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.
- [13] A.M. Ostrowski. *Solutions of equations and system of equations*. Academic Press, 1960.

附录

附录 1

对化简 Halley 方法二阶导数的代码

```
1  import sympy as sp
2
3  fx = sp.symbols('fx')      #  $f(x)$ 
4  dx = sp.symbols('dx')      #  $f'(x)$ 
5  x = sp.symbols('x')        #  $x$ 
6  dy = sp.symbols('dy')      #  $f'(y)$ 
7
8  y = x - fx / dx             #  $y$ 
9  ddx = (dy - dx) / (y - x)  #  $f''(x)$ 
10 halley = x - (2 * fx * dx) / (2 * dx * dx - fx * ddx)
11
12 print(sp.simplify(halley - x))
```

附录 2

对收敛性证明的代码

```
1  # %%
2  import sympy as sp
3
4  # %%
5  c2, c3, c4, c5, c6 = sp.symbols('c2 c3 c4 c5 c6')
6  c1 = 1
7  alpha = sp.symbols('alpha')
8  e = sp.symbols('e')
9  x = alpha + e
10 e1 = e
11 e2 = e ** 2
12 e3 = e ** 3
13 e4 = e ** 4
```

```
14 e5 = e ** 5
15 e6 = e ** 6
16
17 # %%
18 def taylor_1plus_theta(the, order):
19     inv = 1 / (1 + the)
20     return inv.series(e, 0, order)
21
22 def get0(the, order):
23     return taylor_1plus_theta(the, order) - taylor_1plus_theta(the, order).remove0()
24
25 o3 = get0(e, 3)
26 o4 = get0(e, 4)
27 o6 = get0(e, 6)
28 o7 = get0(e, 7)
29 o6, o7
30
31 # %%
32 def df(err, expand=False):
33     eee = lambda x: x
34     if expand:
35         eee = sp.expand
36     return eee(1 + 2*c2*err + 3*c3*err**2 + 4*c4*err**3 + 5*c5*err**4 + 6*c6*err**5 + o6)
37
38
39 def f(err):
40     return c1*err + c2*err**2 + c3*err**3 + c4*err**4 + c5*err**5 + c6*err**6 + o7
41
42
43 def taylor_inv(fenmu, order):
44     inv = 1 / fenmu
45     return 1 / inv.series(e, 0, order)
46
```

```
47 def texify(algo):
48     sp.print_latex(sp.collect(sp.expand(algo), e))
49     print()
50
51     # %%
52     fx = f(e)
53     fx
54
55     # %%
56     dx = df(e)
57     dx
58
59     # %%
60     print(" 公式 18")
61     texify(1 / taylor_inv(df(e), 6))
62
63     # %%
64     fx_div_dx = sp.collect(sp.expand(fx / taylor_inv(df(e), 6)), e)
65     print(" 公式 19")
66     texify(fx_div_dx)
67
68     # %%
69     y = x - fx_div_dx
70     print(" 公式 20")
71     texify(y)
72
73     # %%
74     dy = df(y - alpha, True)
75
76     # %%
77     print(" 公式 21")
78     texify(sp.collect(dy, e))
79
```

```
80 # %%
81 z = x - 2 * fx / taylor_inv(dx + dy, 6)
82 print(" 公式 24")
83 texify(sp.collect(sp.expand(z), e))
84
85 # %%
86 fz = f(z-alpha)
87 dz = df(z-alpha)
88 print(" 公式 25")
89 texify(sp.expand(z - fz / taylor_inv(dz, 6)))
90
91 # %%
92 fy = f(y - alpha)
93 newx = z - 2 * fz / taylor_inv(dz + dy, 6)
94 print(" 公式 27")
95 texify(sp.collect(sp.expand(newx), e))
96
97
```